# Survival Prediction Using Gene Expression Data

**Zhu Wang**
University of Tennessee Health Science Center

**Abstract**

This document describes applications of R package **bujar** for predicting survival in diffuse large B cell lymphoma treated with chemotherapy plus Rituximab using gene expression data.

*Keywords*: survival, Buckley-James regression, prediction, boosting, variable selection.

## 1. Introduction

Researchers have been interested in predicting survival in diffuse large B cell lymphoma (DLBCL) treated with chemotherapy. We re-evaluate clinical and microarray data in Lenz *et al.* (2008). Data from two treatment plans were collected: CHOP and R-CHOP. CHOP is a combination chemotherapy with cyclophosphamide, doxorubicin, vincristine and prednisone. The current gold standard includes rituxima immunotherapy in addition to the chemotherapy (R-CHOP). It was interesting to identify genes that predict survival among patients who received CHOP also retain their prognostic power among patients who received R-CHOP. Data from 181 CHOP patients (training data) are used to build predictive models and data from 233 R-CHOP patients (test data) are used to validate the models. Due to the nature of high-dimensional data with 54675 probe sets or covariates, we first conduct a pre-selection procedure on the training data. This procedure filters out genes with lower variations if a sample variance for a gene was smaller than the 10th percentile for that gene. Details on how the data were read and pre-screened may be found in `dlbcl_raw.R` in the vignettes directory. Afterwards we have 3833 remaining probe sets which were stored in the R package **bujar** (Wang 2015). Test data with the same remaining genes is used for validation.

One of the challenges is that we have right censored data, as often occurred in survival data analysis. Although proportional-hazards regression is a popular option, we focus on Buckley-James (BJ) regression for high-dimensional data proposed by Wang and Wang (2010). Essentially, BJ regression iteratively imputes the censored survival outcomes using the conditional expectation obtained from the current estimates, and the imputed survival outcomes are refitted with regression techniques, which in turn lead to different BJ predictive models depend on what regression model is chosen. We evaluate BJ boosting algorithms and BJ penalized regression implemented in **bujar**. To avoid overfitting, predictive modeling typically requires tuning parameter(s) selection. Tuning parameter is selected by data-driven cross-validation, unless otherwise specified. However, if twin boosting is conducted, the boosting iteration number is fixed in the second round of boosting. The analysis results can be different from Wang and Wang (2010) for two reasons. First, computer codes have been changed for BJ

boosting with componentwise least squares or smoothing splines as base learner. In the current implementation since version 0.1-10, every BJ iteration uses the optimal boosting iteration obtained from the last BJ iteration if tuning parameter is selected. The change results in more sparse variable selection and also consistent with BJ boosting with trees as base learner. Second, BJ boosting (not twin boosting) with trees involves an internal random mechanism for tree model building. In the current implementation since version 0.1-10, a new parameter `rng` in function `bujar` can be used for reproducible results.

```
R> library("bujar")
R> data("chop")
R> data("rchop")
R> ###censoring rate in the CHOP data
R> sum(chop$status==0)/nrow(chop)

[1] 0.4199

R> rchop <- subset(rchop, select=colnames(rchop)%in% colnames(chop))
R> chop$survtime <- chop$survtime + 1 ### add 1 for log-transformation
```

# 2. BJ Boosting with Linear Regression

## 2.1. BJ boosting with componentwise least squares (BJ-LS)

```
R> set.seed(123)
R> res.lin <- bujar(y=log(chop[,1]), cens=chop[,2], x=chop[,-(1:2)], tuning=TRUE,
 cv=TRUE, n.cores=1, mstop=1000)
R> ###number of genes selected with BJ-LS
R> sum(res.lin$xselect==1)
R> coef.bj <- coef(res.lin)
R> ###estimated non-zero coefficients (only list 10)
R> coef.bj[abs(coef.bj)>0][1:10]

R> library("survival")
R> cutyear <- 3

R> pred.bj <- predict(res.lin, newx=rchop[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff
```

```
R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 1: Kaplan-Meier survival curves for BJ-LS regression.

### 2.2. BJ twin boosting with componentwise least squares

```
R> res.lin2 <- bujar(y=log(chop[,1]), cens=chop[,2], x=chop[,-(1:2)], tuning=TRUE,
 cv=FALSE, mstop=1000, twin=TRUE, mstop2=100)
R> ### number of genes selected with BJ-LS
R> sum(res.lin2$xselect==1)
R> coef.bj <- coef(res.lin2)
R> coef.bj[abs(coef.bj)>0]
R> pred.bj <- predict(res.lin2, newx=rchop[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 2: Kaplan-Meier survival curves for twin BJ-LS regression.

To reduce computing burden for other modeling methods, we adopted a supervised gene screening to select the top 100 probe sets based on univariate BJ regression.

```
R> library("rms")
R> res <- rep(NA,ncol(chop))
R> for(i in 3:ncol(chop)){
   #It is possible bj function fails with the following message
   #  Error in exp(fit$stats["g"]) :
   #non-numeric argument to mathematical function
     bjres <- try(bj(Surv(survtime, status) ~ chop[,i],data=chop, link="log"))
   ###if BJ convergence fails, still included for further analysis
   if(inherits(bjres, "try-error")) res[i] <- 1e-5
   else res[i] <- anova(bjres)[1,3]  #p-value
 }
R> nsel <- 100
R> ### select top nsel=100 genes with most significant p-values
R> chop2 <- chop[, c(1, 2, sort.list(res,decreasing=FALSE)[1:nsel])]
R> rchop2 <- rchop[, c(1, 2, sort.list(res,decreasing=FALSE)[1:nsel])]
R> colnames(chop2)[-(1:2)] <- colnames(rchop2)[-(1:2)]  <-
 paste("x",colnames(chop2)[-(1:2)],sep="")
R> detach(package:rms)
```

## 3. BJ LASSO

Within each BJ iteration, the LASSO is used to fit the imputed survival outcomes. The penalty tuning parameter is fixed at the 20th value in 100 penalty sequence values determined within each BJ iteration.

```
R> res.lasso <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="enet2", tuning=FALSE, whichlambda=20)
R> ### how many genes selected by BJ-LASSO
R> sum(res.lasso$xselect==1)
R> ###estimated non-zero coefficients (only list 10)
R> coef.bj <- coef(res.lasso)
R> coef.bj[abs(coef.bj)>0][1:10]
R> pred.bj <- predict(res.lasso, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff
```

```
R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 3: Kaplan-Meier survival curves for BJ-LASSO regression

## 4. BJ SCAD

BJ-SCAD is an extension of BJ-LASSO (Wang and Wang 2010; Fan and Li 2001). Within each BJ iteration, the SCAD is used with the imputed survival outcomes. The penalty tuning parameter is fixed at the 20th value in 100 penalty sequence values determined within each BJ iteration.

```
R> res.scad <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="snet", tuning=FALSE, whichlambda=20)
R> ### how many genes selected by BJ-SCAD
R> sum(res.scad$xselect==1)
R> ###estimated non-zero coefficients (only list 10)
R> coef.bj <- coef(res.scad)
R> coef.bj[abs(coef.bj)>0][1:10]
R> pred.bj <- predict(res.scad, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 4: Kaplan-Meier survival curves for BJ-SCAD regression.

# 5. BJ Boosting with Smoothing Splines

## 5.1. BJ boosting with componentwise smoothing splines (BJ-SM)

```
R> set.seed(123)
R> res.ss <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="pspline", tuning=FALSE, cv=FALSE, mstop=100)
R> ### how many genes selected by BJ smoothing splines, only list 10
R> sum(res.ss$xselect==1)
R> colnames(res.ss$x)[res.ss$xselect==1][1:10]
R> pred.bj <- predict(res.ss, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 5: Kaplan-Meier survival curves for BJ-SM regression.

### 5.2. BJ twin boosting with componentwise smoothing splines

```
R> set.seed(123)
R> res.ss2 <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="pspline", tuning=TRUE, cv=TRUE, n.cores=1, mstop=100, twin=TRUE, mstop2=200)
R> ### how many genes selected by BJ twin smoothing splines, only list 10
R> sum(res.ss2$xselect==1)
R> colnames(res.ss2$x)[res.ss2$xselect==1][1:10]
R> pred.bj <- predict(res.ss2, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 6: Kaplan-Meier survival curves for twin BJ-SM regression.

# 6. BJ Boosting with Regression Trees

## 6.1. BJ boosting with regression stumps (BJ-Tree)

```
R> res.tree1 <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="tree",tuning=TRUE, cv=TRUE, mstop=1000, n.cores=1, rng=123)
R> ###Number of genes selected with tree, only list 10
R> sum(res.tree1$xselect==1)
R> colnames(res.tree1$x)[res.tree1$xselect==1][1:10]
R> pred.bj <- predict(res.tree1, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 7: Kaplan-Meier survival curves for BJ-Tree regression.

### 6.2. BJ twin boosting with regression stumps

```
R> res.tree2 <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="tree", tuning=TRUE, cv=TRUE, mstop=1000, twin=TRUE, mstop2=100,
 n.cores=1, rng=123)
R> ###Number of genes selected with tree, only list 10
R> sum(res.tree2$xselect==1)
R> colnames(res.tree2$x)[res.tree2$xselect==1][1:10]
R> pred.bj <- predict(res.tree2, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 8: Kaplan-Meier survival curves for twin BJ-Tree regression.

### 6.3. BJ boosting with regression trees of degree 4

```
R> res.tree4 <- bujar(y=log(chop2[,1]), cens=chop2[,2], x=chop2[,-(1:2)],
 learner="tree",degree=4, tuning=TRUE, cv=TRUE, mstop=100, rel.inf=TRUE,
 n.cores=1,rng=123)
R> ###Number of genes selected with tree, only list 10
R> sum(res.tree4$xselect==1)
R> colnames(res.tree4$x)[res.tree4$xselect==1][1:10]
R> pred.bj <- predict(res.tree4, newx=rchop2[,-(1:2)])
R> pred.bj <- exp(pred.bj) - 1
R> group <- cut(pred.bj, breaks=c(-1, cutyear, 100), labels=c("high", "low"))
R> dat.km <- data.frame(survtime=rchop$survtime, status = rchop$status, group=group)
R> fit.diff <- survdiff(Surv(survtime, status) ~ group, data=dat.km)
R> fit.diff

R> fit.surv <- survfit(Surv(survtime, status) ~ group, data=dat.km )
R> plot(fit.surv, xlab="Year past therapy",ylab="Survival probability",
 lty = 1:2, col=c("red","blue"), mark.time=TRUE)
R> legend(1, .1, c("High risk", "Low risk"), lty = 1:2, col=c("red","blue"))
```

Figure 9: Kaplan-Meier survival curves for BJ-Tree (degree 4) regression.

Partial dependence plots can be utilized to show the impact of one or more covariates on the response after taking account the average effects of all other covariates in the model. We can generate plots for selected genes. See Figure 5 in Wang and Wang (2010).

```
R> gene <- c("x1558999_x_at", "x212713_at", "x224043_s_at", "x229839_at",
 "x237515_at", "x237797_at", "x242758_x_at", "x244346_at")
R> library("gridExtra")
R> for(i in 1:length(gene))
 eval(parse(text=(paste("a", i, " <- plot(res.tree4$res.fit,
                        i.var=which(colnames(res.tree4$x) == gene[i]))", sep=""))))
R> grid.arrange(a1, a2, a3, a4, a5, a6, a7, a8, ncol=4)
```

Figure 10: Partial plots of selected genes based on BJ-Tree (degree=4) regression.

The two-way interaction partial plots display gene-gene interactions similar to Figure 6 in Wang and Wang (2010).

```
R> for(i in 1:6)
 eval(parse(text=(paste("b", i, " <- plot(res.tree4$res.fit,
                        i.var=unlist(res.tree4$interactions$rank.list[i,c(1, 3)]))",
 sep=""))))
R> grid.arrange(b1, b2, b3, b4, b5, b6, ncol=2)
```

Figure 11: Gene-gene interactions based on BJ-Tree (degree=4) regression.

# 7. SessionInfo

```
R> sessionInfo();

R version 4.5.2 (2025-10-31)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.3 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK versio

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] survival_3.8-6 bujar_0.2-11

loaded via a namespace (and not attached):
 [1] libcoin_1.0-10      farver_2.1.2        R.utils_2.13.0      S7_0.2.1
 [5] fastmap_1.2.0       TH.data_1.1-5       digest_0.6.39       rpart_4.1.24
 [9] lifecycle_1.0.5     cluster_2.1.8.1     magrittr_2.0.4      compiler_4.5.2
[13] rlang_1.1.7         Hmisc_5.2-5         tools_4.5.2         partykit_1.2-24
[17] plotrix_3.8-13      data.table_1.18.2.1 knitr_1.51          htmlwidgets_1.6.4
[21] RColorBrewer_1.1-3  earth_5.3.5         multcomp_1.4-29     polspline_1.1.25
[25] R.cache_0.17.0      foreign_0.8-91      numDeriv_2016.8-1.1 sys_3.4.3
[29] R.oo_1.27.1         stats4_4.5.2        nnet_7.3-20         grid_4.5.2
[33] lars_1.3            colorspace_2.1-2    ggplot2_4.0.2       scales_1.4.0
[37] iterators_1.0.14    MASS_7.3-65         cli_3.6.5           inum_1.0-5
[41] mvtnorm_1.3-3       rmarkdown_2.30      rms_8.1-0           rstudioapi_0.18.0
[45] stringr_1.6.0       modeltools_0.2-24   splines_4.5.2       nnls_1.6
[49] parallel_4.5.2      base64enc_0.1-6     vctrs_0.7.1         glmnet_4.1-10
[53] Matrix_1.7-4        mpath_0.4-2.26      sandwich_3.1-1      SparseM_1.84-2
[57] Formula_1.2-5       htmlTable_2.4.3     maketools_1.3.2     foreach_1.5.2
[61] glue_1.8.0          WeightSVM_1.7-16    stabs_0.7-1         plotmo_3.7.0
[65] codetools_0.2-20    mboost_2.9-11       stringi_1.8.7       mda_0.5-5
[69] shape_1.4.6.1       gtable_0.3.6        quadprog_1.5-8      elasticnet_1.3
```

```
[73] htmltools_0.5.9     quantreg_6.1        pscl_1.5.9          gbm_2.2.3
[77] R6_2.6.1            doParallel_1.0.17   evaluate_1.0.5      lattice_0.22-7
[81] R.methodsS3_1.8.2   backports_1.5.0     class_7.3-23        MatrixModels_0.5-4
[85] Rcpp_1.1.1          bst_0.3-24          R.rsp_0.46.0        gridExtra_2.3
[89] nlme_3.1-168        checkmate_2.3.4     xfun_0.56           zoo_1.8-15
[93] buildtools_1.0.0
```

## References

Fan J, Li R (2001). "Variable selection via nonconcave penalized likelihood and its oracle properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360.

Lenz G, Wright G, Dave SS, Xiao W, Powell J, Zhao H, Xu W, Tan B, Goldschmidt N, Iqbal J, Vose J, Bast M, Fu K, Weisenburger DD, Greiner TC, Armitage JO, Kyle A, May L, Gascoyne RD, Connors JM, Troen G, Holte H, Kvaloy S, Dierickx D, Verhoef G, Delabie J, Smeland EB, Jares P, Martinez A, Lopez-Guillermo A, Montserrat E, Campo E, Braziel RM, Miller TP, Rimsza LM, Cook JR, Pohlman B, Sweetenham J, Tubbs RR, Fisher RI, Hartmann E, Rosenwald A, Ott G, Muller-Hermelink H, Wrench D, Lister TA, Jaffe ES, Wilson WH, Chan WC, Staudt LM (2008). "Stromal gene signatures in large-B-cell lymphomas." *New England Journal of Medicine*, **359**(22), 2313–2323.

Wang Z (2015). *bujar: Buckley-James regression for survival data with high-dimensional covariates*. R package version 0.2-5, URL http://CRAN.R-project.org/package=bujar.

Wang Z, Wang CY (2010). "Buckley-James boosting for survival analysis with high-dimensional biomarker data." *Statistical Applications in Genetics and Molecular Biology*, **9**(1), Article 24.

### Affiliation:

Zhu Wang
Department of Preventive Medicine
University of Tennessee Health Science Center
E-mail: zwang145@uthsc.edu