Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Survival Analysis in R: A Tutorial

David M Diez
Harvard Biostatistics

Abstract

This tutorial introduces the **survival** package in R to readers with a passing familiar with survival analysis and basic familiarity with objects and plotting in R. This tutorial begins with the basics of building survival objects all the way up through fitting Cox proportional hazards for time-varying covariates.

Keywords: survival analysis, `survival` package, R, tutorial, OpenIntro.

1. Introduction

Survival analysis is the study of time-to-event data. This document is intended to assist individuals knowledgable in the basics of survival analysis and interested in applying survival analysis in R with the **survival** package (Therneau and original R port by Thomas Lumley 2009). Familiarity with data objects, plotting, and linear models in R is assumed. The reader who plans to print this document will find reading easiest if this paper is printed front-to-back and bound like a book. Under this construction, topics running two pages will be fully visible without the turn of a page.

Each section introduces a new set of survival analysis tools in R, and these topics are introduced in the following order. The **survival**, **OIsurv**, and **KMsurv** packages are introduced in Section 2. Methods for constructing survival objects are introduced in Section 3. The nonparametric Kaplan-Meier estimate of the survival curve, its pointwise bounds, and confidence bands are introduced in Sections 4 and 5. Section 6 introduces the cumulative hazard function. Estimation of the mean, median, and their corresponding confidence intervals are described in Section 7. Techniques for conducting simple tests on survival data are introduced in Section 8. Sections 9 and 10 introduce methods for fitting Cox proportional hazards models when covariates are constant and time-varying, respectively. Accelerated failure-time models are outlined and discussed in Section 11.

2. Three packages: survival, OIsurv, and KMsurv

Methods and data sets from the **survival**, **OIsurv**, and **KMsurv** packages in R are utilized throughout this document (Therneau and original R port by Thomas Lumley 2009; Diez 2011; Klein, Moeschberger, and modifications by Jun Yan 2010). The **survival** package is a plentiful supply of survival analysis functions and is one of the packages that comes with a standard R installation. The **OIsurv** and **KMsurv** packages offer supplemental functions and data. Both packages can be installed through the `install.packages` function. Installing **OIsurv** will install both packages.

```
> install.packages('OIsurv')
```

Similarly, loading the **OIsurv** package using the `library` function will load other dependent packages, including **survival** and **KMsurv**.

```
> library(OIsurv)  # the survival package depends on the splines package
Loading required package: survival
Loading required package: splines
Loading required package: KMsurv
```

To view available data sets in the **KMsurv** package, use `library(help=KMsurv)`; these data sets are featured in Klein and Moeschberger (2003). To load a data set, use the `data` function, and to view the help file of a data set or function, use the `?` command:

```
> data(aids)
> aids
    infect induct adult
1     0.00   5.00     1
...
295   7.25   0.25     0
> ?aids    # opens help document for aids data set
> ?OIsurv  # OIsurv's help file examples include all code in this tutorial
```

The '...' denotes output omitted for brevity. The code of any individual section of this tutorial may be run when the **survival**, **OIsurv**, and **KMsurv** packages are loaded into an R session.

For brevity in code, this document will make columns from a data frame accessible as free-standing variables via the `attach` function.

```
> attach(aids)
> infect
  [1] 0.00 0.25 0.75 0.75 0.75 1.00 1.00 1.00 1.00 1.25 1.25 1.25 1.25 1.50
...
[295] 7.25
```

Detaching the data set after the analysis can prevent errors, as some data frames share column (variable) names.

```
> detach(aids)
```

Unless otherwise noted, functions and data sets introduced and used in this tutorial come from the **survival** and **KMsurv** packages, respectively.

3. Survival objects

Many functions in the **survival** package apply methods to objects (variables) of class `"Surv"`. The `Surv` function is used to put data into this proper format. Here we discuss how to construct right-censored, left-censored, and interval censored objects of class `"Surv"`. For a review of these data types, see ReliaSoft Corporation website (2006) or Klein and Moeschberger (2003). The `Surv` function has four commonly used arguments: `time`, `time2`, `event`, and `type`.

Right-censored data occur when each case is tracked and either the event time is known and observed, or it has not occurred up to some time where observation is ended. Formatting right-censored data using `Surv` requires only two arguments. The first argument represents the time and the second is an indicator specifying whether the time is for an event (`1`) or a right-censored (`1`) observation.

```
> data(tongue)
> attach(tongue)
> mySurvObject <- Surv(time, delta)
> mySurvObject
 [1]   1    3    3    4   10   13   13   16   16   24   26   27
...
[73] 129  181    8+  67+  76+ 104+ 176+ 231+
> detach(tongue)
```

The '+' symbol is used to signify right-censoring.

Left-censoring occurs when the event is known to have occurred before some time. For these data, the `time`, `event`, and `type` arguments are used to specify **left-censoring** in the `Surv` function. The first two arguments are identical to those for right-censoring, and the `type` argument must be specified as `"left"`:

```
> # Surv(time, event, type="left")
```

Interval-censoring occurs when the event is known to lie in some interval. Here we introduce one technique for creating interval-censored data that will be applied in Section 10 for time-dependent covariates. Suppose we encounter two types of individuals:

1. those who have an event between two times, where those times may be distinct for each individual, and
2. individuals who are observed for an interval of time but the event is not observed, i.e. these observations are right-censored.

Then we might summarize the intervals for these two groups as $(t_1, t_2]$ and $(t_1, t_2]+$, respectively. We can construct such data in R by specifying the t_1 and t_2 times as `time` and `time2`, and also specifying the event argument:

```
> # Surv(t1, t2, event)
```

In this code, `t1` and `t2` are vectors representing t_1 and t_2.

There are additional ways to construct survival data, which are described in the help file for the `Surv` function. For instance, a second type of interval censoring can be accomplished using `type="interval"`.

4. Kaplan-Meier estimate and pointwise bounds

The Kaplan-Meier estimate is a nonparametric maximum likelihood estimate (MLE) of the survival function, $S(t) = P(T > t)$. The estimate is a step function with jumps at observed event times, t_i. The description below assumes the t_i are ordered: $0 < t_1 < t_2 < \cdots < t_D$. If the number of individuals with an observed event time t_i is d_i, and Y_i is the number of individuals at risk – i.e. no event has occurred for these cases – at a time before t_i, then the Kaplan-Meier estimate of the survival function and its estimated variance is given by

$$\begin{aligned} \hat{S}(t) &= \begin{cases} 1 & \text{if } t < t_1 \\ \prod_{t_i \le t} \left[1 - \frac{d_i}{Y_i}\right] & \text{if } t_1 \le t \end{cases} \\ \widehat{V}[\hat{S}(t)] &= \left[\hat{S}(t)\right]^2 \hat{\sigma}^2_S(t) = \left[\hat{S}(t)\right]^2 \sum_{t_i \le t} \frac{d_i}{Y_i(Y_i - d_i)} \end{aligned}$$

One estimate of the pointwise confidence bounds is given by

$$\hat{S} \pm Z_{1-\alpha/2} \hat{\sigma}_S(t) \hat{S}(t) \tag{1}$$

The Kaplan-Meier estimate is fit in R using the function `survfit` function from the **survival** package. The only required argument is a formula where the survival object is the response. To fit the estimated survival curve of a set of data, the survival object is regressed against `1`:

```
> data(tongue)
> attach(tongue)
> mySurv <- Surv(time[type==1], delta[type==1])
> (myFit <- survfit(mySurv ~ 1))
Call: survfit(formula = mySurv ~ 1)

records   n.max n.start  events  median 0.95LCL 0.95UCL
     52      52      52      31      93      67      NA
```

The confidence level may be changed using an argument called `conf.int` (e.g. `conf.int=0.90` for 90% confidence bounds). The `conf.type` argument describes the type of confidence interval. More specifically, it describes the transformation for constructing the confidence interval. The default is `'log'`, which equates to the transformation function $g(t) = \log(t)$. The `'log-log'` option uses $g(t) = \log(-\log(t))$. A linear confidence interval fitting the description of Equation (1) is created using the `conf.type='plain'` option. At this time, the arcsine-squareroot transformation is not an available option, but it can be computed manually using the elements of the summary of an object returned by `survfit`.

Like the `lm` function in R, we can generate a summary of the output of `survfit`.

```
> summary(myFit)
Call: survfit(formula = mySurv ~ 1)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    1     52       1    0.981  0.0190        0.944        1.000
    3     51       2    0.942  0.0323        0.881        1.000
...
  167      4       1    0.229  0.0954        0.101        0.518
```

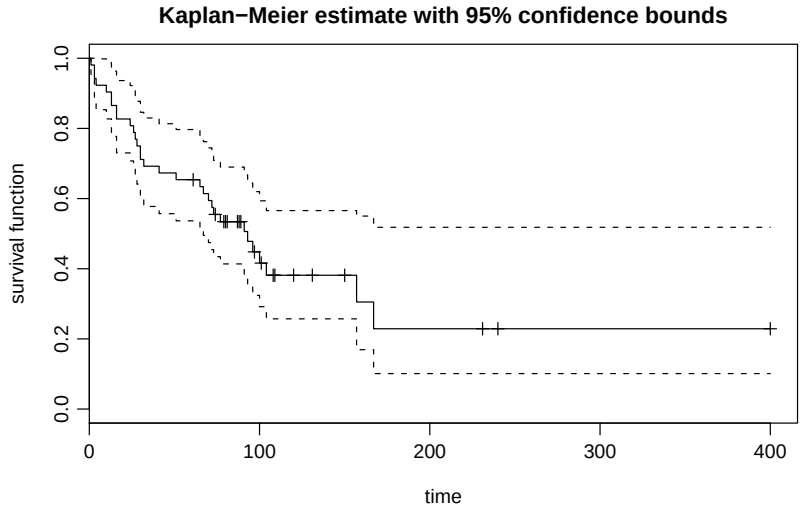

Figure 1: Sample output of `plot(myFit, ...)`, where the title, x-axis and y-axis labels have been specified via the `main`, `xlab`, and `ylab` arguments, respectively.

Also just like `lm`, the output of `survfit` is a list, and each item in the list may be accessed using the `$` operator. We consider many of these elements below; the actual output of each is hidden for brevity.

```
> myFit$surv     # outputs the Kaplan-Meier estimate at each t_i
> myFit$time     # {t_i}
> myFit$n.risk   # {Y_i}
> myFit$n.event  # {d_i}
> myFit$std.err  # standard error of the K-M estimate at {t_i}
> myFit$lower    # lower pointwise estimates (alternatively, $upper)
```

The `plot` method may also be applied to the Kaplan-Meier estimate from the `surv.fit` function, as shown in Figure 1. The familiar arguments in the plot function may be used to improve the graphical aesthetics:

```
> plot(myFit, main="Kaplan-Meier estimate with 95% confidence bounds",
+      xlab="time", ylab="survival function")
```

Sometimes different groups are mixed together in a single `Surv` object. If a separate vector specifies which cases correspond to which groups, then these groups can be separated by regressing the `Surv` object on that vector:

```
> myFit1 <- survfit(Surv(time, delta) ~ type)  # 'type' specifies the grouping
```

The list output by the summary of `myFit1` will contain an additional list item called `strata` – accessible via `summary(myFit1)$strata` – which designates which components of the output correspond to which groups.

Finally, for good coding practices, we detach the `tongue` data set.

```
> detach(tongue)
```

5. Kaplan-Meier confidence bands

The confidence interval bands constructed on the previous pages are only pointwise confidence intervals. General confidence bands can also be constructed using the `confBands` function from the **OIsurv** package. These bands provide bounds on an entire range of time.

We start by constructing the regular confidence interval bands, just as did in the last section:

```
> data(tongue)
> attach(tongue)
> mySurv <- Surv(time[type==1], delta[type==1])
> plot(survfit(mySurv ~ 1), xlab='time',
+      ylab='Estimated Survival Function',
+      main='Confidence intervals versus confidence bands')
```

Next the `confBands` function is used to generate an object of class `"confBands"`, which is a list of three items: `time`, `lower`, and `upper`. These values describe the confidence bands, and they are impacted by a number of options within the `confBands` function. The only required element of the function is a survival object.

```
> myCB <- confBands(mySurv)
> lines(myCB, lty=3)
> legend('topright', legend=c('K-M survival estimate',
+      'pointwise intervals','EP confidence bands'), lty=1:3)
> detach(tongue)
```

Users may opt to specify more precise types of confidence bands. For instance, the confidence bands can be constructed using different transformations using the `confType` argument, or a different confidence level may be specified with `confLevel`. These additional options and others are described in the `confBands` help file.

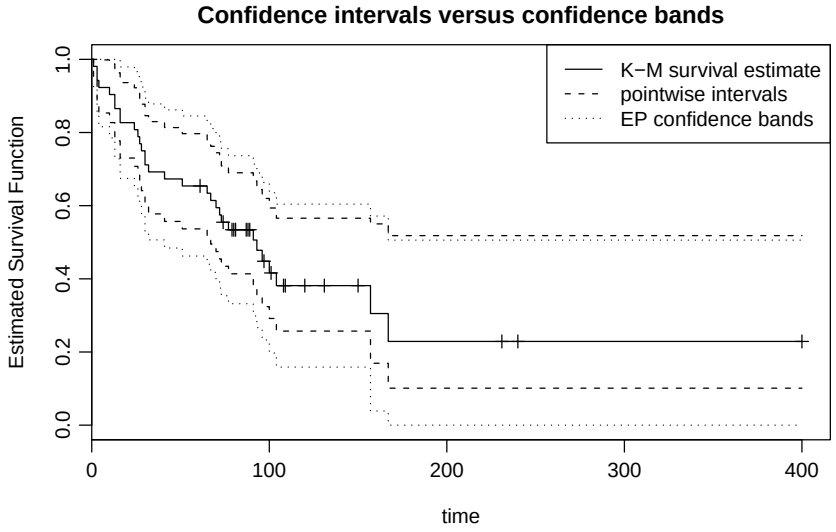

Figure 2: Confidence intervals and bans. The default for the pointwise confidence bands is a log transformation, which results in non-symmetric pointwise confidence intervals.

6. Cumulative hazard

The relationship between the cumulative hazard function and the survival function is shown in the left side of Equation (2). Thus, the MLE estimate of the cumulative hazard function may be estimated by transforming the Kaplan-Meier estimate:

$$S(t) = \exp\{-H(t)\} \qquad \hat{H}(t) = -\log \hat{S}(t) \tag{2}$$

Another method to estimate $H(t)$ is the Nelson-Aalen estimator:

$$\widetilde{H}(t) = \sum_{t_i \le t} \frac{d_i}{Y_i} \text{ (assuming } t_1 \le t \text{, otherwise it is 0)}, \qquad \hat{\sigma}^2_{\widetilde{H}}(t) = \sum_{t_i \le t} \frac{d_i}{Y_i^2}$$

No function in the **survival** package computes either form of the cumulative hazard function, but this can be accomplished using output from `survfit()`:

```
> data(baboon)
> attach(baboon)
> mySurv  <- Surv(time, observed)
> myFit   <- summary(survfit(mySurv ~ 1))
> Hhat    <- -log(c(myFit$surv, tail(myFit$surv, 1)))
```

A plot of both the MLE and Nelson-Aalen curves may also be constructed:

```
> hSortOf <- myFit$n.event / myFit$n.risk
> Htilde  <- c(cumsum(hSortOf), sum(hSortOf))
> plot(c(myFit$time, 1200), Hhat, xlab='time', ylab='cumulative hazard',
+      main='Comparing cumulative hazards', type='s')
> lines(c(myFit$time, 1200), Htilde, lty=2, type='s')
> legend('topleft', legend=c('MLE', 'Nelson-Aalen'), lty=1:2)
> detach(baboon)
```

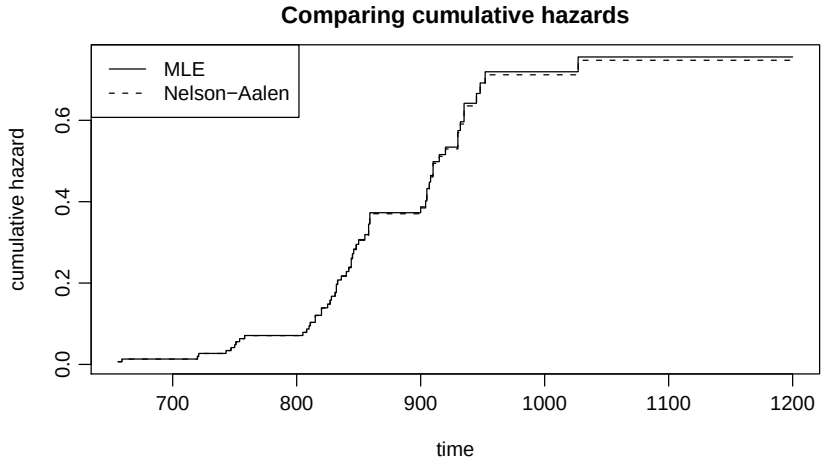

Figure 3: Estimates of the cumulative hazard function via MLE and Nelson-Aalen methods.

7. Mean and median estimates with bounds

The median survival time is the time $t_{0.5}$ such that $S(t_{0.5}) = 0.5$, visualized by graphing the survival function estimate, drawing a horizontal line at 0.5, and identifying the time at which the survival curve and line intersect. The confidence bounds for $t_{0.5}$ are given by the points at which this horizontal line crosses over the confidence bounds of $\hat{S}(t)$. The median and its bounds may be estimated using the `survfit` function:

```
> data(drug6mp)
> attach(drug6mp)
> mySurv <- Surv(t1, rep(1, 21))   # all placebo patients observed
> survfit(mySurv ~ 1)
Call: survfit(formula = mySurv ~ 1)

records   n.max n.start  events  median 0.95LCL 0.95UCL
     21      21      21      21       8       4      12
```

The mean survival time and its respective estimate are given by

$$\mu = \int_0^\infty S(t)dt \qquad\qquad \hat{\mu} = \int_0^\infty \hat{S}(t)dt$$

If $S(t)$ (or $\hat{S}(t)$) does not converge to zero, the integral diverges. To avert this disaster, it is common to use a finite value τ as the bound for the integral, creating a biased estimate of the mean, shown in the left side of Equation (3). For instance, one might choose τ as the largest observed or censored time. Using the values t_i, Y_i, d_i, and D defined in Section 4, the estimated variance of $\hat{\mu}_\tau$ is given in the right side of Equation (3).

$$\mu_\tau = \int_0^\tau S(t)dt \qquad\qquad \hat{V}(\hat{\mu}_\tau) = \sum_{i=1}^{D} \left[\int_{t_i}^\tau \hat{S}(t)dt\right]^2 \frac{d_i}{Y_i(Y_i - d_i)} \tag{3}$$

The mean survival time and its standard error are accessed through the `survfit` function and `print` method:

```
> print(survfit(mySurv ~ 1), print.rmean=TRUE)
Call: survfit(formula = mySurv ~ 1)

   records       n.max     n.start      events      *rmean *se(rmean)
     21.00       21.00       21.00       21.00        8.67        1.38
    median     0.95LCL     0.95UCL
      8.00        4.00       12.00
    * restricted mean with upper limit =  23
> detach(drug6mp)
```

The `print.rmean=TRUE` argument is used to obtain the mean and its standard error, and τ is automatically set as the largest observed or censored time. Alternatively, τ may be specified using the `rmean` argument. Additional details on this printing function may be found in the `print.survfit` help file.

8. Tests for two or more groups

Given two or more samples, is there a difference between the survival times? Hypotheses for this problem may be built around the hazard functions, assumed to be continuous:

- $H_0 : h_1(t) = h_2(t) = \cdots = h_n(t)$ for all t.
- $H_A : h_i(t_0) \neq h_j(t_0)$ for at least one pair i, j and time t_0.

Define the following variables:

- t_i represents times where events are observed, where these values are assumed to be ordered with the index i and that there are D such times;
- d_{ik} is the number of observed events from group k at time t_i;
- Y_{ik} is the number of subjects in group k that are at risk at time t_i;
- $W(t_i)$ be the weight of the observations at time t_i.

A vector Z is computed, which will be used to compute the test statistic. The k^{th} element of Z is given by

$$Z_k = \sum_{i=1}^{D} W(t_i) \left[d_{ik} - Y_{ik} \frac{d_{i\bullet}}{Y_{i\bullet}} \right]$$

A bullet ($\bullet$) is used in place of an index to represent the sum over that index, e.g. $d_{i\bullet} = \sum_{j=1}^{n} d_{ij}$. The covariance matrix $\widehat{\Sigma}$ is also computed from the data (see Klein and Moeschberger (2003)). Under the null hypothesis, the test statistic $X^2 = Z'\hat{\Sigma}^{-1}Z$ follows a χ^2 distribution with n degrees of freedom. That is, if $X^2 > \chi^2_{1-\alpha,df=n}$, the data provides convincing evidence against null hypothesis (we reject H_0) at significance level α.

The `survdiff` function is used for the grunt work of the hypothesis test. The first argument is an equation regressing a survival object against a categorical covariate variable, such as a variable representing treatment groups in a drug trial.

```
> data(btrial)
> attach(btrial)
> survdiff(Surv(time, death) ~ im)   # output omitted
```

The `survdiff` function provides the χ^2 statistic and a corresponding p-value.

A second optional argument, `rho`, designates the weights $W(t_i)$ according to $\hat{S}(t)^\rho$. The default setting is `rho=0`, which corresponds to weights of one and the log-rank test. To emphasize the first part of the survival curve in the test, specify a `rho` greater than zero. To weight the later component of the survival curve more, specify a negative `rho`. The **Peto & Peto modification of the Gehan-Wilcoxon test** is computed using one for `rho`:

```
> survdiff(Surv(time, death) ~ im, rho=1)   # some output omitted
...

 Chisq= 4.4  on 1 degrees of freedom, p= 0.037
> detach(btrial)
```

9. Cox proportional hazards model, constant covariates

The Cox proportional hazards (PH) model is shown in the left side of Equation (4). The *baseline hazard function* $h_0(t)$ is assumed to vary by a multiplied constant $e^{\beta' z}$ for an individual with covariate values z. The right side of Equation (4) represents the equivalent model using a survival function.

$$h(t|z) = h_0(t) \exp\left\{\beta' z\right\} \qquad \hat{S}(t|z_k) = [S_0(t)]^{\exp(\beta' z_k)} \tag{4}$$

where β is a vector of unknown parameters and z is a vector representing covariates. (It is common for the baseline hazard or survival function to be estimated using nonparametric methods.) Researcher's primary interest lies in identifying β, estimated by solving the partial likelihood:

$$L(\beta) = \prod_{i=1}^{D} \frac{\exp\left[\beta' z_{(i)}\right]}{\sum_{j \in R(t_i)} \exp\left\{\beta' z_j\right\}}, \; R(t_i) \text{ is the 'risk set' at time } t_i$$

The MLE $\hat{\beta}$ (a vector) is normally distributed in the limit with mean β. This framework allows for **local testing**, where a single test statistic is used to assess null values for many elements of β simultaneously, much like how an ANOVA is sometimes used to simultaneously assess whether several group means are equal, or tests for each coefficient separately. We consider the more general case of a local test. We assume the null hypothesis take the form $C\beta = d$, where C is a $q \times p$ matrix of full rank and d is a vector of length q. For instance, C may be chosen as the identity matrix and d the zero vector, which leads to a test considering whether β itself is the zero vector. A test statistic for the local test is

$$X_W^2 = \left(C\hat{\beta} - d\right)' \left[C\hat{I}^{-1}C'\right]^{-1} \left(C\hat{\beta} - d\right),$$

where $\hat{I}^{-1}$ is the estimated covariance matrix of $\hat{\beta}$. Under the null hypothesis, X_W^2 follows a χ_q^2 distribution. The null is rejected if X_W^2 is in the upper α tail of this distribution, where α is the significance level. Formally this is known as the **Wald test**.

A Cox PH model is fit using the `coxph` function. The first argument is a formula, regressing a survival object on the covariates:

```
> data(burn)
> attach(burn)
> mySurv    <- Surv(T1, D1)
> (coxphFit <- coxph(mySurv ~ Z1 + as.factor(Z11)))
Call:
coxph(formula = mySurv ~ Z1 + as.factor(Z11))


                   coef exp(coef) se(coef)     z     p
Z1                0.512     1.668    0.208  2.46 0.014
as.factor(Z11)2 -0.892     0.410    0.498 -1.79 0.073
as.factor(Z11)3 -1.677     0.187    0.802 -2.09 0.037
as.factor(Z11)4 -0.406     0.667    0.395 -1.03 0.300

Likelihood ratio test=15.4  on 4 df, p=0.00402  n= 154
```

Two covariates have been used in this example. Additional information regarding the Cox PH model is found using the `summary` method. For instance, the risk ratio with confidence bounds is obtained for each parameter, and p-values for likelihood ratio, Wald and score tests for the global null, $H_0 : \beta_i = 0$ for all i.

The `coxph` function uses the Efron method by default. Another commonly used technique is the Breslow method, which may be accessed through the `method` argument in the `coxph` function.

More complex hypotheses may be checked using other elements from the model fit:

```
> coxphFit$coefficients  # may use coxphFit$coeff instead
> coxphFit$var           # I^(-1), estimated cov matrix of the beta-hats
> coxphFit$loglik        # log-likelihood for alt and null MLEs, resp.
```

The baseline survival function is obtained by applying `survfit` to the output of `coxph`:

```
> mySurvfit <- survfit(coxphFit)
```

The object `mySurvfit` is the familiar output of `survfit`. For instance, the `plot` method may be applied to the output to see the **baseline** survival function with its bounds.

A local test may be accomplished by extracting the estimate of β and its estimated covariance matrix, I^{-1}:

```
> betaHat <- coxphFit$coef
> betaCov <- coxphFit$var
```

Matrices C and d can also be constructed for a local test.

The `anova` function may be used for simple tests assessing whether a factor variable should be included in the model. The `Z11` variable is a factor with four levels, and the inclusion of this variable in the model may be weighed through a test using an analysis-of-deviance table:

```
> anova(coxphFit)
Analysis of Deviance Table
 Cox model: response is mySurv
Terms added sequentially (first to last)

                loglik  Chisq Df Pr(>|Chi|)
NULL           -420.48
Z1             -416.86 7.2439  1   0.007114 **
as.factor(Z11) -412.80 8.1114  3   0.043765 *
...
> detach(burn)
```

The p-value is less than the typical significance level, $\alpha = 0.05$, so we reject the null hypothesis that the variability in the factor estimates is due to chance alone. We might say there was convincing evidence that `Z11` is associated with survival according to this model.

10. Cox PH model, time-dependent covariates

Last section we considered modeling the survival function using covariates that remained constant for each case. However, some covariates do not remain constant within the same observational unit. There may be a change in environmental factors, an intervention, or a behavioral change. For instance, whether a person smokes or not is an important covariate in some drug trials. However, smokers quit and others take up the habit. One way to model these changes is through time-varying covariates.

Using time-dependent covariates in R is an exercise in organization. We will use censoring and truncation liberally. For example, if there is an intervention for patient i, then we split patient i into two separate observations: pre and post-intervention. More explicitly, suppose the patient intervention took place at time $t_i = 45$ and the event for patient i was observed at $t_{event} = 58$. Then in R, we would split this patient's record into two pieces: 0 to 45 and 45 to 58. The first interval will be right-censored, and the second interval will be left-*truncated*.

We consider the following example, a simulated data set from the `OIsurv` package. Patient records were obtained for 150 days after they joined a rehabilitation program. The event of interest was drug-relapse and two covariates were recorded. The `event` variable describes the observed or censored time; the `delta` variable describes whether the time denotes an observed relapse (`TRUE`) or a censored time; the `gender` variable is a time-independent covariate; and `inter` is a time-dependent covariate indicating whether the patient was (randomly) assigned a second intervention: working 10 hours a week for a nonprofit. Each of these special interventions were assigned *after* the patients entered the clinic, meaning the intervention covariate changes for those patients who had an intervention before relapse.

```
> data(relapse)
> relapse
    event delta gender inter
1     150 FALSE      0    84
2      53  TRUE      1    50
3      12  TRUE      1    NA
4     150 FALSE      0    89
5     150 FALSE      1    77
6     135  TRUE      1     7
7     150 FALSE      0    21
...
298    62  TRUE      0    24
299    10  TRUE      0    NA
300    94  TRUE      1     4
```

We model this data within R using a two-step procedure:

1. Construct survival records that may include left-truncation or right-censoring. The survival record of each patient with an intervention is broken into two survival records: one before the intervention and one after.

2. The new survival records can be analyzed with the `coxph` function.

The first step is programming intensive. The approach taken below is the construction of a new set of records. The row for each individual with an intervention is broken into two

separate intervals. Rows representing individuals who did not have the intervention remain unchanged.

```
> attach(relapse)
> N  <- dim(relapse)[1]
> t1 <- rep(0, N+sum(!is.na(inter)))  # Initialize start times at 0
> t2 <- rep(NA, length(t1))           # The end times for each record
> e  <- rep(NA, length(t1))           # Was the event censored?
> g  <- rep(NA, length(t1))           # Gender
> PI <- rep(FALSE, length(t1))        # Initialize intervention at FALSE
>
> R  <- 1                           # Row of new record
> for(ii in 1:dim(relapse)[1]){
+   if(is.na(inter[ii])){         # no intervention, copy survival record
+     t2[R] <- event[ii]
+     e[R]  <- delta[ii]
+     g[R]  <- gender[ii]
+     R <- R+1
+   } else {                    # intervention, split records
+     g[R+0:1] <- gender[ii]  # gender is same for each time
+     e[R]     <- 0           # no relapse observed pre-intervention
+     e[R+1]   <- delta[ii]   # relapse occur post-intervention?
+     PI[R+1]  <- TRUE        # Intervention covariate, post-intervention
+     t2[R]    <- inter[ii]-1 # End of pre-intervention
+     t1[R+1]  <- inter[ii]-1 # Start of post-intervention
+     t2[R+1]  <- event[ii]   # End of post-intervention
+     R <- R+2                # Two records added
+   }
+ }
```

The code above has created many new R variables. The `t1` variable represents the start of each interval; `t2` the end of each interval; `e` indicates if the patient relapsed; `g` is the gender covariate; and `PI` is the indicator variable for whether this interval is associated with a patient post-intervention.

While patients had time-varying covariates, the new intervals do not. With all covariates now constant in each interval, the `coxph` function may be used to fit the model:

```
> mySurv   <- Surv(t1, t2, e)
> coxphFit <- coxph(mySurv ~ g + PI)
```

This example was a simple case: there was a single time-dependent covariate, and it changed at most once per case. In some instances, there may be many time-varying covariates, even some that change every time unit. In most of these instances, the same 2-step technique may be applied. Whenever a covariate changes from one time unit to the next, split the interval into two and use censoring.

For a more extensive review of Cox PH models in R, including a more extensive discussion of time-varying covariates, see Fox (2002).

11. Accelerated failure-time models

An accelerated failure-time (AFT) model is a parametric model with covariates and failure times follow the survival function $S(x|Z) = S_0\left(x * \exp\left[\theta' Z\right]\right)$, where S_0 is a function for the baseline survival rate. The term $\exp\left[\theta' Z\right]$ is called the **acceleration factor**. The AFT model uses covariates to place individuals on different time scales – note the scaling by the covariates in $S(t|Z)$ via $\exp\left[\theta' Z\right]$. The AFT model can be rewritten to a log-linear form, where the log of failure time $\log X$ is linearly related to the mean μ, the acceleration factor, and an error term σW:

$$\log X = \mu - \theta' Z + \sigma W$$

where W describes the error distribution. The following models for W are discussed in Klein and Moeschberger (2003):

distribution	df	included in **survival**?
exponential	1	yes
Weibull	2	yes
lognormal	2	yes
log logistic	2	yes
generalized gamma	3	no

The function `survreg()` function from the **survival** package is used for AFT modeling. The first argument is `formula` or a survival object regressed on predictors. The argument `dist` has several options to describe the parametric model used (`'weibull'`, `'exponential'`, `'gaussian'`, `'logistic'`, `'lognormal'`, and `'loglogistic'`). The example code below follows Example 12.2 from Klein and Moeschberger (2003):

```
> data(larynx)
> attach(larynx)
> srFit <- survreg(Surv(time, delta) ~ as.factor(stage) + age, dist='weibull')
> summary(srFit)
...
                    Value Std. Error      z        p
(Intercept)        3.5288     0.9041  3.903 9.50e-05
as.factor(stage)2 -0.1477     0.4076 -0.362 7.17e-01
as.factor(stage)3 -0.5866     0.3199 -1.833 6.68e-02
as.factor(stage)4 -1.5441     0.3633 -4.251 2.13e-05
age               -0.0175     0.0128 -1.367 1.72e-01
Log(scale)        -0.1223     0.1225 -0.999 3.18e-01

Scale= 0.885

Weibull distribution
Loglik(model)= -141.4   Loglik(intercept only)= -151.1
Chisq= 19.37 on 4 degrees of freedom, p= 0.00066
Number of Newton-Raphson Iterations: 5
...
```

In the output, `(Intercept)` and `Log(scale)` correspond to estimates of μ and $\log \sigma$. The other estimates correspond to covariate coefficients. We might compare this to a result for the exponential model, where the scale is fixed at 1:

```
> srFitExp <- survreg(Surv(time, delta) ~ as.factor(stage) + age, dist='exponential')
> summary(srFitExp)
...
                    Value Std. Error      z        p
(Intercept)        3.7550     0.9902  3.792 1.49e-04
as.factor(stage)2 -0.1456     0.4602 -0.316 7.52e-01
as.factor(stage)3 -0.6483     0.3552 -1.825 6.80e-02
as.factor(stage)4 -1.6350     0.3985 -4.103 4.08e-05
age               -0.0197     0.0142 -1.388 1.65e-01

Scale fixed at 1

Exponential distribution
Loglik(model)= -141.9   Loglik(intercept only)= -151.1
Chisq= 18.44 on 4 degrees of freedom, p= 0.001
Number of Newton-Raphson Iterations: 4
...
```

The Weibull model with $\sigma = 1$ is equivalent to the exponential model. We consider two strategies for selecting which of the two models is more appropriate:

- A likelihood ratio test, which evaluates the null hypothesis $\sigma = 1$ against the two-sided alternative.
- Examination of the significance of the `Log(scale)` coefficient (see the output to `summary(srFit)`).

In the example, both approaches result in the same conclusion: there is insufficient evidence to reject the notion that $\sigma = 1$ (H_0).

Interested users may explore the many stored components in a `survreg()` object:

```
> # the output is omitted from each command below
> srFitExp$coeff    # covariate coefficients
> srFitExp$icoef    # intercept and scale coefficients
> srFitExp$var      # variance-covariance matrix
> srFitExp$loglik   # log-likelihood
> srFit$scale       # not using srFitExp (defaulted to 1)
> detach(larynx)
```

References

Diez DM (2011). *Survival analysis tutorial*. R package version 0.1, URL http://CRAN.R-project.org/package=OIsurv.

Fox J (2002). "Cox Proportional-Hazards Regression for Survival Data. Appendix to An R and S-PLUS Companion to Applied Regression." *Comprehensive R Archive Network*. URL http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-cox-regression.pdf.

Klein, Moeschberger, modifications by Jun Yan (2010). *KMsurv: Data sets from Klein and Moeschberger (1997), Survival Analysis*. R package version 0.1-4, URL http://CRAN.R-project.org/package=KMsurv.

Klein JP, Moeschberger ML (2003). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer Verlag, New York.

ReliaSoft Corporation website (2006). "Data Classification." URL http://www.weibull.com/LifeDataWeb/data_classification.htm.

Therneau T, original R port by Thomas Lumley (2009). *survival: Survival analysis, including penalised likelihood*. R package version 2.35-8, URL http://CRAN.R-project.org/package=survival.

Affiliation:

David M Diez
Department of Biostatistics
Harvard School of Public Health
655 Huntington Avenue
SPH2, 4th Floor
E-mail: david.m.diez@gmail.com
URL: http://www.openintro.org/