Appendix A
A Basic Guide to Using R for Survival Analysis

A.1 The R System

This first section of the appendix provides a brief but necessarily incomplete introduction to the R system. Readers with little prior exposure to R can start here, and then follow up with one of the many books or online guides to the R system. Succeeding sections cover specialized R topics relevant to using R for survival analysis.

The R statistical system, which henceforth we will refer to as just "R", is a programming language geared to doing statistical analyses. It was created by Ross Ihaka and Robert Gentleman at the University of Auckland in New Zealand and, since 1997, has been maintained by a core group of about twenty developers in diverse locations. More information about the R system and its maintenance may be found at the website http://cran.r-project.org. It is an interpretative language, meaning that it interprets and executes code as the user types it, or as it reads code from a file. It includes features for creating and manipulating variables, vectors and matrices. It can also work with the more advanced structures known as data frames, arrays and lists. It also has facilities for creating plots, and it has a special format for handling missing values, which are a common feature of data sets. Its facilities overlap with those of widely used commercial statistical software packages. Like them, it provides a wide range of statistical procedures, and includes facilities for manipulating statistical data. Unlike those packages, however, R is "open source," meaning essentially that the code is freely available and free to distribute. There are, however, certain licensing restrictions, a key one being that any code derived from existing R code must also be made freely available. The absence of any cost to downloading and installing R has led to widespread worldwide adoption of the system, and encouraged researchers who develop new statistical methods to make those methods available in R.

D. Moore, *Applied Survival Analysis Using R*, Use R!,
DOI 10.1007/978-3-319-31245-3

To install the system in Microsoft Windows, go to http://cran.r-project.org/bin/windows/base/ and follow the instructions. The Windows installer will create an icon on the desktop or the start menu. (On a 64-bit Windows installation, two icons will be created, one for a 32-bit version and one for a 64-bit version. The latter version will include "x64" in the icon name.) R is also available for Apple OS and Linux OS; see the main R site at http://cran.r-project.org for details.

To start R, click on the icon (for now, either the 32- or 64-bit version is fine) as you would with any standard Windows program. A command window will open up with a ">" prompt. The command window includes several dropdown menus for opening or creating files of R commands, for installing "packages" to provide additional functionality to the system, and for accessing the help system. While one can carry out any R procedure from the base system, more advanced users may prefer to use a separate editor for creating R programs. A free editor that works well with R is "Tinn-R", which may be downloaded from https://sourceforge.net/projects/tinn-r. Alternatively, one can use a full-fledged programming environment called Rstudio. When launched, R Studio automatically detects the most current version of R on your system, and opens that up in one of four window panes. Other panes include an editor for writing R code, a viewer that shows the names of objects you have created, and a plot viewer. Rstudio may be downloaded from http://www.rstudio.com/ide/download/desktop.

This guide provides a brief introduction to those aspects of R most useful for survival analysis. A guide to more complete treatments of the R language may be found at http://www.r-project.org/doc/bib/R-books.html.

A.1.1 A First R Session

The purpose of this session is to show how to start up R, carry out a few simple numerical operations, and then exit the system. First, start R from the start menu of Windows or from the Desktop. You will see the R Console, a window with a ">" prompt. This is the R window into which you type commands and receive responses. To get a feel for how R works, enter a numerical expression, such as this:

```
> 2 + 3
[1] 5
```

The ">" symbol is the prompt that R provides, and following that is "2 + 3", which the user types. The "[1]", which is perhaps superfluous here, just indicates that the printed result is the first (and in this simple case only) element of the result. (The purpose of the output format will become clear when you look at long vectors that stretch out over more than one line.)

Now try some other operations. Note that the "#" symbol indicates the start of a "comment", and is not interpreted by the computer.

```
> 2^3
[1] 8
> 2**3  # same thing
[1] 8

> x <- 3    # assign 3 to the variable x
> y <- 2
> y^x
[1] 8

> y**x
[1] 8

> q()   # end the R session
```

In addition to constants, R can work with vectors, as shown in the following code:

```
> x.vec <- c(1, 3.5, 7)
> y.vec <- c(2, 7, 8.6)
> x.vec
[1] 1.0 3.5 7.0
> y.vec
[1] 2.0 7.0 8.6
```

Here we have defined two vectors, "x.vec" and "y.vec", each of length 3. The ".vec" ending of the names is for the convenience of the user only; any name that consists of letters and numbers and certain separators such as "." or "_" can be use. Beginners should note that unlike in some other statistical systems, R is case sensitive; that is, upper- and lower-case letters are interpreted as distinct. Thus, for example, "X.vec" and "x.vec" are two different names.

Vectors may be added and multiplied, as long as they are the same length. Also, constants may multiply vectors.

```
> x.vec
[1] 1.0 3.5 7.0
> y.vec
[1] 2.0 7.0 8.6
> x.vec + y.vec
[1]  3.0 10.5 15.6

> z.vec <- 2*y.vec
> z.vec
[1]  4.0 14.0 17.2
```

The "c()" function may also be used to combine vectors,

```
> z.vec <- c(x.vec, y.vec)
> z.vec
[1] 1.0 3.5 7.0 2.0 7.0 8.6
```

and individual components of vectors may be accessed by index. For example, to list the first four elements of z.vec, we can use the ":" to get the indices from 1 to 4,

```
> z[1:4]
[1] 1.0 3.5 7.0 2.0
```

Vectors may also contain characters,

```
> w.vec <- c("a", "A", "aBc")
> w.vec
[1] "a"   "A"   "aBc"
```

In survival analysis, there is a special structure for right-censored survival data. To use this, one first must load the "survival" package, which is included in the main R distribution,

```
library(survival)
```

Next, define the survival times "tt" and the censoring indicator "status", where "status = 1" indicates that the time is an observed event, and "status = 0" indicates that it is censored. Then the "Surv" function binds them into a single object. In the following example, time 6 is right censored, while the others are observed event times,

```
> tt <- c(2, 5, 6, 7, 8)
> status <- c( 1, 1, 0, 1, 1)
> Surv(tt, status)
[1] 2  5  6+ 7  8
```

If you enter an R command that is syntactically incomplete, it will continue onto the next line with a "+" symbol, where you can complete the command. For example,

```
> tt <- c(2, 5, 6,
+   7, 8)
> tt
[1] 2 5 6 7 8
```

This feature is particularly convenient for long commands that will not fit on a single line.

The character "#" is used to introduce a comment; anything written after this character will be ignored by the R system. This is useful for annotating code, e.g.

```
> Surv(tt, status)  # Create a survival data structure
```

Finally, to quit the R session, use the "q()" function, with no arguments,

```
> q()
```

A.1.2 Scatterplots and Fitting Linear Regression Models

We use linear regression methods in survival analysis in a number of ways, so we introduce it along with the plot function here. To illustrate, let's create two vectors,

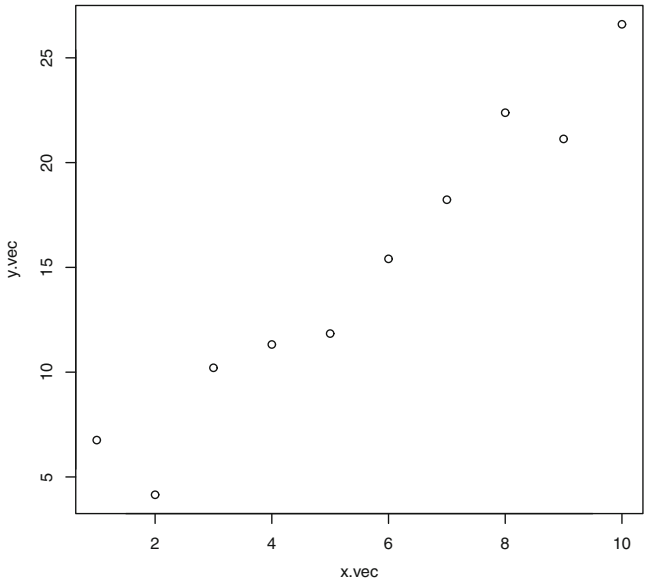
```
> x.vec <- 1:10
> x.vec
[1]  1  2  3  4  5  6  7  8  9 10
>
> y.vec <- 3 + 2*x.vec + rnorm(10, mean=0, sd=2)
> y.vec
[1]  6.758104  4.148936 10.208296 11.320387 11.840879 15.407382
    18.228854
[8] 22.381251 21.130111 26.600463
```

The vector "x.vec" was created using the ":" operator to obtain the integers from 1 to 10, and the vector "y.vec" is defined as

$$y = 3 + 2x + \varepsilon,$$

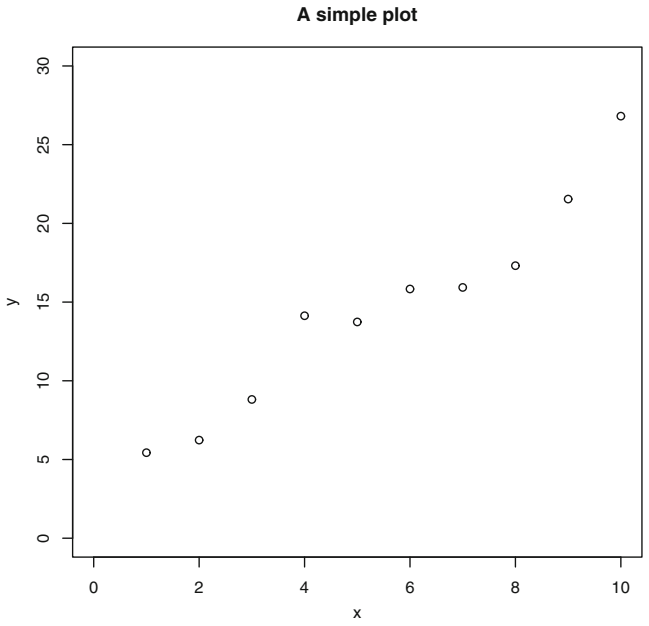
where ϵ is a standard normal random variable with mean 0 and standard deviation 2. Notice that when "y.vec" is printed, its values wrap onto the second line. the "[8]" on the second line indicates that this line begins with the 8th element of the vector. One may easily plot y.vec vs. x.vec,

```
plot(y.vec ~ x.vec)
```



The plot may be enhanced by specifying ranges for the x and y variables, and with specific labels for the axes,

```
> plot(y.vec ~ x.vec, xlim=c(0, 10), ylim=c(0, 30),
+   xlab="x", ylab="y")
> title("A simple plot")
```

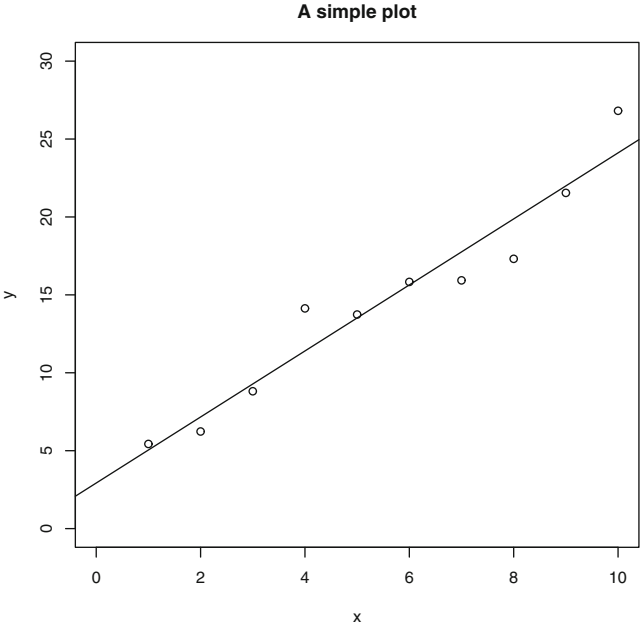


To fit a linear regression line through these points, use the "lm" function, with "y.vec" as the outcome variable, "x.vec" as the predictor variable, and "~" meaning "regressed on". In the following, the results of fitting a linear regression model are put into a "data structure" which we have chosen to call "result.lm". To print out a brief summary of the structure, just type its name,

```
> result.lm <- lm(y.vec ~ x.vec)
> result.lm
Call: lm(formula = y.vec ~ x.vec)
Coefficients:
(Intercept)        x.vec
      2.925        2.119
```

The output indicates that the fitted regression model is given by $y = 2.925 + 2.119x$. To plot this fitted line on the above scatterplot, use the "abline" function,

```
> abline(result.lm)
```



A more complete output of the linear regression, including standard errors and hypothesis tests, may be obtained by entering "summary(result.lm)".

A.1.3 Accommodating Non-linear Relationships

A non-linear relationship between y and x will require more sophisticated tools. To illustrate, let us suppose that the true relationship between y and x is given by $y = 2x^3 - 9x^2 + 5x + 6$. We may define this as an R function as follows:

```
ff <- function(x) {
  result <- 2*x^3 - 9*x^2 + 5*x + 6
  result
  }
```

This function, which we have named "ff", takes a value (or a vector of values), evaluates the defined polynomial at those values, and puts the result in an R object named "result". The last object in the function ("result") is the value that the function returns. For example, to evaluate the function at 0, 1, and 2, we can do the following:

```
> ff(x=c(0, 1, 2))
[1]  6  4 -4
```

We simulate points with this relationship, with error, by defining the relationship as a function, creating a series of x values, and then the y values, as follows:

```
> x.vec <- (-99:400)/100    # create 500 points between -1 and 4
> y.vec <- ff(x.vec) + 10*rnorm(500)   # fixed and random effects
```

We may plot these points, and the "true" functional relationship, shown as a red curve, as follows:

```
> plot(y.vec ~ x.vec, col="gray")
> curve(ff, from=-1, to=4, col="red", lwd=2, add=T)
```

A straightforward way to model such a non-linear relationship is to create quadratic and cubic forms of the x-values, and incorporate them into a linear model as follows:

```
> x2.vec <- x.vec^2
> x3.vec <- x.vec^3
> result.lm <- lm(y.vec ~ x.vec + x2.vec + x3.vec)
> summary(result.lm)

             Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.9369     0.7911   8.769  < 2e-16 ***
x.vec         4.1090     0.9969   4.122  4.4e-05 ***
x2.vec       -9.1537     0.9079 -10.082  < 2e-16 ***
x3.vec        2.0945     0.1936  10.818  < 2e-16 ***
--- Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 10.23 on 496 degrees of freedom
```

We see that the coefficient estimates closely match the originals, as does the "Residual standard error", which matches $\sigma = 10$ in the original error function.

In this case, the "true" relationship was a cubic polynomial function. But if the relationship between y and x were some other function, not necessarily a polynomial one, what could we do? There is a technique called "locally weighted scatterplot smoothing", often abbreviated by "loess" for, presumably, "LOcal rESSion". This is implemented in R as the "loess" function. We may use this function as follows:

```
result.smooth <- loess(y.vec ~ x.vec)
smooth.estimates <- predict(result.smooth)
lines(smooth.estimates ~ x.vec, col="blue", lwd=2)
```

However, we shall find it helpful to plot not only the smooth function but also 95 % confidence intervals. To do this, we define a function that fits a loess curve and also 95 % confidence intervals for this curve, and plots them:

```
smoothSEcurve <- function(yy, xx) {
  # use after a call to "plot"
  # fit a lowess curve and 95% confidence interval curve

  # make list of x values
  xx.list <- min(xx) + ((0:100)/100)*(max(xx) - min(xx))

  # Then fit loess function through the points (xx, yy)
  #   at the listed values
  yy.xx <- predict(loess(yy ~ xx), se=T,
     newdata=data.frame(xx=xx.list))
```

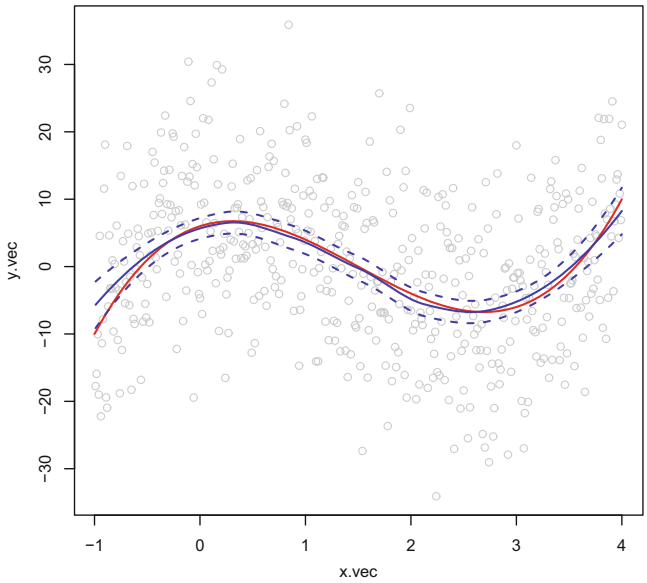


Fig. A.1 Smooth loess curve (*blue*) and true functional relationship (*red*)

```
  lines(yy.xx$fit ~ xx.list, lwd=2)
  lines(yy.xx$fit -
       qt(0.975, yy.xx$df)*yy.xx$se.fit ~ xx.list, lty=2)
  lines(yy.xx$fit +
       qt(0.975, yy.xx$df)*yy.xx$se.fit ~ xx.list, lty=2)
  }
```

We use this function to add the smooth curve and confidence limits as follows:

```
smoothSEcurve(y.vec, x.vec)
```

The plot is shown in Fig. A.1.

A.1.4 Data Frames and the Search Path for Variable Names

R provides a special data structure, the "data frame", to conveniently store variables for statistical data analysis. A data frame is a two-dimensional array with named columns. Like many R packages, the survival package includes data sets as in the form of data frames to use as examples. The data set "lung" contains survival data on 228 patients with advanced lung cancer; here is a subset of the first six rows and seven columns:

```
> lung[1:6,1:7]
  inst time status age sex ph.ecog ph.karno
1    3  306      2  74   1       1       90
2    3  455      2  68   1       0       90
3    3 1010      1  56   1       0       90
4    5  210      2  57   1       1       90
5    1  883      2  60   1       0      100
6   12 1022      1  74   1       1       50
```

The survival variables are "time" (days from enrollment until death or censoring), "status" (1 for censoring, 2 for dead), and a number of other possibly relevant covariates. A detailed description of the data set may be found by typing "?lung" at the R prompt. Individual columns of the data frame may be accessed in two ways: by column number or by column name. Here are examples of accessing the "time" column in these two ways; in each case, the first few components are shown:

```
> time.A <- lung[,2]
> time.B <- lung$time
> time.A[1:5] [1]  306  455 1010  210  883
> time.B[1:5] [1]  306  455 1010  210  883
```

Alternatively, one can "attach" the data frame, in which case all of the variable names can be accessed by name:

```
> attach(lung)
> time[1:5]
[1]  306  455 1010  210  883
```

To avoid errors in referencing variables, it is important to realize that the variable name "time" is placed on what is known as a "search path". When a user types a variable name such as "time", the R system first searches the current workspace for a definition of the variable; finding none, it then looks into any attached data frames; in this case, it is "lung" and there it finds the variable named "time". If we redefine the variable "time" in the workspace, it will take preference:

```
> time <- c(1,2,3,4)
> time
[1] 1 2 3 4
```

Now, the variable "time" has been defined in the workspace as the numbers from 1 to 4. If we remove (i.e. delete) this variable with the "rm" command, this version of the variable goes away, and the version of "time" in the attached "lung" data frame again becomes visible:

```
> rm(time)
> time[1:5]
[1]  306  455 1010  210  883
```

This example illustrates the importance of keeping track of the variable names visible in an attached data frame, and ensuring that no variables of the same name are defined in the user workspace.

A.1.5 Defining Variables Within a Data Frame

When one needs to re-define variables in a data frame, it is often helpful to carry out the necessary calculations within the data frame itself using the "within" function. For example, suppose for the "lung" data we want to define a new censoring variable "delta" which takes the values 0 for a censored variable and 1 for an event. We can do this as follows:

```
lung <- within(lung, {
     delta <- status - 1 })
```

We may see the new variable "delta" as follows:

```
> lung[1:6, c(1:7, 11)]
  inst time status age sex ph.ecog ph.karno delta
1    3  306      2  74   1       1       90     1
2    3  455      2  68   1       0       90     1
3    3 1010      1  56   1       0       90     0
4    5  210      2  57   1       1       90     1
5    1  883      2  60   1       0      100     1
6   12 1022      1  74   1       1       50     0
```

In this way, one can directly incorporate new variables into the data frame without creating new ones in the R workspace.

A.1.6 Importing and Exporting Data Frames

Data frames may be most easily imported and exported using "comma-separated" files. Such files, when saved with a ".csv" extension, will open in Windows as an Excel file by default. Such files can easily be imported into other statistical packages if needed, since the file contains no special-purpose non-printing characters. For example, suppose we need to export the "lung" data. We can export it to a directory, say, "C:\survival" as follows:

```
> setwd("c:\\survival")
> write.csv(lung, file="lung.csv", na=".", row.names=F)
```

Since R treats the backslash character "\" as an escape character (imparting special meaning to certain letters), it cannot be used alone when referring to a Windows directory. Rather, it has to be doubled to correctly reference the windows directory "C:\survival". In this example, the function "setwd" sets the "working directory" to the "C:\survival" Windows folder, assuming that this folder has been created previously outside of the R program. The command "write.csv" writes out the file in comma separated form into the file named "lung.csv". The option "na=" defines the outputted missing value code to be the specified value, here a dot, ".". Without this option, R will write out "NA" (the R missing value code) for missing values. Finally, the "row.names=F" option suppresses numbered row names, which

are usually unnecessary for exported files. The first few rows of the resulting file, if viewed using a text editor, look like this:

```
"inst","time","status","age","sex","ph.ecog","ph.karno","meal.cal"
3,306,2,74,1,1,90,1175
3,455,2,68,1,0,90,1225
3,1010,1,56,1,0,90,.
5,210,2,57,1,1,90,1150
1,883,2,60,1,0,100,.
```

The first row is a list of column names, and the remaining rows contain the data. If this data set weren't already in R, and one needed to import it, one would do the following:

```
> setwd("c:\\survival")
> lung2 <- read.csv("lung.csv", na.strings=".", header=T)
> head(lung2)
  inst time status age sex ph.ecog ph.karno  meal.cal
1    3  306      2  74   1       1       90      1175
2    3  455      2  68   1       0       90      1225
3    3 1010      1  56   1       0       90        NA
4    5  210      2  57   1       1       90      1150
5    1  883      2  60   1       0      100        NA
6   12 1022      1  74   1       1       50       513
```

The option "na.strings=" tells R that, in the Windows file, the "." character indicates missing values, so that they are recognized as such during the import process, and represented using R's missing data indicator "NA". The "header=T" option tells R that the first row consists of column names.

A.2 Working with Dates in R

Often survival data come in the form of calendar dates. Typically we are given the date of entry into a trial, the date of death, and the date a patient was last seen, if still alive. We must then compute the intervening times, and determine if a final date represents a death or a censored observation. With medical data, time is measured in days, although that may be later converted to months or years for presentation purposes. The R package "date", which must be explicitly downloaded and installed, allows us to work with data in date format. To do this from the R window, click on the "Packages" tab to get a pull-down menu. Then click on "Install packages". You may be asked to select a "repository." In that case, choose your country and then a location near you. Then you will get a pop-up window that lists all available packages in alphabetical order. Highlight the "date" package and then "install". The package will then be installed on your R system, and will be available for use in this and future occasions when you use R.

A.2.1 Dates and Leap Years

R includes a special "date" format. When you examine a variable with dates in that format, you will see a listing that is given in "day-month-year" format. Internally, however, the date is stored as an integer that represents the number of days between the date of interest and the reference date, which is January 1, 1960. This reference date is arbitrary, but often used by computer packages; when you subtract date objects, the results will be the number of intervening days between the two dates. The date package is written to accommodate leap years, by including February 29 in calculations only when a leap year is involved. It also understands that the year 2000 was a leap year but that the year 1900 was not a leap year. This latter fact would become relevant if, for example, one works with birth dates of individuals born before 1900. (Leap years in the Gregorian calendar occur in years that are multiples of 4, except for years that are multiples of 100; a further exception is that years that are multiples of 400 *are* leap years, which is why the year 2000 was a leap year. See http://aa.usno.navy.mil/faq/docs/leap_years.php, maintained by the United States Naval Academy, for more details.)

A.2.2 Using the "as.date" Function

Once the date package has been installed, you may load it by typing "library(date)" at the R prompt. The "as.date" function can then be used to convert dates in character form into R dates. Here are some examples of two dates:

```
> date.1 <- as.date("8/31/1956")
> date.2 <- as.date("7/5/1957")
> date.1
[1] 31Aug56
> date.2
[1] 5Jul57
```

We may see that there are 308 days separating these two dates as follows:

```
> date.2 - date.1
[1] 308
```

We may "look inside" the dates to reveal the internally-stored number of days using the "as.numeric" function. For example,

```
> as.numeric(date.1)
[1] -1218
> as.numeric(as.date("1/1/1960"))
[1] 0
```

This shows that the first date, August 31, 1956, is 1,218 days before the reference date of January 1, 1960. Also, we see that the reference date itself is stored as 0.

We may illustrate the leap-year issue as follows:

```
> as.date("2/29/2000")
[1] 29Feb2000
> as.date("2/29/1900")
[1] <NA>
```

This shows that February 29, 2000 is a legitimate date, whereas February 29, 1900 does not exist. (The value "NA" is a missing value indicator in R.)

Dates may also be input in text format:

```
> as.date("January 30 2005")
[1] 30Jan2005
```

The default format is "month-day-year", but it is possible to specify dates in "day-month-year" format using the "order" option:

```
> as.date("30/1/2005", order="dmy")
[1] 30Jan2005
```

Dates may also be vectors. Here is a small example that illustrates what may arise in survival analysis:

```
> entry.dates <- c("9/20/2010", "9/30/2010", "11/2/2010",
                   "1/5/2011")
> death.dates <- c("5/4/2013", NA, "6/9/2013", "4/5/2012")
> lastSeen.dates <- c("5/4/2013", "8/21/2013", "6/9/2013",
                      "4/5/2012")
>
> entry <- as.date(entry.dates)
> death <- as.date(death.dates)
> lastSeen <- as.date(lastSeen.dates)
```

We have defined entry, death, and date last seen dates for four patients. The second patient was known to still be alive as of August 21, 2013, so that person's death date is denoted by the missing value "NA". We define survival and censoring times as follows:

```
> censor <- as.numeric(!is.na(death))
> censor
[1] 1 0 1 1
> survTime.temp <- death - entry
> survTime.temp
[1] 957  NA 950 456
```

We have defined the censoring variable to be 1 if a death is observed and 0 if the person is censored, i.e., still alive at the time the data are analysed. The survival times are defined for all but the second patient; we fix that up as follows:

```
> survTime <- survTime.temp
> survTime[censor == 0] <- lastSeen[censor == 0]
    - entry[censor == 0]
> survTime
[1]  957 1056  950  456
> censor
[1] 1 0 1 1
```

The variables "survTime" and "censor" are now fully-formed survival variable ready for analysis. We may combine them into a survival object using the "Surv" function in the "survival" package,

```
> library(survival)
> Surv(survTime, censor)
[1]   957  1056+  950   456
```

This form of a survival variable show that the second survival time is censored at 1056 days (the time to death, though unknown, *is* known to be larger than 1056 days), whereas the others represent numbers of days until death.

A.3 Presenting Coefficient Estimates Using Forest Plots

The results of fitting a statistical model are typically presented as a table of coefficient names, coefficient estimates, standard errors, Z values, and p-values. Consider for example the survival dataset "veteran" that is included in the survival package. This data set of lung cancer patients consists of survival variable "time", censoring indicator "status", and several covariates, including "trt" (treatment), which takes the values "standard" and "test", and "celltype", which can be either squamous, small cell, adeno, or large cell. In the output below, we re-define "treatment" as a factor with levels "standard" and "treatment" and then fit a Cox proportional hazards model with treatment and cell type as predictors. (See Chaps. 5 and 6 for a discussion of the Cox model and examples of model fitting.)

```
> library(survival)
> head(veteran)
  trt celltype time status karno diagtime age prior
1   1 squamous   72      1    60        7  69     0
2   1 squamous  411      1    70        5  64    10
3   1 squamous  228      1    60        3  38     0
4   1 squamous  126      1    60        9  63    10
5   1 squamous  118      1    70       11  65    10
6   1 squamous   10      1    20        5  49     0

> trt.f <- factor(trt, labels=c("standard", "test"))
> result <- coxph(Surv(time, status) ~ trt.f + celltype,
+   data=veteran)
> result

                   coef exp(coef) se(coef)    z       p
trt.ftest         0.198      1.22    0.197 1.00 3.1e-01
celltypesmallcell 1.096      2.99    0.272 4.02 5.7e-05
celltypeadeno     1.169      3.22    0.295 3.96 7.4e-05
celltypelarge     0.297      1.35    0.286 1.04 3.0e-01
```

The result of the Cox model is put into the data structure named "result". Typing "result" produces the parameter estimates. The coefficient "trt.ftest" is the result of comparing "test" to "standard" therapy. The next three coefficient estimates are for three cell types compared to the reference cell type, which is "squamous".

It is helpful to present these results in graphical form using a display tool called a "forest plot". This type of display was originally developed as a way to present the results of a meta-analysis, which is a type of study that summarizes the results of a large number of related studies. We adapt this display for our purposes, using the "forestplot" function in the package also named "forestplot" (which must be downloaded from CRAN and installed). We set up the parameter estimates and confidence limits as follows; we use "NA"s, empty strings, and extra spaces to control the format of the plot.

```
coef.est <- c(NA, NA, 0, 0.198, NA, NA, NA, 0, 1.096, 1.169, 0.297)
se.est <- c(NA, NA, 0, 0.197, NA, NA, NA, 0, 0.272, 0.295, 0.286)
lower <- coef.est - 1.96*se.est
upper <- coef.est + 1.96*se.est
label.factors <- matrix(c("Treatment Group", "", "  standard",
   "  test", "", "Cell Type", "", "  sqamous", "  smallcell",
   "  adeno", "  large"), ncol=1)
```

Finally, we produce the plot. We use constant box sizes, and the option "txt_gp" to control the label sizes.

```
library(forestplot)
forestplot(label.factors, coef.est,  lower=lower, upper=upper,
      boxsize=0.4, xticks=c(-0.5,0,0.5, 1, 1.5, 2),
      txt_gp=fpTxtGp(label=gpar(cex=1.5)))
```

The resulting plot is shown in Fig. A.2. We can see that the log hazard ratio for the test treatment is slightly positive, indicating a small (non-significant) deleterious

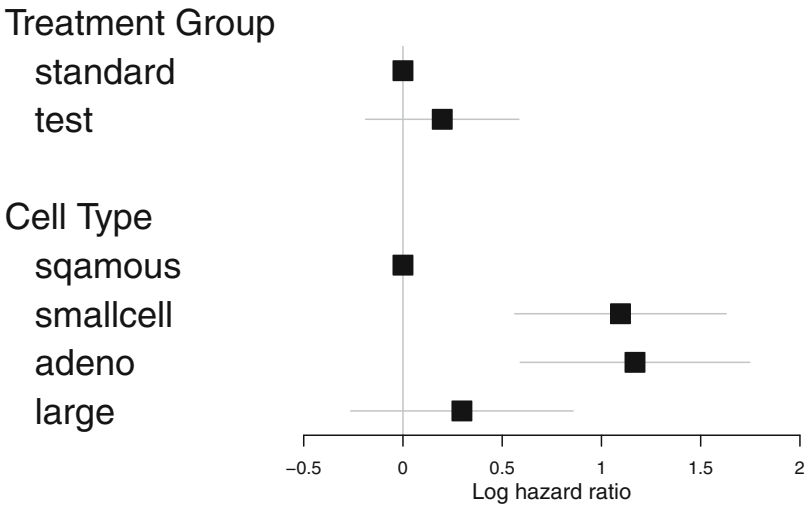


Fig. A.2 Forest plot of parameter estimates for the "veterans" dataset

effect of the test treatment as compared to the standard. We also see that the effect of small cell and adeno celltypes is similar, and larger than squamous (the reference level) and large cell. Of course, if additional covariates are included in the model, they can be included in the plot by direct extension of the code given above.

A.4 Extracting the Log Partial Likelihood and Coefficient Estimates from a coxph Object

As explained in Chap. 5, the log partial likelihood is a crucial component of a Cox model. With appropriate options, we may use the "coxph" function to evaluate the log partial likelihood at a particular value of the parameters. This specialized procedure is not necessary in ordinary data analysis, but is useful for specialized applications and for illustrating concepts, as in Sects. 5.3 and 5.4. We shall illustrate this by again using the "veteran" survival data. For simplicity, we shall define a new variable "treatInd" which is 1 for the test and 0 for the control treatments respectively. Then we fit a Cox model and examine the result:

```
> library(survival)
> attach(veteran)
> testInd <- trt - 1  # now 0 refers to standard, and 1 to test
> result <- coxph(Surv(time, status) ~ testInd)
> result
          coef exp(coef) se(coef)      z    p
testInd 0.0177      1.02    0.181 0.0982 0.92

Likelihood ratio test=0.01  on 1 df, p=0.922
```

We can explicitly evaluate the log partial likelihood at a particular value of the coefficient by specifying the initial value of the coefficient, and blocking iteration by setting the maximum number of iterations to 0:

```
> result.cox.0 <- coxph(Surv(time, status) ~ testInd,
+    init=0, control=list(iter.max=0))
> loglik.0 <- result.cox.0$loglik[2]
> loglik.0
[1] -505.4491
```

The log (partial) likelihood evaluated at $\beta = 0$ is the second element of the "loglik" component, specifically, -505.4491.

To get the log partial likelihood at the maximum, we use the m.p.l.e from the output or, to get a more precise value, we do the following:

```
> coef.mple <- as.numeric(result$coef)
> coef.mple
[1] 0.01774257
```

We may evaluate the log partial likelihood at the maximum and compute the likelihood ratio test statistic, $D = 2\left(l(\hat{\beta}) - l(0)\right)$ as follows:

```
> result.cox.max <- coxph(Surv(time, status) ~ testInd,
+    init=coef.mple, control=list(iter.max=0))
> loglik.max <- result.cox.max$loglik[2]
> 2*(loglik.max - loglik.0)
[1] 0.009643379
```

According to standard statistical theory, this is to be compared to a chi-square distibution with one degree of freedom. We evaluate this using the "pchisq" function:

```
> pchisq(0.009643379, 1, lower.tail=F)
[1] 0.9217729
```

We see that the p-value is approximately 0.92 (the same as given in the standard coxph output above), so that the treatment difference is not statistically significant.

References

1. Aalen, O.: Nonparametric inference for a family of counting processes. Ann. Stat. 701–726 (1978)
2. Agresti A.: Categorical Data Analysis, 3rd edn. Wiley, Hoboken (2012)
3. Andersen, P.K., Borgan, O., Gill, R.D., Keiding, N.: Statistical Models Based on Counting Processes, corrected edition. Springer, New York (1996)
4. Andersen, P.K., Geskus, R.B., de Witte, T., Putter, H.: Competing risks in epidemiology: possibilities and pitfalls. Int. J. Epidemiol. **41**(3), 861–870 (2012)
5. Barker, C.: The mean, median, and confidence intervals of the Kaplan-Meier survival estimate - computations and applications. Am. Stat. **63**(1), 78–80 (2009)
6. Bernstein, D., Lagakos, S.W.: Sample size and power determination for stratified clinical trials. J. Stat. Comput. Simul. **8**(1), 65–73 (1978)
7. Betensky, R.A., Finkelstein, D.M.: A non-parametric maximum likelihood estimator for bivariate interval censored data. Stat. Med. **18**(22), 3089–3100 (1999)
8. Chatterjee, N., Wacholder, S.: A marginal likelihood approach for estimating penetrance from kin-cohort designs. Biometrics **57**(1), 245–252 (2001)
9. Clark, T.G., Bradburn, M.J., Love, S.B., Altman, D.G.: Survival analysis part I: Basic concepts and first analyses. Br. J. Cancer **89**(2), 232–238 (2003)
10. Collett, D.: Modelling Survival Data in Medical Research, 3rd edn. Chapman and Hall/CRC, Boca Raton (2014)
11. Cox, D.R., Oakes, D.: Analysis of Survival Data. Chapman and Hall/CRC, London; New York (1984)
12. Cox, D.R.: Regression models and life-tables. J. R. Stat. Soc. Ser. B Methodol. 187–220 (1972)
13. De Boor, C.: A Practical Guide to Splines. Revised edition (1994)
14. de Wreede, L.C., Fiocco, M., Putter, H., et al.: mstate: an R package for the analysis of competing risks and multi-state models. J. Stat. Softw. **38**(7), 1–30 (2011)
15. Demarqui, F.N., Loschi, R.H., Colosimo, E.A.: Estimating the grid of time-points for the piecewise exponential model. Lifetime Data Anal. **14**(3), 333–356 (2008)
16. Epstein, B., Sobel, M.: Life testing. J. Am. Stat. Assoc. **48**(263), 486–502 (1953)

17. Fay, M.P.: Comparing several score tests for interval censored data. Stat. Med. **18**(3), 273–285 (1999)
18. Finkelstein, D.M.: A proportional hazards model for interval-censored failure time data. Biometrics **42**(4), 845–854 (1986)
19. Fleming, T.R., Harrington, D.P.: Counting Processes and Survival Analysis. Wiley, Hoboken (2011)
20. Freedman, L.S.: Tables of the number of patients required in clinical trials using the logrank test. Stat. Med. **1**(2), 121–129 (1982)
21. Gail, M.H.: Does cardiac transplantation prolong life? A reassessment. Ann. Intern. Med. **76**(5), 815–817 (1972)
22. Gasser, T., Muller, H.-G.: Kernel estimation of regression functions. In: Gasser, T., Rosenblatt, M. (eds.) Smoothing Techniques for Curve Estimation, vol. 757 in Lecture Notes in Mathematics, pp. 23–68. Springer, Berlin, Heidelberg (1979)
23. Goeman, J., Meijer, R., Chaturvedi, N.: L1 and L2 penalized regression models, R package Version 0.9-45, http://cran.r-project.org (2014)
24. Goeman, J.J.: L1 penalized estimation in the Cox proportional hazards model. Biom. J. **52**(1), 70–84 (2010)
25. Grambsch, P.M., Therneau, T.M.: Proportional hazards tests and diagnostics based on weighted residuals. Biometrika **81**(3), 515–526 (1994)
26. Gray, R.J.: Flexible methods for analyzing survival data using splines, with applications to breast cancer prognosis. J. Am. Stat. Assoc. **87**(420), 942–951 (1992)
27. Hall, W.J., Wellner, J.A.: Confidence bands for a survival curve from censored data. Biometrika **67**(1), 133–143 (1980)
28. Harrell, F.E.: Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, 2nd edn. Springer Science & Business Media, New York (2015)
29. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, New York (2009)
30. Hess, K.R., Serachitopol, D.M., Brown, B.W.: Hazard function estimators: a simulation study. Stat. Med. **18**(22), 3075–3088 (1999)
31. Holford, T.R.: The analysis of rates and of survivorship using log-linear models. Biometrics **36**, 299–305 (1980)
32. Hosmer, D.W., Jr., Lemeshow, S., May, S.: Applied Survival Analysis: Regression Modeling of Time to Event Data. Wiley, Hoboken (2008)
33. Huster, W.J., Brookmeyer, R., Self, S.G.: Modelling paired survival data with covariates. Biometrics **45**, 145–156 (1989)
34. Kalbfleisch, J.D., Prentice, R.L.: The Statistical Analysis of Failure Time Data, 2nd edn. Wiley, Hoboken (2002)
35. Kaplan, E.L., Meier, P.: Nonparametric estimation from incomplete observations. J. Am. Stat. Assoc. **53**(282), 457–481 (1958)
36. Klein, J.P., Moeschberger, M.L.: Survival Analysis: Techniques for Censored and Truncated Data, 2nd edn. Springer, New York (2005)
37. Kleinbaum, D.G., Klein, M.: Survival Analysis: A Self-Learning Text, 3rd edn. Springer, New York (2011)
38. Kuhn, M., Johnson, K.: Applied Predictive Modeling. Springer, New York (2013)
39. Lagakos, S.W., Barraj, L.M., De Gruttola, V.: Nonparametric analysis of truncated survival data, with application to aids. Biometrika **75**(3), 515–523 (1988)
40. Laird, N., Olivier, D.: Covariance analysis of censored survival data using log-linear analysis techniques. J. Am. Stat. Assoc. **76**(374), 231–240 (1981)
41. Lee, E.W., Wei, L.J., Amato, D.A., Leurgans, S.: Cox-type regression analysis for large numbers of small groups of correlated failure time observations. In: Klein, J.P., Goel, P. (eds.) Survival Analysis: State of the Art, pp. 237–247. Springer, New York (1992)

42. Li, L., Yan, J., Xu, J., Liu, C.-Q., Zhen, Z.-J., Chen, H.-W., Ji, Y., Wu, Z.-P., Hu, J.-Y., Zheng, L., et al.: CXCL17 expression predicts poor prognosis and correlates with adverse immune infiltration in hepatocellular carcinoma. PloS One **9**(10), e110064 (2014)
43. Li, L., Yan, J., Xu, J., Liu, C.-Q., Zhen, Z.-J., Chen, H.-W., Ji, Y., Wu, Z.-P., Hu, J.-Y., Zheng, L., et al.: Data from: CXCL17 expression predicts poor prognosis and correlates with adverse immune infiltration in hepatocellular carcidata. Dryad Digital Repository, http://datadryad.org (2014)
44. Lin, D.Y.: Cox regression analysis of multivariate failure time data: the marginal approach. Stat. Med. **13**(21), 2233–2247 (1994)
45. Lin, D.Y., Wei, L.J.: The robust inference for the cox proportional hazards model. J. Am. Stat. Assoc. **84**(408), 1074–1078 (1989)
46. Lu-Yao, G.L., Albertsen, P.C., Moore, D.F., et al.: Outcomes of localized prostate cancer following conservative management. J. Am. Med. Assoc. **302**(11), 1202–1209 (2009)
47. Matthews, D.: Exact nonparametric confidence bands for the survivor function. Int. J. Biostat. **9**(2), 185–204 (2013)
48. McCullagh, P., Nelder, J.A., McCullagh, P.: Generalized Linear Models, Vol. 2. Chapman and Hall London, (1989)
49. Moore, D.F., Chatterjee, N., Pee, D., Gail, M.H.: Pseudo-likelihood estimates of the cumulative risk of an autosomal dominant disease from a kin-cohort study. Genet. Epidemiol. **20**(2), 210–227 (2001)
50. Morse, M.A., Niedzwiecki, D., Marshall, J.L., Garrett, C., Chang, D.Z., Aklilu, M., Crocenzi, T.S., Cole, D.J., Dessureault, S., Hobeika, A.C., et al.: A randomized Phase II study of immunization with dendritic cells modified with poxvectors encoding CEA and MUC1 compared with the same poxvectors plus GM-CSF for resected metastatic colorectal cancer. Ann. Surg. **258**(6) (2013)
51. Moss, R.A., Moore, D., Mulcahy, M.F., Nahum, K., Saraiya, B., Eddy, S., Kleber, M., Poplin, E.A.: A multi-institutional phase 2 study of imatinib mesylate and gemcitabine for first-line treatment of advanced pancreatic cancer. Gastrointest. Cancer Res. **5**(3), 77–83 (2012)
52. Muller, H.-G., Wang, J.-L.: Hazard rate estimation under random censoring with varying kernels and bandwidths. Biometrics **50**(1), 61–76 (1994)
53. Narula, S.C., Li, F.S.: Sample size calculations in exponential life testing. Technometrics **17**(2), 229–231 (1975)
54. Piantadosi, S.: Clinical Trials: A Methodologic Perspective. Wiley, Hoboken (2013)
55. Preston, S.H., Heuveline, P., Guillot, M.: Demography: Measuring and Modeling Population Processes. Blackwell Malden, MA (2000)
56. Putter, H., Fiocco, M., Geskus, R.B.: Tutorial in biostatistics: competing risks and multi-state models. Stat. Med. **26**, 2389–2430 (2007)
57. Ross, E.A., Moore, D.: Modeling clustered, discrete, or grouped time survival data with covariates. Biometrics **55**(3), 813–819 (1999)
58. Rubinstein, L.V., Gail, M.H., Santner, T.J.: Planning the duration of a comparative clinical trial with loss to follow-up and a period of continued observation. J. Chronic Dis. **34**(9-10), 469–479 (1981)
59. Schemper, M., Smith, T.L.: A note on quantifying follow-up in studies of failure time. Control. Clin. Trials **17**(4), 343–346 (1996)
60. Schoenfeld, D.A.: The asymptotic properties of nonparametric tests for comparing survival distributions. Biometrika **68**, 316–319 (1981)
61. Schoenfeld, D.A.: Sample-size formula for the proportional-hazards regression model. Biometrics **39**(2), 499–503 (1983)
62. Shih, W.J., Aisner, J.: Statistical Design and Analysis of Clinical Trials: Principles and Methods. Chapman & Hall/CRC, (2016)
63. Steinberg, M.B., Greenhaus, S., Schmelzer, A.C., Bover, M.T., Foulds, J., Hoover, D.R., Carson, J.L.: Triple-combination pharmacotherapy for medically ill smokers: A randomized trial. Ann. Intern. Med. **150**(7), 447–454 (2009)

64. Struewing, J.P., Hartge, P., Wacholder, S., Baker, S.M., Berlin, M., McAdams, M., Timmerman, M.M., Brody, L.C., Tucker, M.A.: The risk of cancer associated with specific mutations of BRCA1 and BRCA2 among Ashkenazi Jews. N. Engl. J. Med. **336**(20), 1401–1408 (1997)
65. Tableman, M., Kim, J.S.: Survival Analysis Using S: Analysis of Time-to-Event Data. Chapman and Hall/CRC press, (2004)
66. Therneau, P.M., Grambsch, T.M., Shane Pankratz, V.: Penalized survival models and frailty. J. Comput. Graph. Stat. **12**(1), 156–175 (2003)
67. Therneau, T., Crowson, C.: Using time dependent covariates and time dependent coefficients in the cox model. Vignette for R survival package, http://cran.r-project.org/web/packages/survival, July 2015
68. Therneau, T.M., Grambsch, P.M.: Modeling Survival Data: Extending the Cox Model. Springer, New York (2000)
69. Therneau, T.M., Grambsch, P.M., Fleming, T.R.: Martingale-based residuals for survival models. Biometrika **77**(1), 147–160 (1990)
70. Therneau, T.M., Offord, J.: Expected survival based on hazard rates (update). Technical Report 63, Mayo Clinic Department of Health Science Research (1999)
71. Tibshirani, R.: Regression Shrinkage and Selection via the Lasso. J. R. Stat. Soc. Ser. B Methodol. **58**, 267–288 (1996)
72. Tibshirani, R.: The lasso method for variable selection in the Cox model. Stat. Med. **16**, 385–395 (1997)
73. Turnbull, B.W.: The empirical distribution function with arbitrarily grouped, censored and truncated data. J. R. Stat. Soc. Ser. B **38**, 290–295 (1976)
74. Wang, Y., Yu, Y.-y., Li, W., Feng, Y., Hou, J., Ji, Y., Sun, Y.-h., Shen, K.-t., Shen, Z.-b., Qin, X.-y., Liu, T.-s.: A phase II trial of xeloda and oxaliplatin (XELOX) neo-adjuvant chemotherapy followed by surgery for advanced gastric cancer patients with para-aortic lymph node metastasis. Cancer Chemother. Pharmacol. **73**(6), 1155–1161 (2014)
75. Ware, J.H., Demets, D.L.: Reanalysis of some baboon descent data. Biometrics 459–463 (1976)
76. Wei, L.J., Lin, D.Y., Weissfeld, L.: Regression analysis of multivariate incomplete failure time data by modeling marginal distributions. J. Am. Stat. Assoc. **84**(408), 1065–1073 (1989)
77. Weisberg, S.: Applied Linear Regression, 4th edn. Wiley, Hoboken (2014)

Index

D. Moore, *Applied Survival Analysis Using R*, Use R!,
DOI 10.1007/978-3-319-31245-3

R Package Index

D. Moore, *Applied Survival Analysis Using R*, Use R!,
DOI 10.1007/978-3-319-31245-3