Practical session 1:
Survival and event history analysis using R: Introduction

R packages

- Tools available in the package *survival*, *KMsurv* and *eha*
- Check if this package has been installed (type installed.packages() at the console and press enter)
- If the above-mentioned packages are not installed then type install.packages("KMsurv") at the console to install *KMsurv* provided you have internet connection. If you have a zip file of the package in your PC then go to Packages Menu and select Install Package(s) from local zip files... and then select the zip file.
- To use the functions available in the package *survival*, load the package by typing *library(survival)*
- Functions which we will be using during the course:
  - *Surv* - create a survival object
  - *lifetab* - Life-table estimates (available in the package *KMsurv*)
  - *survfit* - Kaplan-Meier estimates
  - *survdiff* - log-rank test
  - *weibreg* - parametric survival regression model available in *eha*
  - *coxph* - Cox proportional hazards model
  - *rexp, rweibull, rgamma* - generate random sample from exponential, Weibull and gamma distribution

Try the following

1. Read in the data set veteran. The data set can be found from the R survival package:

```
library(survival)
?veteran # for explanations for the variables in the data set
data(veteran) # load the data
str(veteran) # show records of the data
```

(a) Plot a histogram of the survival times corresponding to uncensored observations (veteran$status == 1). (See help(hist))
Note: Try

```
hist(veteran$time[veteran$status == 1], nclass=30, main='', xlab='Survival time (days)',
     col='gray90')
```

by changing values of nclass and also without nclass.

(b) Create an output file where the histogram is stored.

```
postscript(file.path(outpath, 'survtimes.eps'), width=6, height=6, paper='special',
           horizontal=FALSE)
op <- par(mar=c(4,4,0,0), mgp=c(2,1,0))
hist(veteran$time[veteran$status == 1], nclass=30, main='', xlab='Survival time (days)',
     col='gray90')
par(op)
dev.off()
```

(c) See the ordering of the levels of cell type using

`levels(veteran$celltype).`

(d) Create a R object celltypeo in the veteran data frame by reordering the levels of celltype so that the category "large" is the reference category. You can do this by using *relevel()* function. Hint:

`veteran$celltypeo<-relevel(veteran$celltype, ref="large")`

Check the levels of the newly created variable.

(e) Use the *survfit* routine in R to Calculate the Kaplan-Meier estimate of overall survival in the data. In the survival routine of R, the response variable needs to be specified as a *survival object*. If the observed faliure time variable is *time* and the failure indicator is *status*, the response variable is created as

`S(time, status).`

Applying the plot command to the output object from the survfit routine, you can draw the estimate and its confidence limits. Experiment with different connfidence levels (e.g. 95%, and 80%).

You can also practice with the plot command options (e.g. xlab, ylab).

(f) Plot the Kaplan-Meier estimates of the survival functions separately for the two treatment groups (standard vs. test). Does there appear to be a difference between the two groups in survival? Irrespective of the treatment group, comapre the survival in groups defined by the histological type of tumor (variable celltype). You can also explore the effect of survival of the other covariates in the data.

2. The data set oldmort in eha contains survival data from the parish Sundsvall in the mid-east of 19th century Sweden. Read in this data.

```
library(eha)
data(oldmort)
?oldmort
```

Every person who was present and alive and 60 years of age or above anytime between 1 January 1860 and 31 December 1879 was followed from the entrance age (for most people that would be 60) until the age when last seen, determined by death, out-migration, or surviving until 31 December 1879. Those born during the eighteenth century would enter observation at an age above 60, given that they lived long enough, that is, at least until 1 January 1860.

(a) Look at the first few lines of the data set (*h*ead(oldmort)), and also the types of the variables stored (*s*tr(oldmort)).

(b) Explore the data by tabulating enter and exit (use integer function to use only completed years), birth year, event etc.

(c) Special interest: (enter, exit, event)

Notes:

The first individual was born around 1 July 1765, and so almost 95 years of age when the study started. Suppose that this woman had died at age 94; then she had not been in our study at all. This property of our sampling procedure is a special case of a phenomenon called length-biased sampling. That is, of those born in the eighteeenth century, only those who live well beyond 60 will

be included. This bias must be compensated for in the analysis, and it is accomplished by conditioning on the fact that these persons were alive at 1 January 1860. This technique is called left truncation.

More specifically, the sampling frame is all persons observed to be alive and above 60 years of age between 1 January 1860 and 31 December 1879. The start event for these individuals is their 60th anniversary, and the stop event is death. Clearly, many individuals in the data set did not die before 1 January 1880, so for them we do not know the full duration between the start and stop events; such individuals are said to be right censored.

Those who are above 60 at this start date are included only if they did not die between the age of 60 and the age at 1 January 1860. If this is not taken into account, a bias in the estimation of mortality will result. The proper way of dealing with this problem is to use left truncation, which is indicated by the variable enter. The statistical implication (description) of left truncation is that its presence forces the analysis to be conditional on survival up to the age enter.

(d) Research questions are: (i) Do women live longer than men? (Yes), (ii) Is it advantageous for a long life to be married? (Yes), (iii) Does socioeconomic status play any role for a long life? (Dont know), and (iv) Does place of birth have any impact on a long life, and if so, is it different for women and men?

(e) Plot the cumulative hazard functions separately for men and women and interprete it.

```
with(oldmort, plot(Surv(enter, exit, event), strata = sex))
```

3. Generate 100 random numbers from exponential distribution with mean 0.01 and store it in T. Before you do this, look at the help(rexp) and the parameter which it accepts.

(a) Plot the empirical distribution function.

(b) Esimate the rate from the simulated data and overlay th plot of the distribution function $1 - \exp(-obsrate * t)$.

(c) Overlay the plot of the true exponential distribution function.

4. Explore the possibilities for different kinds of line and point plots. Vary the plot symbol, line type, line width, and colour. Also, try to give legend in the above graph.

5. Use sapply to simulate the result of taking the mean of 10 random numbers from the standard normal distribution for 100 independent samples. Try replicate() function for the same.

Getting started with R

The latest version of R is available from `http://www.r-project.org/`. Download "R-X.X.X-win32.exe" and double-click this file to start installation, and follow the instructions. Packages that are not part of the base distribution must be downloaded and installed separately.

Example: R code for the normal model

Key in the following at the console and press enter after each statement.

```
y <- rnorm(10000, 0, 1)

# sequence of FALSE and TRUE depending on y < 1.64 or y >= 1.64:
exceed <- (y >= 1.64)
exceed.prob <- mean(exceed)

# this should be exactly equal to exceed.prob:
check  <- mean((y >= 1.64))

exceed.prob

check

# computes 1 - P(Y =< 1.64) = P(Y >= 1.64):
1 - pnorm(1.64, 0, 1)

# computes P(Y > 1.64) = P(Y >= 1.64):
pnorm(1.64, 0, 1, lower.tail = FALSE)
```

Is `exceed.prob` comparable to the exact probability?

A tour of R

Getting help

- For official and contributed documentation, visit `http://cran.r-project.org` and find the nearest CRAN (Comprehensive R Archive Network) mirror site. Click R FAQ to get help.

- Get help using the Help on the menu.
- Type `?` "word" at the console and help about the "word" will be displayed.
- There is no official support for R. The most effective way to obtain help is to subscribe to the "R-help" mailing list, which gives access to an informal support network. This is available from `http://www.R-project.org/ ->` Mailing lists.
- Email archives can be searched for questions that may have been previously answered. You can also post your query if you do not find it already listed in the archive.

Using R as a calculator and basic commands

Try the following commands by typing them in at the console and observe the results.

```
# : comment

# assignment statement;
# x is assigned value 10 (you can also use x = 10):
x <- 10

# show the value of x:
x

# show the value of X:
X
```

Note the error message `Error: object "X" not found`. This is because R is case-sensitive and differentiates between uppercase and lowercase letters.

```
# a logical condition which returns TRUE
# if x is not equal to 3 and FALSE otherwise:
(x != 3)

# x is assigned a logical value:
x <- (x != 3)
```

Note that R does not give any warning when overwriting an object by another object with the same name. Up arrow brings up the previous commands.

```
# list names of the objects in use:
ls()
objects()

# remove the object x.
# Try ls() after using rm(x) and see the list.
rm(x)

# remove all objects from the workspace:
rm(list=ls())

# show the installed packages in R:
installed.packages()

# quit from an R session:
q()

# Use getwd() to print the working directory
getwd()

# Use setwd() to set the working directory
setwd("c:/sangita")

# To see what filles you have in the working directory
dir()
```

When exiting from R you are aksed about saving all the objects in your current work space. This is not recommended as the work space can fill up with temporary objects, and it is easy to forget what these are when you resume the session. It is better to build up a script file as you work, and to run this at the start of a new session and also to save the results which you will require for later use.
To save the output from an R command in a file the sink() command is used. For example,

```
> sink("c:/sangita/results/parametricreg.txt")
```

```
> table(sex)
> table(sex,age)
> sink()
```

While a sink is open all output will go to it. Opening a file with sink() will overwrite its contents - to append output to a file, use the append=TRUE option with sink(). To close a sink, use sink() without arguments.
You can save any R object to disc using save command

```
> sexdist <- table(sex)
> sexagedist <- table(sex,age)
> save(sexdist, sexagedist, file = "c:/sangita/results/Rtrial")
> save(sexdist, sexagedist, file = "c:/sangita/results/Rtrial.txt", ascii=T)
> load(filename)
```

You can read data in R using read.table(). Type ?read.table in R.
Use cut function in R to create grouping variable. For example, to create age-groups 25-29, 30-34, ..., 70-74, use

```
> age<-seq(25,74,1)
> agegrp <- cut(age, breaks = seq(25, 75, 5), right = FALSE)
> agegroup <- cut(age, breaks = 10, right = FALSE)
```