ReadMe file: A user’s guide for the vitality model parameter fitting routine and the S-Plus functions contained in file VitalityModelFitting.ssc.

The VitalityModelFitting.ssc file contains all functions necessary to run the MLE parameter fitting routine for the vitality-based survival model. The functions are written in the S-Plus programming language and are to be run in the S-Plus programming environment. All functions were written by David H. Salinger of Columbia Basin Research at The University of Washington.

To use the routine, simply open the VitalityModelFitting.ssc file in your S-Plus environment and compile (F10 key, on the PC).

An important set-up tip: If subjection of the population to a stressor or toxin occurs for only a time interval and is then removed (such as an interval of starvation), the survival curve is defined only for the period after the stressor has been applied. The time component of the study should then begin after the imposition of the environmental stressor or toxin has been completed.

This ReadMe file contains:

1. The header for the main function vitality.fit. The header explains all user options and provides some example function calls.
2. A list of subordinate functions with brief descriptions.

For more information on the fitting routine, see Salinger et. al (to be published). For more information on the vitality based survival model, see Anderson (1992, 2000).

1. Header for vitality.fit:

```
"vitality.fit"<-
function(time,sdata,rc.data=F,se=F,gfit=F,datatype="CUM",ttol=.000001,
                  init.param=F,s=F,pplot=T,tlab="days",lplot=T,cplot=F,silent=F)
#
#     Vitality based survival model: parameter fitting routine:
#
# REQUIRED PARAMETERS:
#     time - time component of data: time from experiment start.  Time should
#          start after the imposition of a stressor is completed.
#     sdata - survival or mortality data.  The default expects cumulative
#          survival fraction.  If providing incremental mortality fraction
#          instead, use option: datatype="INC".
#          The default also expects the data to represent full mortality.
#          Otherwise, use option: rc.data=T to indicate right censored data.
#
# OPTIONAL PARAMETERS:
#     rc.data =T  - specifies Right Censored data.   If the data does not
#          represent full mortality, it is probably right censored.  The default
#          is rc.data=F.  A third option is rc.data="TF".  Use this case to add
#          a near-term zero survival point to data which displays nearly full
#          mortality ( <.01 survival at end).  If rc.data=F but the data does
#          not show full mortality, rc.data="TF" will be
#          invoked automatically.
#     se =<population>  calculates the standard errors for the MLE parameters.
#          Default is se=F.  The initial study population is necessary for
#          computing these standard errors.
#     gfit =<population>  provides a Pearson C type test for goodness of fit.
#          Default is gfit=F. The initial study population is necessary for
#          computing goodness of fit.
#     datatype ="CUM" -cumulative survival fraction data- is the default.
#          Other option: datatype="INC" - for incremental mortality fraction
#          data.  ttol (stopping criteria tolerence.)  Default is .000001 .
#          specify as ttol=.0001.
#          If one of the liklihood plots (esp. for "k") does not look optimal,
#          try decreasing ttol.   If the program crashes, try increasing ttol.
#     init.params =F  has the routine choose initial parameter estimates for
#          r,s,k  (default: =F). If you wish to specify initial parameter values
#          rather than have the routine choose them, specify
#          init.params=c(r,s,k) in that order (eg. init.params=c(.1,.02,.003)).
#     pplot =T provides plots of cumulative survival and incremental mortality -
#          for both data and fitted curves (default: =T).  pplot=F provides no
#          plotting.  A third option:  pplot=n  (n>=1) extends the time axis of
#          the fitting plots  (beyond the max time in data).  For example:
#          pplot=1.2 extends the time axis by 20%.  (Note:  the incremental
#          mortality plot is a continuous representation of the appropriately-
#          binned histogram of incremental mortalities.)
#     tlab ="<time units>" specifies units for x-axis of plots.  Default is
#          tlab="days".
#     lplot =T provides likelihood function plotting (default =T).
#          Note:  these plots are not "likelihood profiles" in that while one
#          parameter is varied, the others are held fixed, rather than
#          re-optimized. (must also have pplot=T.)
#     cplot =T provides a likelihood contour plot for a range of r and s values
#          (can be slow so default is F).   Must also have lplot=T (and pplot=T)
#          to get contour plots.
#     silent =T stops all print and plot options (still get most warning and all
#          error messages) Default is F.  A third option, silent="verbose" also
#          enables the trace setting in the ms (minimum sum) S-Plus routine.
#
# RETURN:
#     vector of final MLE r,s,k parameter estimates.
#     standard errors of MLE parameter estimates (if se=<population> is
#     specified).
```

Examples:

1. Basic call, with times and cumulative survival fractions in (tdata,sdata):

   ```
   vitality.fit(tdata,sdata)
   ```

2. If data is right censored (<100% mortality at completion of study) and if standard errors and Pearson’s goodness of fit test are desired (given that study population is 113):

   ```
   vitality.fit(tdata,sdata,rc.data=T,se=113,gfit=113)
   ```

3. If time units are weeks, and the data covers 10 weeks but you desire plots showing 15 weeks:

   ```
   vitality.fit(tdata,sdata,tlab=”weeks”,pplot=1.5)
   ```

4. If sdata is incremental mortality (rather than cumulative survival) and no plots or output is desired (the final parameters will still be returned):

   ```
   vitality.fit(tdata,sdata,datatype=”INC”,silent=T)
   ```

2. **List of functions** (in the order they appear in VitalityModelFitting.ssc):

- vitality.fit() – the main program described above.
- dataPrep() – removes NAs from data; creates incremental mortality data from cumulative survival data (or vise – versa); deals with right censored data.
- rsk.init() – computes initial *r,s,k* estimates as starting values for MLE optimization.
- r.s.slope – (not a function) a short script to create the data frame of normalized *r,s,slope* values for use by the rsk.init function.
- SurvFn() – the vitality based survival model.
- SurvProbInc() – computes incremental survival probabilities.
- logLikelihood() – returns a vector of terms which sum to the negative log likelihood.
- indexFinder() – an auxiliary function which returns the index of the first value of vector x that is less than or equal to a given value.
- stdErr() – computes the standard error of the MLE estimates.
- C1.calc() – provides the Pearson’s C1 goodness of fit test.
- plotting() – provides the cumulative survival and incremental mortality plots.
- profilePlot() – provides the likelihood plots (one each when varying *r,s,k* about their MLE) and a contour likelihood plot (created by varying *r* and *s).*