Chapter 6

Parametric survival analysis

Until now, most of the course has focused on non- or semi-parametric methods, namely the Kaplan–Meier estimate of $S(t)$ and the Cox model its extensions. The Cox model is semi-parametric because the form of the baseline hazard is not specified, only the form of the effect of covariates. Because of this, it is particularly popular in survival analysis.

In parametric survival analaysis, on the other hand, all parts of the model are specified, both the hazard function and the effect of any covariates. The strength of this is that estimation is easier and estimated survival curves are smoother as they draw information from the whole data. In addition, it is possible to do more sophisticated analyses with parametric models, such as including random effects or using Bayesian methodology to pool sources of information. The main drawback to parametric methods is that they require extra assumptions that may not be appropriate.

In this chapter, we cover some of the possibilities in parametric modelling of survival. We begin by recapping some of chapter 1. Then we consider accelerated failure time models, that allow covariates to predict survival. We conclude by introducing frailty, or random effects, models, that allow additional sources of variability to be accounted for.

6.1 Preliminaries

Consider the case in which there are no covariates or censoring. Then if we assume a particular model M (say, Weibull) with parameter(s) $\boldsymbol{\theta}$, the likelihood contribution from an individual failing at time t_i is $f(t_i|\boldsymbol{\theta}, M)$. Since failures are independent (most of the time), the likelihood from m observations is

$$f(\mathbf{t}|\boldsymbol{\theta}, M) = \prod_{i=1}^{m} f(t_i|\boldsymbol{\theta}, M)$$

The maximum likelihood estimate of the parameters is typically obtained by taking the logarithm, and ideally by then taking the (partial) derivatives with respect to $\boldsymbol{\theta}$. Setting *all* partial derivatives equal to zero and solving the resulting system of equations for $\boldsymbol{\theta}$ will yield the MLE $\hat{\boldsymbol{\theta}}$. An approximate confidence interval can be obtained by appealing to the asymptotic normality of the MLE, using the observed information

$$\hat{\mathbf{V}}(\hat{\boldsymbol{\theta}}) = I^{-1}(\hat{\boldsymbol{\theta}}) = \left(-\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log f(\mathbf{t}|\boldsymbol{\theta}, M)\right)^{-1}.$$

Often, no analytic solution for the MLE exists. In such cases, a numerical approach must be taken. A popular method is to use Newton–Raphson (see chapter 3), but this is very sensitive to initial conditions and requires differentiation. There are several other methods, and two are outlined below.

6.1.1 Cross Entropy

One alternative (of many) is called Cross Entropy, which has the advantages of simplicity and generalisability. For information on the method, see `www.cemethod.org`. It may require many function calls, though, which on occasion may make it prohibitively expensive.

Consider the Weibull example. For illustration, it will be nice to have a contour plot of the likelihood. This can be obtained by making a grid of points, calculating the (log) likelihood, and creating a plot; this is very easy in R using the following code, where `t` is a vector of data input elsewhere.

Firstly, the following code defines a function to calculate the log-likelihood:

```
logl=function(kappa,lambda)
{
  logf=rep(0,length(kappa))
  ok=as.logical((lambda>0)*(kappa>0))
  for(i in 1:length(kappa))
  {
    if(ok[i])logf[i] = m*log(lambda[i])+m*log(kappa[i])
       +(kappa[i]-1)*sum(log(t)) -lambda[i]*sum(t^kappa[i])
    if(!ok[i])logf[i]=-999999999 #ie very small
  }
  logf
}
```

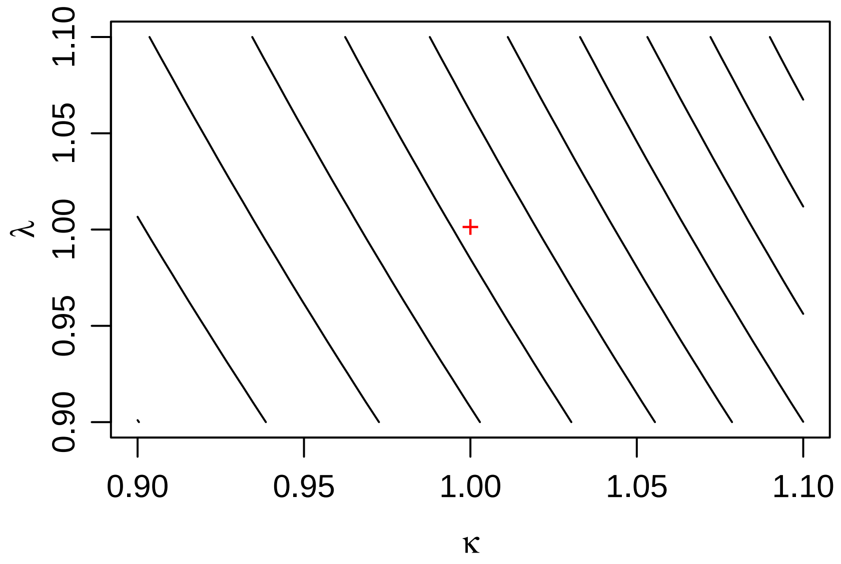
```
grid=20
mink=0.45
maxk=1.0
minl=0.1
maxl=0.6
Kappa=seq(mink,maxk,length.out=grid)
Lambda=seq(minl,maxl,length.out=grid)
Logf = matrix(0,grid,grid)
for(i in 1:grid)
{
  for(j in 1:grid)
  {
    k=Kappa[i];l=Lambda[j];Logf[i,j]=logl(k,l)
  }
}
Logf=Logf-max(Logf)
contour(kappa,lambda,exp(logf),drawlabels=FALSE,
  xlab=expression(kappa),xlim=c(0,1.5),
  ylab=expression(lambda),ylim=c(0,1.5))
```

The resulting contour plot will be in the background for our other plots illustrating the CE method. Note that the contour plot is only really do-able for toy problems with two parameters: it may guide the eye towards the MLE and with and ad hoc sequence of iterations of zooming in on where we think the maximum is, can provide the MLE to the desired accuracy. Grid searches (as effectively the contour plot is) are horrendously inefficient and infeasible for more than two or three parameters, a limitation not shared by Cross Entropy.

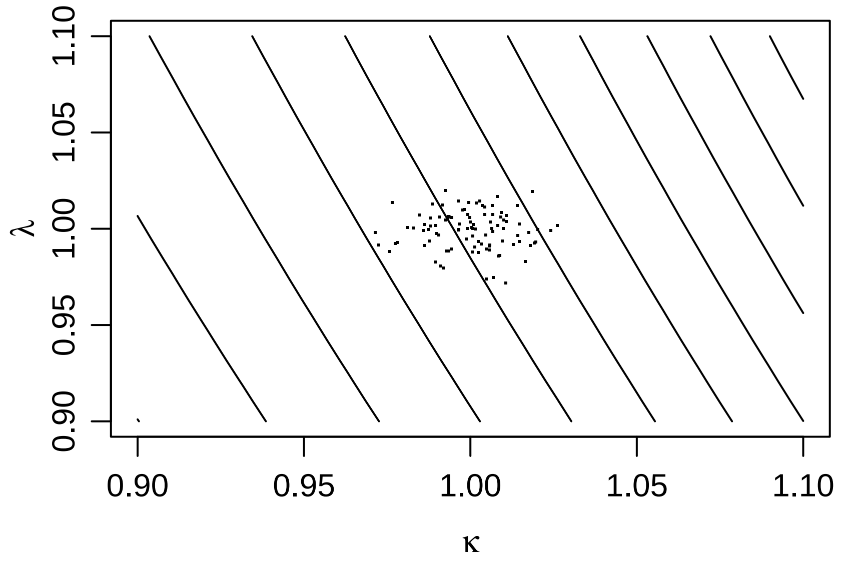
In Cross Entropy, we create a random sample of candidate points for the MLE. At each point, we evaluate the log likelihood. We discard the worst $X\%$ of these, and take the mean of those retained: this then defines the mean for the next iteration of random sampling. When the mean changes by less than some threshold, the routine stops.

For the Weibull problem, we specify the following arguments:

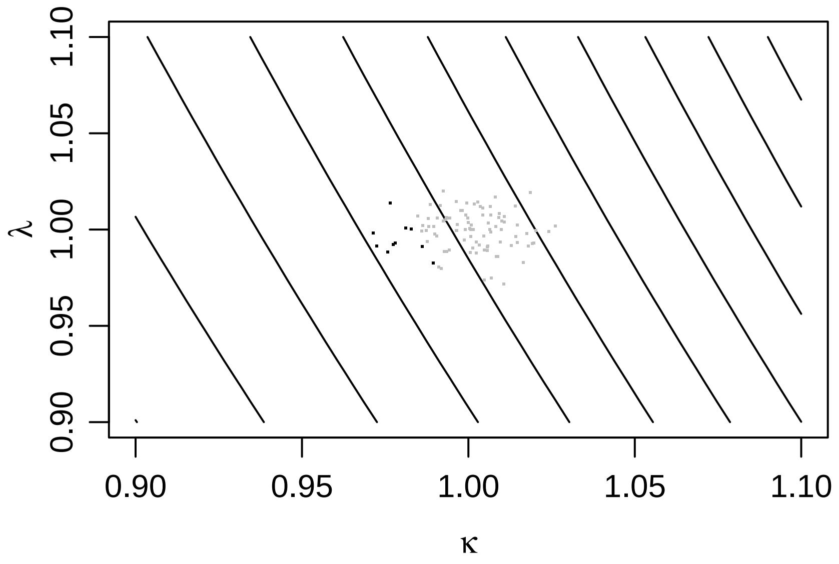
- $\hat{\kappa}^0$, $\hat{\lambda}^0$ (initial guesses at the MLE)
- $\sigma_\kappa = \sigma_\lambda = 0.01$ (tuning arguments)
- $n_{\text{part}} = 100$ (the number of particles in the sample)
- $n_{\text{top}} = 10$ (the number of particles in the retained set)
- $\epsilon = 0.0001$ (threshold for ending the routine)



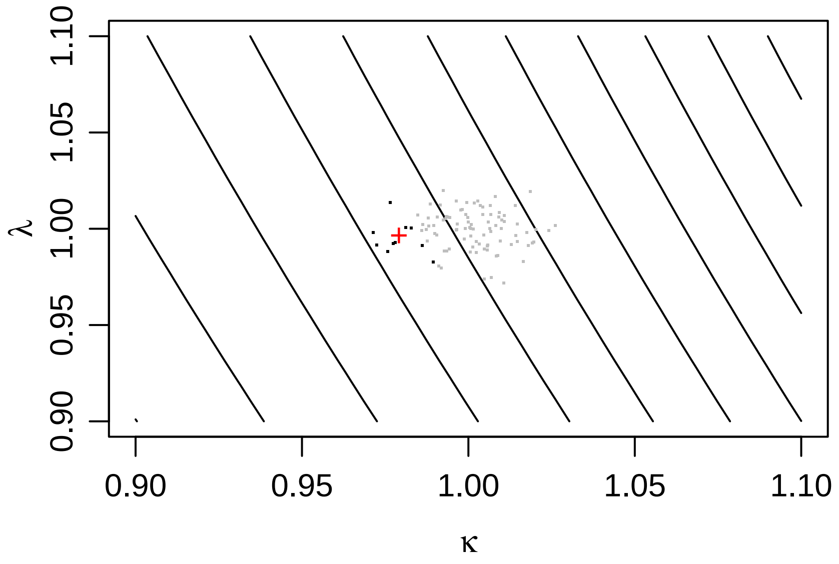
Step 1: cross marks starting guess for MLE



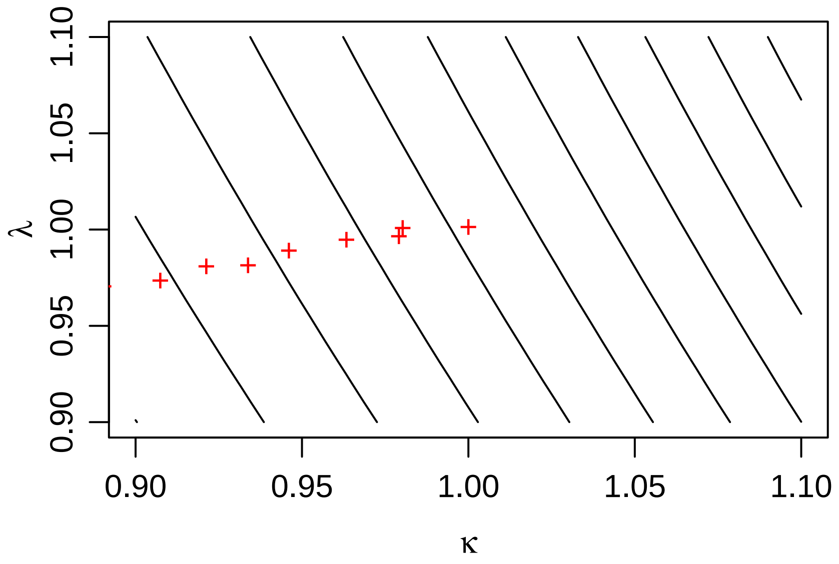
Step 2: points mark sample



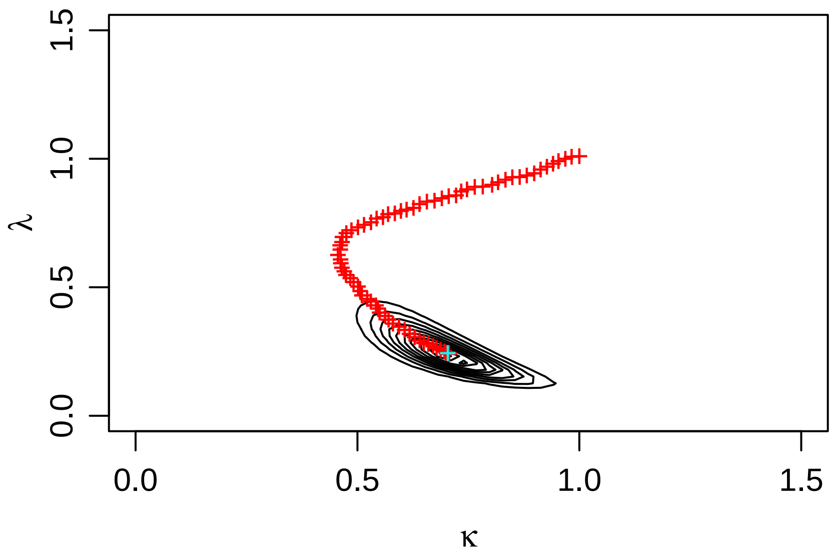
Step 3: grey points have been discarded



Step 4: cross marks next guess at MLE



Step 5: crosses mark guesses at MLE



Step 5: red crosses mark guesses at MLE; cyan cross marks final estimate

The code to run this algorithm was:

```
#t are the data
npart=100
muk=1
mul=1
sigmak=0.01
sigmal=0.01
finished=FALSE
topn=10
bestk=muk
bestl=mul
bestlogf=logl(bestk,bestl)
while(!finished)
{
  old_muk=muk;old_mul=mul
  kappa=rnorm(npart,old_muk,sigmak)
  lambda=rnorm(npart,old_mul,sigmal)
  logf=logl(kappa,lambda)
  toplot=order(logf,decreasing=TRUE)[1:topn]

  muk=mean(kappa[toplot])
  mul=mean(lambda[toplot])
  if(max(logf)>bestlogf)
  {
    bestlogf=max(logf)
    bestk=kappa[which.max(logf)]
    bestl=lambda[which.max(logf)]
  }

  if((abs(muk-old_muk)<0.0001) && (abs(mul-old_mul)<0.0001))finished=TRUE
  kappahat=muk
  lambdahat=mul
  points(kappahat,lambdahat,col=5,pch='+')
}
kappahat=bestk
lambdahat=bestl
points(kappahat,lambdahat,col=2,pch='+')
```

6.1.2 Simulated annealing

Simulated annealing was introduced by Kirkpatrick et al. (1983) as a stochastic optimisation method. Unlike Cross Entropy, it is not really suitable for stochastic response surfaces, but for deterministic objective functions is probably more efficient. It is *very* similar to Markov chain Monte Carlo (MCMC) simulation, so if you have learned MCMC you may readily apply the same technique with minor adaptation to optimisation problems.

Let us begin with a recap of MCMC. MCMC is a simulation based approach to sample a complicated distribution, of a (usually multivariate) random variable θ. You need to be able to evaluate the density function $f(\theta)$ of the distribution *up to a constant of proportionality*, or the log density plus some unknown constant, $l(\theta)$. You specify initial conditions θ_0 for the random variable whose distribution you wish to sample and a proposal distribution, $q(\theta^\star|\theta_i)$, often a function of the current value θ_i of the random variable, that you can simulate from and evaluate the density of. You then successively propose new values of the random variable and either accept them or retain the old value according to the Metropolis–Hastings algorithm. This leads to a correlated sample of θ with density equal to the target density. Here is the algorithm:

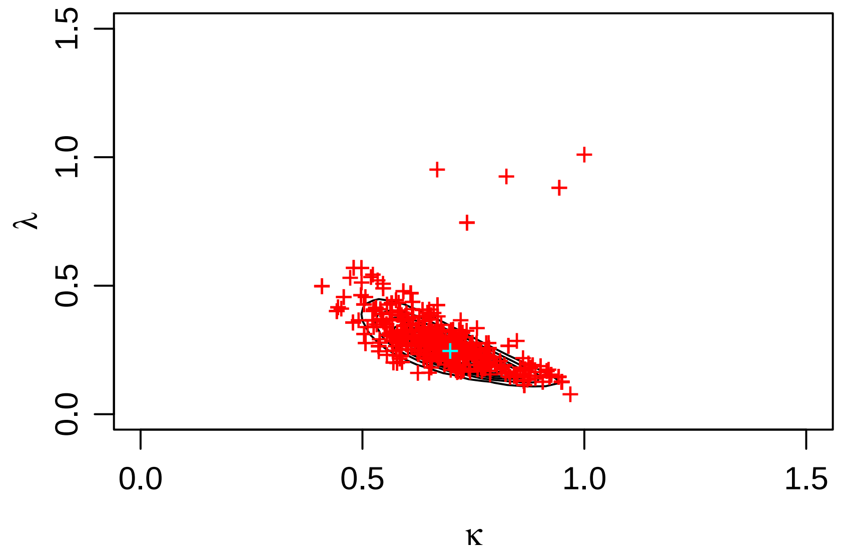
1. Set $i = 0$ to be the iteration.
2. Set θ_0 to be an arbitrary initial choice of θ.
3. Repeat the following steps until $i = M$, some large number (e.g. 10 000).
   - (3a) Draw $\theta^\star$ from $q(\theta^\star|\theta_i)$. A common choice is a normal distribution with mean θ_i and an arbitrary covariance.
   - (3b) Evaluate $l(\theta^\star)$.
   - (3c) With probability $\exp(l(\theta^\star) - l(\theta_i))q(\theta_i|\theta^\star)/q(\theta^\star|\theta_i)$ let $\theta_{i+1} = \theta^\star$, otherwise let $\theta_{i+1} = \theta_i$.
   - (3d) Increase i by one.

MCMC is very popular in Bayesian statistics, for it provides a way to sample posterior distributions of parameters. It is not often used in frequentist statistics, but is actually quite useful there too. Although the likelihood function is not a probability density for the parameters, as long as it has finite integral you can treat it as such, and use MCMC to draw a sample proportional to it. The MLE can be taken to be the mode of the sample, which can be evaluated using kernel density estimation, or the sampled value with highest log-likelihood.

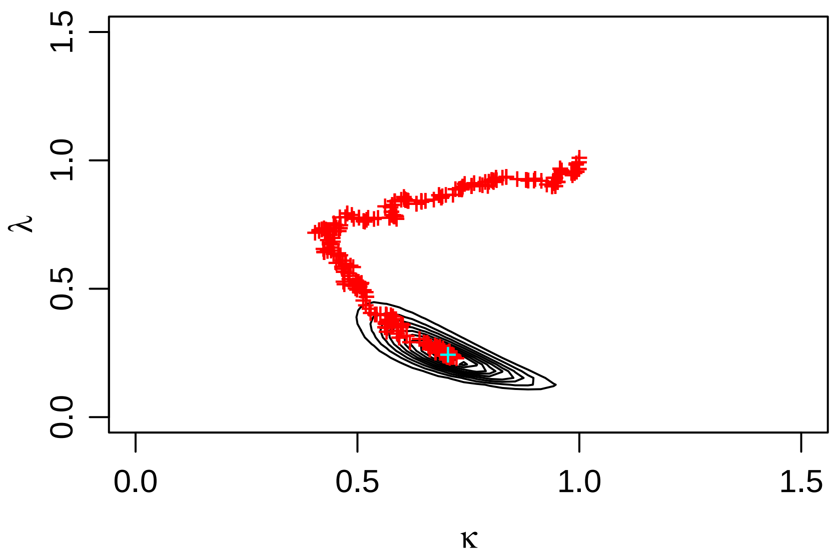
Simulated annealing replaces step 3c by the following

(3c) If $l(\theta^\star) > l(\theta_i)$ then $\theta_{i+1} = \theta^\star$. Otherwise, with probability $\exp[(l(\theta^\star) - l(\theta_i))q(\theta_i|\theta^\star)/(T_i q(\theta^\star|\theta_i))]$ let $\theta_{i+1} = \theta^\star$, otherwise let $\theta_{i+1} = \theta_i$. Here, T_i is the annealing schedule, and a simple choice is $T_0 = 1$, $T_{i+1} = 0.99T_i$.

In other words, simulated annealing is the same as MCMC except that it becomes progressively harder to move downhill as the routine continues. This encourages the resulting chain to stay nearer the maximum of the function. Note though that this is an heuristic optimisation method in that it does not guarantee an optimum be found.



MCMC search for MLE



SA search for MLE

The code to do the MCMC sample is as follows:

```
contour(Kappa,Lambda,exp(Logf),drawlabels=FALSE,
  xlab=expression(kappa),xlim=c(0,1.5),
  ylab=expression(lambda),ylim=c(0,1.5))
NITS=1000
kappa=1
lambda=1
logf=logl(kappa,lambda)
sigmak=0.1
sigmal=0.1
bestk=muk
bestl=mul
bestlogf=logf
for(i in 1:NITS)
{
  old_kappa=kappa;old_lambda=lambda;old_logf=logf
  kappa=rnorm(1,kappa,sigmak)
  lambda=rnorm(1,lambda,sigmal)
  logf=logl(kappa,lambda)
  accept=TRUE;if(kappa<0)accept=FALSE;if(lambda<0)accept=FALSE
  if(log(runif(1))>(logf-old_logf))accept=FALSE
  if(!accept){kappa=old_kappa;lambda=old_lambda;logf=old_logf}
  if(logf>bestlogf)
  {
    bestlogf=logf
    bestk=kappa
    bestl=lambda
  }
  points(kappa,lambda,col=2,pch='+')
}
kappahat=bestk
lambdahat=bestl
points(kappahat,lambdahat,col=5,pch='+')
```

The simulated annealing routine is similar:

```
contour(Kappa,Lambda,exp(Logf),drawlabels=FALSE,
  xlab=expression(kappa),xlim=c(0,1.5),
  ylab=expression(lambda),ylim=c(0,1.5))
NITS=1000
kappa=1
lambda=1
logf=logl(kappa,lambda)
sigmak=0.01
sigmal=0.01
bestk=muk
bestl=mul
bestlogf=logf
temperature=1
for(i in 1:NITS)
{
  old_kappa=kappa;old_lambda=lambda;old_logf=logf
  kappa=rnorm(1,kappa,sigmak)
  lambda=rnorm(1,lambda,sigmal)
  logf=logl(kappa,lambda)
  accept=TRUE;if(kappa<0)accept=FALSE;if(lambda<0)accept=FALSE
  if(logf<old_logf)
  {
    if(log(runif(1))>((logf-old_logf)/temperature))accept=FALSE
  }
  if(!accept){kappa=old_kappa;lambda=old_lambda;logf=old_logf}
  if(logf>bestlogf)
  {
    bestlogf=logf
    bestk=kappa
    bestl=lambda
  }
  points(kappa,lambda,col=2,pch='+')
  temperature=temperature*.99
}
kappahat=bestk
lambdahat=bestl
points(kappahat,lambdahat,col=5,pch='+')
```

6.1.3 Comparing parametric models

We can compare parametric models in the usual way. If they are nested, the likelihood ratio test can be used to test for a significant deviation from the more parsimonious model. If they are not, the Akaike information criterion (AIC) can be used to select the best fitting.

Note that it is fine to use the AIC to select between parametric models, or to select between semi-parametric Cox models, but not to select from a mixture of the two. This is because the data are different:

- parametric models are fit to the event times, whereas
- Cox models are fit to the ordered event times only (as the baseline hazard function is not estimated).

It would thus be an unfair comparison as the Cox model would be trying to explain a much simpler data set. Alternatives do exist but these are beyond the scope and remaining time-scale of the course.

6.1.4 Censored data

Censored data are dealt with trivially within parametric models. Limit attention to right-censored data for simplicity. We imagine there is a true failure time T. We observe a pair (t, δ).

- If $\delta = 1$, we know $T = t$, i.e. the individual failed at the time we have observed.
- If $\delta = 0$, we know $T > t$, i.e. the individual was right-censored at the time we have observed.

If we had observed the set of T_i for $i = 1, \ldots, m$, the likelihood would be...

6.2 Assessing the validity of common parametric models

The AIC or likelihood ratio test allow us to assess relative model goodness of fit, but not *absolute* model goodness of fit. Just because the log-logistic fits better than the exponential does not mean the log-logistic adequately describes the data, for example.

Thus, we would like a method, at least a graphical one, that lets us assess the absolute goodness of fit of a parametric model.

One way to do this is to create plots of a function of survival against a function of time that should be linear if the model is true. Obvious non-linearities would suggest the model does not characterise salient points in the data. It turns out that this is easy to do for some common parametric models, including the exponential, Weibull and log-logistic.

Our objective for each model is to obtain something linear in a function of time, where this function of time has no unobserved parameters, that is:

$$f_1\{S(t)\} = f_2(\boldsymbol{\theta}) + f_3(\boldsymbol{\theta})f_4(t).$$

This would then allow us to plot $f_1\{\hat{S}(t)\}$ using the Kaplan–Meier estimate versus $f_4(t)$ to obtain a straight line if the model were true.

6.3 Regression parametric models

So far in this chapter, we have ignored any covariates. But it is perfectly possible to introduce covariates into a parametric model—indeed, if we couldn't it would be fairly pointless to study them.

Consider first the exponential model with a single covariate x.

Without covariates, the hazard would be

$$h(t) = \lambda.$$

We can make this into a PHM by making λ a function of x:

$$h(t) = \exp(\beta_0 + \beta_1 x) = e^{\beta_0} e^{\beta_1 x} = h_0 e^{\beta_1 x}$$

where β_0 and β_1 (or, equivalently, h_0 and β_1) are parameters to be estimated.

But we can also make it into a different kind of model called an accelerated failure time model (AFT) by writing the hazard thus:

$$h(t) = 1/\exp(\alpha_0 + \alpha_1 x) = h_0 e^{-\alpha_1 x}$$

where now α_0 and α_1 are the parameters to be estimated.

For the exponential model, these are equivalent, but this is not true in general. It is possible for a model to be PHM but not AFT, or AFT but not PHM, or both, or neither.

6.3.1 Accelerated failure time models

In an AFT model we assume that the time t_i to $S(t_i) = \sigma$ for one individual i is a constant times the time t_j to $S(t_j) = \sigma$ for individual j. The same constant holds regardless of whether we consider $\sigma = 0.5$, i.e. the median, or any other quantile. Written mathematically, we have

$$S(t_i) = S(\psi(x_i, x_j)t_j)$$

where $\psi(x_i, x_j)$ is the acceleration factor for i relative to j.

If a parametric model has just one scale parameter λ, like the exponential does, that parameter is replaced by

$$\lambda = \exp(-\alpha_0 - \alpha_1 x_1 + \ldots).$$

If the model has both a scale λ and a shape parameter κ, as the Weibull or log-logistic do, the scale parameter only is replaced by this function of the covariates.

So for example, the exponential, Weibull and log-logistic AFT models with two covariates are:

- Exponential: $S(t, x_1, x_2) = \exp\{-t \exp(-\alpha_0 - \alpha_1 x_1 - \alpha_2 x_2)\}$
- Weibull: $S(t, x_1, x_2) = \exp\{-t^\kappa \exp(-\alpha_0 - \alpha_1 x_1 - \alpha_2 x_2)\}$
- Log-logistic: $S(t, x_1, x_2) = \{1 + t^\kappa \exp(-\alpha_0 - \alpha_1 x_1 - \alpha_2 x_2)\}^{-1}$

We can estimate the parameters as before, e.g. using maximum likelihood, and again their distribution will be asymptotically normal.

We can compare models with varying degrees of complexity in exactly the same way as with the Cox PHM: i.e. with the likelihood ratio test if the models are nested and with the AIC if they are not.

We can also assess the validity of the chosen parametric forms using the plots discussed in the last section, but stratified over covariates.

Model building techniques as described in chapter 4 can also be used for parametric models.

Note, however, that methods for dealing with violations of the PHA are not appropriate for AFT parametric models, as these already model the baseline hazard function. It *is* possible to stratify over covariates if the shape parameter differs with the covariates.

6.4 Parametric analyses in R

In this section, we illustrate how the methods mentioned in the last section can be implemented in R. In particular, we will illustrate the following things:

- how to fit a parametric model in R and interpret the output;
- how to plot the estimated survival function(s);
- how to plot the complementary log–log function(s) to evaluate model appropriateness;
- how to test if covariates influence survival in a parametric model; and
- how to select between competing parametric models.

We will illustrate these with an example we have already covered, albeit non-parametrically, namely the veteran's lung cancer study. This concerns the time until death among veterans diagnosed with lung cancer. The following predictors are available:

1. `treat`: 1 if the patient received a standard treatment and 2 if the test treatment
2. `cell`: this indicates if the cancerous cell type is *large* (1), *adeno* (2), *small* (3) or *squamous* (4). We create dummy categorical variables `c1`, `c2`, `c3`, `c4`, with `ci=1` if `cell=i` and 0 otherwise.
3. `perf`: "performance status" on the scale 0–100.
4. `age`: in years.
5. `prior`: 1 if prior therapy was given, 0 otherwise.

6.4.1 Fitting parametric models

There is a built-in function in the `survival` package that automatically fits a sepecified parametric model to survival data, called `survreg` (as in "survival regression", though semi-parametric methods are also instances of survival regression, so the name is not wholly appropriate). Its syntax is similar to that of `coxph`, so it will be fairly easy to work out how to invoke it.

This function allows four pre-defined parametric models to be fit:

- `weibull`
- `exponential` (a special case of the Weibull)
- `lognormal`
- `loglogistic`

We will concentrate on the Weibull function, mostly because I completely cannot work out the parametrisation of the log-normal and log-logistic functions based on their R help entries. (Sorry.) Also, it appears that the form for the fitted Weibull function differs slightly from that found in the notes. The R version can be seen in the next example.

Example 1

Suppose that we wish just to fit a Weibull model to the survival times without exploiting any of the covariates that we have observed. This can be done with the following command.

```
wei=survreg(Surv(t,delta)~1,dist="w")
```

The `1` means it is not regressing on any covariates. The `dist="w"` means it is using a Weibull distribution (only enough of the name of the distribution to specify it uniquely is needed, meaning you can use `dist="w"`, `dist="e"`, `dist="logn"`, `dist="logl"` instead of the full names given above).

Unfortunately, the output is not in the same format as that given from fitting a Cox PHM. The following output is obtained:

```
> wei
Call:
survreg(formula = S ~ 1, dist = "w")

Coefficients:
(Intercept)
  -1.107436

Scale= 1.173592

Loglik(model)= 7.2   Loglik(intercept only)= 7.2
n= 137
```

There are two parameters in this output to note. `Scale` is κ and `Intercept` is α_0 in the following equations:

$$\begin{aligned} S(t) &= \exp\{-\exp(-\alpha_0)^\kappa t^\kappa\} \\ h(t) &= \exp(-\alpha_0)^\kappa \kappa t^{\kappa-1}. \end{aligned}$$

Note these differ from the definition of the Weibull used by `rweibull()` and similar functions. So, the survival and hazard functions corresponding to these output are:

$$\begin{aligned} S(t) &= \exp\{-\exp(1.11)^{1.17} t^{1.17}\} \\ &= \exp\{-3.67 t^{1.17}\} \\ h(t) &= \exp(1.11)^{1.17} 1.17 t^{0.17} \\ &= 4.30 t^{0.17}. \end{aligned}$$

Thus we see that the hazard is around 3 and slowly increases over time.

Example 2

Now suppose that we wish to use cell type to predict survival. As cell type is categorical with more than two categories, we first create dummy binary variables. As there are four categories, we need three of these in the model, the fourth being identified with the `intercept` only.

We can fit the model as follows:

```
wei=survreg(S~c2+c3+c4,dist="w")
```

using `c1` as the baseline to compare the other categories against.

The output of this model is

```
> wei
Call:
survreg(formula = S ~ c2 + c3 + c4, dist = "w")

Coefficients:
(Intercept)          c2          c3          c4
 -0.4940319  -1.0831923  -1.2162022  -0.2627843

Scale= 1.030480

Loglik(model)= 21.1   Loglik(intercept only)= 7.2
        Chisq= 27.87 on 3 degrees of freedom, p= 3.9e-06
n= 137
```

The scale has the same interpretation as before (though note it is now closer to 1, i.e. the exponential model might be appropriate here). The intercept is applied to all categories. All categories except `c1` have an addition term associated with them. Thus, the survival and hazard functions for this model are:

$$\begin{aligned}
S(t, x = c_1) &= \exp\{-\exp(0.49)^{1.03} t^{1.03}\} \\
S(t, x = c_2) &= \exp\{-\exp(0.49 + 1.08)^{1.03} t^{1.03}\} \\
S(t, x = c_3) &= \exp\{-\exp(0.49 + 1.22)^{1.03} t^{1.03}\} \\
S(t, x = c_4) &= \exp\{-\exp(0.49 + 0.26)^{1.03} t^{1.03}\} \\
h(t, x = c_1) &= \exp(0.49)^{1.03} 1.03 t^{0.03} \\
h(t, x = c_2) &= \exp(0.49 + 1.08)^{1.03} 1.03 t^{0.03} \\
h(t, x = c_3) &= \exp(0.49 + 1.22)^{1.03} 1.03 t^{0.03} \\
h(t, x = c_4) &= \exp(0.49 + 0.26)^{1.03} 1.03 t^{0.03}.
\end{aligned}$$

The interpretation of this is that having an adeno cell (`c_2`) hastens death by a factor of $e^{1.08} = 2.9$ relative to a large cell (`c_1`), etc.

6.4.2 Plotting survival functions

The survival and hazard functions are easy to plot based on the output of the `survreg` function. We just create a vector of times spanning the time period of interest, and then for each combination of covariates of interest, create a vector of survival or hazard functions. These can then be plotted as a smooth line-plot.

It is informative to compare the resulting survival curves against those obtained from the Kaplan–Meier and Cox PHM fits. These will give a partial illustration of the suitability of the model, in the same was as expected versus observed plots did for the PHM.

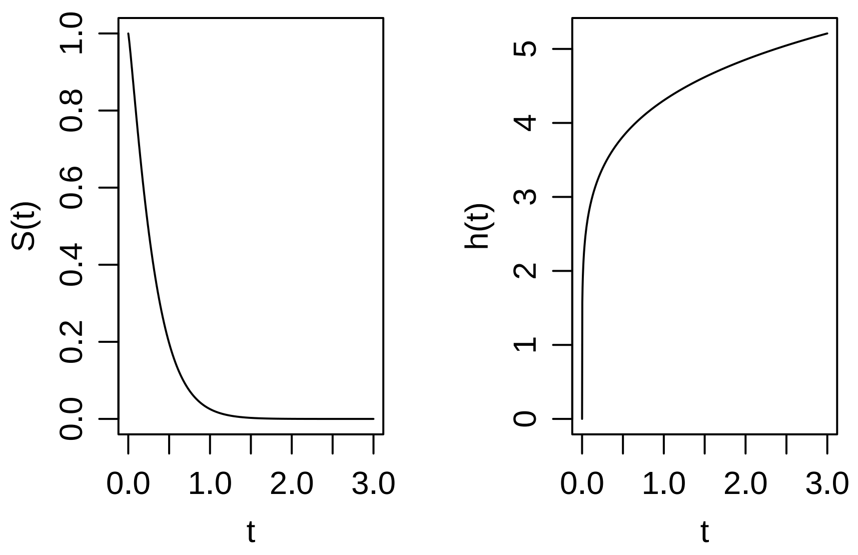
Example 1

Return to the no predictor scenario.

Event times span the range 0–3y. Thus we begin by creating a vector of times. To avoid confusion with the vector of event times, I have called this `zeit`, the German word for time. (Feel free to use any alternative name for this!)

```
wei=survreg(S~1,dist="w")

kappa=wei$scale
lambda=exp(-wei$coeff[1])^kappa

zeit=seq(from=0,to=3,length.out=1000)
s=exp(-lambda*zeit^kappa)
h=lambda*kappa*zeit^(kappa-1)
```

The following plots are the MLE estimates of the survival and hazard functions.



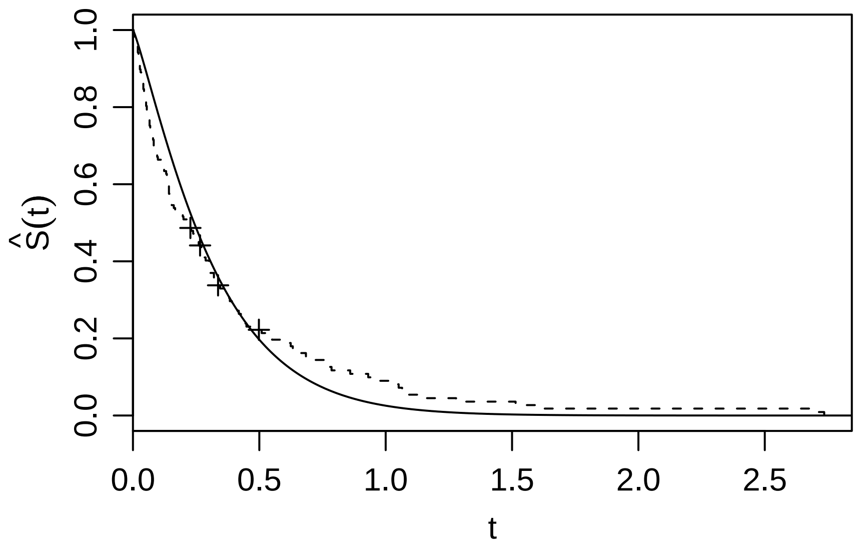
Note that these are much smoother than those based ultimately on the Kaplan–Meier estimate, as the Cox PHM is.

Let us compare the parametric survival curves to the Kaplan–Meier estimate (since there are no predictors, the Kaplan–Meier estimate is the same as the Cox PHM here).

Here are the commands:

```
plot(survfit(S~1),xlab='t',ylab=expression(hat(S)(t)),lty=2,conf.int=F)
lines(zeit,s)
```

The resulting graph is:



The Weibull model looks reasonable, though it doesn't quite match the overall shape of the non-parametric approach. Note though that this model contains no covariates, and the addition of the heterogeneity that they bring may resolve this.

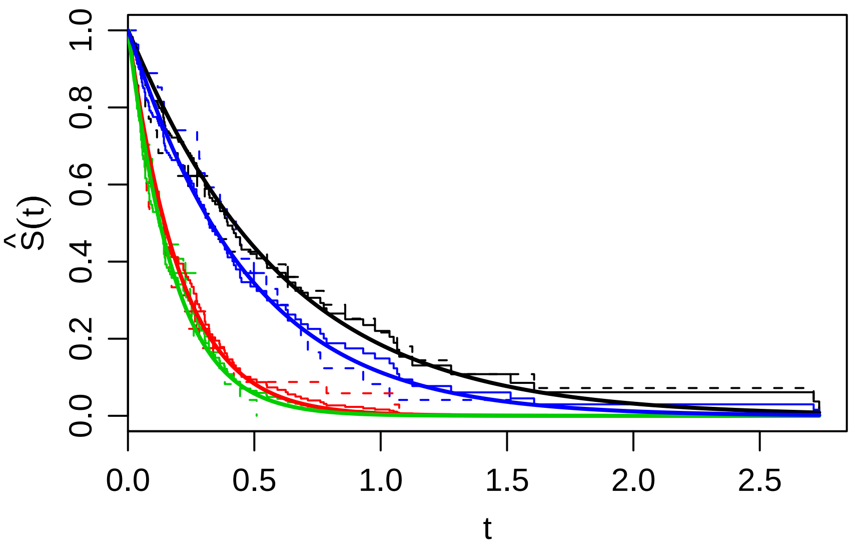
Example 2

Bearing that in mind, consider now the model with cell type as a predictor. Here are the commands to make a graph with the Kaplan–Meier estimates as dashed lines, the Cox PHM as solid lines, and the Weibull model as thick solid lines (`toki` is the Japanese word for time).

```
phm=coxph(S~c2+c3+c4)
dphm=coxph.detail(phm)
km=survfit(S~c2+c3+c4)
wei=survreg(S~c2+c3+c4,dist="w")

plot(km,col=c(1,4,3,2),xlab='t',ylab=expression(hat(S)(t)),lty=2)

toki=dphm$time
h0=dphm$hazard
S0=exp(-cumsum(h0))
beta=phm$coef

xmean=c(mean(c2),mean(c3),mean(c4))
x1=c(0,0,0)-xmean
x2=c(1,0,0)-xmean
x3=c(0,1,0)-xmean
x4=c(0,0,1)-xmean

S1=S0^(exp(beta%*%x1))
S2=S0^(exp(beta%*%x2))
S3=S0^(exp(beta%*%x3))
S4=S0^(exp(beta%*%x4))
lines(toki,S1,type='s',col=1)
lines(toki,S2,type='s',col=2)
lines(toki,S3,type='s',col=3)
lines(toki,S4,type='s',col=4)

kappa=wei$scale
lambda1=exp(-wei$coeff[1])^kappa
lambda2=exp(-wei$coeff[1]-wei$coeff[2])^kappa
lambda3=exp(-wei$coeff[1]-wei$coeff[3])^kappa
lambda4=exp(-wei$coeff[1]-wei$coeff[4])^kappa

zeit=0.001*(0:1000)*max(t/365.25)
s1=exp(-lambda1*zeit^kappa)
s2=exp(-lambda2*zeit^kappa)
s3=exp(-lambda3*zeit^kappa)
s4=exp(-lambda4*zeit^kappa)
```

```
lines(zeit,s1,col=1,lwd=2)
lines(zeit,s2,col=2,lwd=2)
lines(zeit,s3,col=3,lwd=2)
lines(zeit,s4,col=4,lwd=2)
```

The resulting graph is:



The parametric curves go nicely through the Kaplan–Meier estimates. This gives weight to the decision to use a Weibull model. Note also that the red and green curves (small and squamous cells) are very close, suggesting that they might have the same effect, while the black and blue curves (large and adenous) are fairly close, indicating that they might have the same effect as each other.

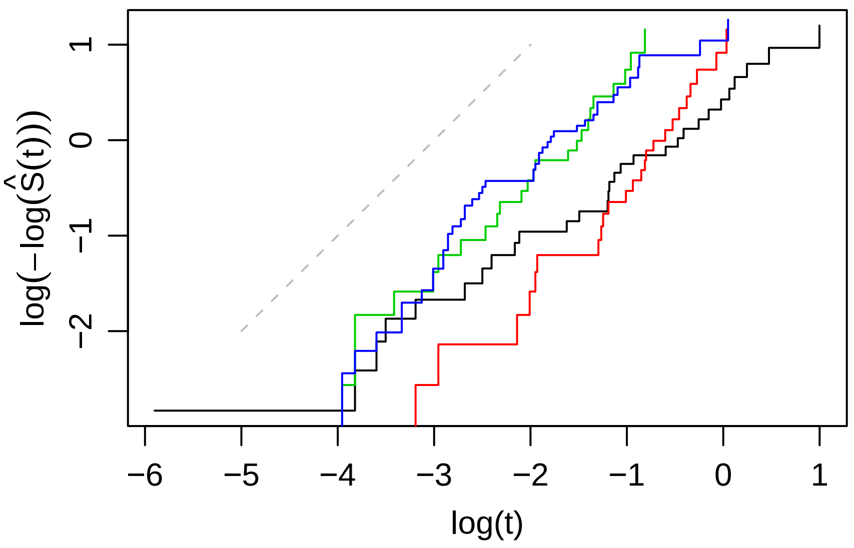
6.4.3 Visual assessment of model appropriateness

We have just seen an example of how the expected versus observed plot could be generalised to the parametric scenario. An alternative is to plot transformations of the Kaplan–Meier estimates versus transformations of time, as described in the last section. For the Weibull model, these transformations should be $\log(-\log(\hat{S}(t)))$ versus $\log t$. This again is easy to effect in R.

Consider only the model with cell type as a predictor (the no-predictor model is similar). (`temp` is the French word for time.)

```
km=survfit(S~c2+c3+c4)
m=1;n=km$strata[1]
temp=km$time[m:n]
cloglog=log(-log(km$surv[m:n]))
plot(log(temp),cloglog,type='s',xlab='log(t)',
  ylab=expression(log(-log(hat(S)(t)))))
m=n+1;n=n+km$strata[2]
temp=km$time[m:n]
cloglog=log(-log(km$surv[m:n]))
lines(log(temp),cloglog,type='s',col=2)
m=n+1;n=n+km$strata[3]
temp=km$time[m:n]
cloglog=log(-log(km$surv[m:n]))
lines(log(temp),cloglog,type='s',col=3)
m=n+1;n=n+km$strata[4]
temp=km$time[m:n]
cloglog=log(-log(km$surv[m:n]))
lines(log(temp),cloglog,type='s',col=4)
lines(c(-5,-2),c(-2,1),col=8,lty=2)
```

The estimated log–log lines should be roughly straight if the Weibull model is appropriate. It seems from the plot below that this is a valid assumption. The last line of code adds a dashed grey line to the plot with slope 1: the other lines should be parallel to this if the exponential special case holds: again, it seems they are.



6.4.4 Testing parameters

We thought above that some of the cell types might be merged. With parametric models, as with semi-parametric ones, we often wish to test whether covariates belong in the model or not. One way to do this is to use the likelihood ratio test on two models. This requires a bit of work. The other is to do the Wald test on a single fitted model. This also requires a bit of work. They are illustrated in the two examples below.

Example 2

Consider cell type as the only predictor. Let us try to merge small and squamous cells. We fit two models: one with parameters for three cell types (again keeping the fourth as the baseline) and one with parameters for just two, i.e. forcing the small and squamous cells to share the same parameter.

This can be done using the following commands:

```
wei1=survreg(S~c2+c3+c4,dist="w")
wei2=survreg(S~I(c2+c3)+c4,dist="w")
```

Twice the difference in log-likelihoods should be distributed according to a chi-squared distribution with one degree of freedom. Thus the *p*-value can be calculated as follows:

```
> teststat=2*(wei1$logl[2]-wei2$logl[2])
> pchisq(teststat,df=1,lower.tail=F)
[1] 0.6036761
```

As the *p*-value is large, we have no evidence to reject the hypothesis that actually, small and squamous cells have the same effect on survival. We would thus proceed with the simpler model in which they share a coefficient.

We can test whether large and squamous cells have the same coefficient in the same way:

```
wei3=survreg(S~I(c2+c3),dist="w")
teststat=2*(wei2$logl[2]-wei3$logl[2])
pchisq(teststat,df=1,lower.tail=F)
```

The *p*-value is 35%, so again there is no evidence against the simpler model.

Example 3

Now suppose that we use all the other covariates as well to predict survival. The output from the fitted model is:

```
> wei4=survreg(S~I(c2+c3)+perf+dur+age+prior+treat,dist="w")
> wei4
Call:
survreg(formula = S ~ I(c2 + c3) + perf + dur + age + prior +
    treat, dist = "w")

Coefficients:
  (Intercept)    I(c2 + c3)          perf           dur           age
-2.5370261600 -0.7735513168  0.0296084295  0.0006208002  0.0060580376
        prior         treat
-0.0022279946 -0.2473433228

Scale= 0.9547677

Loglik(model)= 37.7   Loglik(intercept only)= 7.2
        Chisq= 61.08 on 6 degrees of freedom, p= 2.7e-11
n= 137
```

We would like to test if the other covariates could have no effect or not, i.e. whether the population coefficients could be zero. We could go through, one by one, knocking the covariate out and then checking via the likelihood ratio test if its effect were significant, but this would be a bit of a pain. Doing the Wald test would be easier, but R does not output it automatically. Nevertheless, it can easily be done using the following commands:

```
> p=c();for(i in 1:length(wei4$coefficients))
    p[i]=pnorm(abs(wei4$coefficients[i]/sqrt(wei4$var[i,i])),lower.tail=F)
>  cbind(wei4$coefficients,p)
                                      p
(Intercept) -2.5370261600 1.310409e-04
I(c2 + c3)  -0.7735513168 1.865116e-05
perf         0.0296084295 1.560841e-09
dur          0.0006208002 4.704753e-01
age          0.0060580376 2.475195e-01
prior       -0.0022279946 4.591049e-01
treat       -0.2473433228 7.974881e-02
```

The first column in the `cbind` command output is the parameter name, the second the MLE of the parameter, the third the Wald test statistic. Note that only our binary cell type and performance appear to be significant. A model building approach similar to those described in chapter 4 could be used based on these p-values.

6.4.5 Selecting between competing parametric models

We have arbitrarily chosen the Weibull distribution for our models above. We might like to choose between a set of competing models, say Weibull versus log-normal versus exponential versus log-logistic. This can be done via Akaike's information criterion. Again, it is very simple to do this in R, as the following example shows.

```
> wei=survreg(S~I(c2+c3)+perf,dist="w")
> exp=survreg(S~I(c2+c3)+perf,dist="w",scale=1)
> lgl=survreg(S~I(c2+c3)+perf,dist="logl")
> lgn=survreg(S~I(c2+c3)+perf,dist="logn")
>
> extractAIC(wei)[2]
[1] -65.25415
> extractAIC(exp)[2]
[1] -66.9622
> extractAIC(lgl)[2]
[1] -77.21091
> extractAIC(lgn)[2]
[1] -69.84801
```

These suggest that, yes, the Weibull model is the most appropriate.

6.5 Frailty models

Thus far, for both the Cox PHM and parametric models, we have assumed that, if individuals have the same values of the covariates, they have the same survival function. Thus, suppose for example that we are interested in survival times following installation of a pacemaker. We might regress on sex and age in either a PHM or AFT model. But extra heterogeneities might exist, that we don't include in the model. It might be that smoking status influences survival times. Ideally, we would collect data on all factors we think will influence survival, but there will always be others that have an additional affect that we miss.

The effect of these additional terms gets bundled up into the random component of the model. In a linear regression model, they contribute to the error term $\epsilon \sim \mathrm{N}(0, \sigma^2)$, and so knowledge about them reduces the leftover randomness left to explain. In standard survival analyses, the error term in contained within the distribution of (possibly censored) survival times. There is less flexibility to inflate the variance to account for additional sources of variability. However, such extra variability can be incorporated using *frailty* models.

A frailty model assumes that the hazard for individual i is multiplied by an unobserved random effect ω_i. This random effect might be normal, say, but is usually modelled via a gamma distribution, as it then has strictly positive support (as this is necessary for a hazard function). The mean of ω is 1, but it has variance σ^2, which is an additional parameter to estimate. If $\sigma \approx 0$, then no frailty model is needed, as all the variance in the process is captured by the parametric model for survival times (if a parametric model is used), or the induced distribution of survival times in a Cox model. If $\sigma > 0$, then there is additional between individual variability in survival functions.

If $\omega_i > 1$ then individual i is more frail than typical for his/her other covariates. If $\omega_i < 1$ then i has a lower frailty than his/her peers.

For example, if i smokes and j doesn't, we might have $\omega_i = 1.2$ and $\omega_j = 0.9$ in the above pacemaker example.

Frailty models are much harder to fit to data than the models we have considered thus far, as the random components ω_i act like extra parameters to be estimated (though they are not considered to be parameters in the classical framework), and thus the "parameter" vector becomes very large. Since they are competing with the inherent variability in the process when all survival rates are the same, it can also be difficult to identify them. We therefore consider a special case of frailty models, called *shared frailty*, that has attracted more research efforts and thus may be easier to fit.

6.5.1 Shared frailty

Shared frailty models share many of the facets of individual frailty models. The hazard for each individual is multiplied by a random effect, which comes from a distribution (e.g. gamma) with mean 1 and unknown variance σ^2. What differs, though, is that this unknown random effect is shared among a group that have the same characteristics. For instance, you might consider the duration of unemployment in Singapore as predicted by sex, age, and place of residence. If place of residence were defined broadly, e.g. North, South, East, West or Central, then we could use this a factor in a Cox PHM, say. If it were much more localised than this, e.g. Jurong, Dover, Clementi, ..., there would be too many factors for this to be worthwhile (unless a very large data set were collected). One approach would be to drop place of residence altogether, but this would neglect the effect of place of residence on survival and might influence the estimates of the effect of sex and age. In this situation, we might treat place of residence as a random effect, drawn from a hypothetical population of residential areas. They are allowed to influence survival, but we do not estimate them directly.

The model is still quite difficult to fit. R uses penalised likelihood, an approach that involves inverting a square matrix of size number of parameters + number of groups. See Therneau *et al.* (2003, J. Comp. Graph. Stat. 12:156–75) for details. Other approaches include the Expectation–Maximisation algorithm, and data augmentation within an MCMC routine. If R fails to fit the model, you either have to accept it, and discard frailty from your model, or try to code an alternative up yourself.

6.5.2 Example: rats

We consider an example due to Mantel et al. (1977). They describe an experiment in which rats belonging to litters were purchased and culled until they had 50 litters of 3 rats: one of which was allocated a drug, the others a placebo. Rats from the same litter are siblings, and so may have more similar survival experiences than rats from different litters.

One approach would be to ignore litter. Another would be to treat litter as a factor, with each litter getting its own parameter (except the first one, which is treated as the baseline). Somewhere in between these extremes is the shared frailty model, which allows litter to influence survival, with a parameter characterising between litter differences. (A fourth possibility, but a very bad one, would be to stratify over litter. Since there are only three rats per litter, the estimates of the baseline hazard would be terribly uncertain, so it is not recommended.)

Here are R commands for fitting these models.

```
attach(read.table("rats.dat",header=TRUE))
library(survival)
wei1=survreg(Surv(t,delta)~treat)
wei2=survreg(Surv(t,delta)~treat+frailty.gaussian(litter))
wei3=survreg(Surv(t,delta)~treat+factor(litter))
```

The function `frailty.gaussian` is used to allocate a normal frailty on top of the hazard function. The `survival` package is set up so that only a normal distribution can be used on the parametric models, but either normal or gamma can be used in Cox models. I am not sure why this is.

The output from the three models are as follows. First, the one that ignores litter:

```
> wei1
Call:
survreg(formula = Surv(t, delta) ~ treat)

Coefficients:
(Intercept)       treat
  4.9837346  -0.2389354

Scale= 0.2640395

Loglik(model)= -242.3   Loglik(intercept only)= -246.3
        Chisq= 8.01 on 1 degrees of freedom, p= 0.0047
n= 150
```

Second, the frailty model:

```
> wei2
Call:
survreg(formula = Surv(t, delta) ~ treat + frailty.gaussian(litter))

                         coef   se(coef) se2    Chisq   DF   p
(Intercept)               4.871 0.0637   0.0575 5855.80  1.0 0.0000
treat                    -0.182 0.0665   0.0651    7.51  1.0 0.0061
frailty.gaussian(litter)                          22.84 15.8 0.1100

Scale= 0.192

Iterations: 8 outer, 38 Newton-Raphson
     Variance of random effect= 0.0239
Degrees of freedom for terms=  0.8  1.0 15.8  0.9
Likelihood ratio test=41.2  on 16.5 df, p=0.000667  n= 150
```

Finally, the factor model:

```
> wei3
Call:
survreg(formula = Surv(t, delta) ~ treat + factor(litter))

Coefficients:
     (Intercept)            treat  factor(litter)2  factor(litter)3
     4.902302090     -0.218613033      4.170743001      4.176315766
 factor(litter)4  factor(litter)5  factor(litter)6  factor(litter)7
     3.954011403      3.910864063     -0.178212052     -0.129169562
 factor(litter)8  factor(litter)9 factor(litter)10 factor(litter)11
     0.022127022      4.009684327     -0.157142683     -0.082910266
factor(litter)12 factor(litter)13 factor(litter)14 factor(litter)15
     4.018464823     -0.948505252     -0.006715644      3.926295778
factor(litter)16 factor(litter)17 factor(litter)18 factor(litter)19
     3.828593730      3.974426353      3.951440970      3.802347567
factor(litter)20 factor(litter)21 factor(litter)22 factor(litter)23
    -0.321854056      4.176315766      4.176315766     -0.017264637
factor(litter)24 factor(litter)25 factor(litter)26 factor(litter)27
     3.901575902      4.176315766      3.910864063      3.864771844
factor(litter)28 factor(litter)29 factor(litter)30 factor(litter)31
    -0.155006110     -0.279064002     -0.388280306      3.901592092
factor(litter)32 factor(litter)33 factor(litter)34 factor(litter)35
    -0.236967948      0.020322682     -0.119415487     -0.203992760
factor(litter)36 factor(litter)37 factor(litter)38 factor(litter)39
     0.011923603      4.176315766      0.043012646     -0.358728581
factor(litter)40 factor(litter)41 factor(litter)42 factor(litter)43
    -0.346366208     -0.105691461     -0.300743812     -0.143638827
factor(litter)44 factor(litter)45 factor(litter)46 factor(litter)47
     4.125717135      3.856616241      3.581296425     -0.012036192
factor(litter)48 factor(litter)49 factor(litter)50
     4.032005596      0.038732756      0.032876999

Scale= 0.2026994

Loglik(model)= -201.8   Loglik(intercept only)= -246.3
```

```
        Chisq= 89.06 on 50 degrees of freedom, p= 0.00056
n= 150
```

Obviously, most of the output from the factor model is uninteresting. Instead, the frailty model much more simply characterises the variance in these output in summary form, namely that the effect of frailties in litter i is to multiply the hazard by a $\mathrm{N}(1, 0.15^2)$ variate. Note, though, that 0.15 is not the standard deviation of the exponentiated factor coefficients, it is much smaller. Information on these coefficients is pooled, and the coefficients are thus pulled (or shrunk) towards 0.

The parameter for treatment and the baseline hazard are very similar across the models, and indeed the frailty term is not significant. We would therefore be likely to remove it from the model.

6.6 Other topics

We have considered events that may only occur once, or have only been interested in the first time they occur. But some events may occur repeatedly to one individual. Criminals may be rereleased from gaol after being rearrested, patients may survive one heart attack and then suffer another, leukæmia patients may go into remission several times only for the illness to return. These situations call for *recurrent event survival analysis.* At its most basic level, recurrent event survival analysis can be performed just as we extended the Cox model, by splitting individuals up into records for each recurrence. The number of previous events can be included as a covariate, or we might stratify over number of previous events.

It is also possible for there to be two or more event types. We might be interested in death due to cancer *and* death due to cardiovascular disease. Obviously, one patient can only undergo *one* of these events. The two types of failure can be thought of as competing against each other, so this scenario is called *competing risks survival analysis.* As briefly mentioned earlier in the notes, this can be problematic, as the events might not be independent of each other, but analyses depend upon this. A sensitivity analysis can be performed to assess how results might be influenced by non-independence, but there is no direct method to assess the assumption of non-independence. For a nice discussion, see Kleinbaum and Klein (2005)

Throughout the lectures, I have made oblique references to all the methods requiring that failures in different individuals must occur independently, and warning of dire consequences if you try to use survival methods for situations when failures are correlated. One such example is in infectious diseases, where we might be tempted to treat infection as an interval censored failure time. This would be wrong, however, as diseases spread between individuals, and as a result the hazard for individual i will increase when individual j gets infected if i and j are in contact with each other. Models for epidemics are described in stochastic processes II. Anyone interested in inference for such situations should contact me, as this is the topic of much of my research!