**Notes for Tuesday April 7, 1998**

# Parametric Models

Parametric models for survival data have fallen out of fashion with the advent of less parametric approaches such as the Cox regression model. Yet they remain a useful tool, and usually can be fitted faster than the corresponding semiparametric model. Moreover parametric models are often used in commercial and industrial applications of survival analysis such as *accellerated life testing*. An excellent introduction to industrial applications of survival analysis is Barlow and Proschan (1981) *Statistical Theory of Reliability and Life Testing*, To Begin With Press, Silver Springs MD[1].

Parametric models are often used for regression on covariates.

- The *exponential* model has $\lambda(t) \equiv \lambda > 0$, density $f(t) = \lambda e^{\lambda t}$, and $E(T) = 1/\lambda$, $\mathrm{Var}(T) = 1/\lambda^2$. If there are covariates $z_i$ it is natural to build a regression model for $\log E(T)$, which is equivalent to
$$\log \lambda_i = \beta' z_i \quad \text{or} \quad \lambda_i = e^{\beta' z_i}$$
- The *gamma* model for failure times has hazard function density $f(t) = [\lambda^\alpha/\Gamma(\alpha)]t^{\alpha-1}e^{\lambda t}$, for $\alpha, \lambda > 0$, and $E(T) = \alpha/\lambda$ and $\mathrm{Var}(T) = \alpha/\lambda^2$, and a messy hazard function $\lambda(t)$. There are two common ways to build a linear regression model for Gamma times:
  - A regression model for the *log-mean*, $\log E(T)$, is equivalent to saying $\log \lambda_i = \beta' z_i$
  - A regression model for the *inverse mean*, $1/E(T)$, is equivalent to saying $1/\lambda_i = \beta' z_i$

  The hazard function is $\lambda_i(t) = \left[\int_0^\infty (1 + \frac{u}{t})^{\alpha-1} e^{-\lambda_i u} du\right]^{-1}$, which is increasing in $\lambda_i$.
- The *Weibull* distribution has hazard function $\lambda(t) = \lambda^\alpha \alpha t^{\alpha-1}$, and if we again make $\log \lambda_i = \beta' z_i$ this becomes
$$\lambda_i(t) = \alpha t^{\alpha-1} e^{\alpha\beta' z_i}$$
  Note two things:
  1. The Weibull regression model is a parametric version of the *proportional hazards model*, for it is of the form
  $$\lambda_i(t) = \lambda_0(t) e^{\beta' z_i}$$
  2. Note that if we replace $t$ with $t \cdot \exp\{\beta' z_i\}$ and do the appropriate change-of-variables formula from calculus we obtain
  $$\exp\{\beta' z_i\}\lambda_i(t \cdot \exp\{\beta' z_i\}) = \alpha$$
  thus if we "accellerate" time by a factor of $\exp\{\beta' z_i\}$, we obtain a hazard function that does not depend on time or covariates at all (i.e. back to exponential failure times).

  Turning the problem around if under harsh testing conditions (extreme $z_i$) a computer chip fails quickly but with an exponential distribution of failure times, then under more normal circumstances (decellerate by $\exp\{\beta' z_i\}$) we can expect the chip's failure times to have a Weibull distribution with a long, predictable lifetime.

[1] A 1975 edition was published by Hold, Reinhardt and Winston.

## Feigl and Zelen Leukemia Study (Venables and Ripley, pp. 350 ff)

This data gives survival times and white blood counts for 33 leukemia patients. The variables are:

```
wbc:     white blood count
ag:      a test result, present or absent
time:    survival time in weeks
```

The Sources are D.R. Cox and D. Oakes (1984) *Analysis of Survival Data*, Chapman & Hall, p. 9, who themselves took the data from P. Feigl & M. Zelen (1965) *Estimation of exponential survival probabilities with concomitant information.* Biometrics, 21, 826-838.

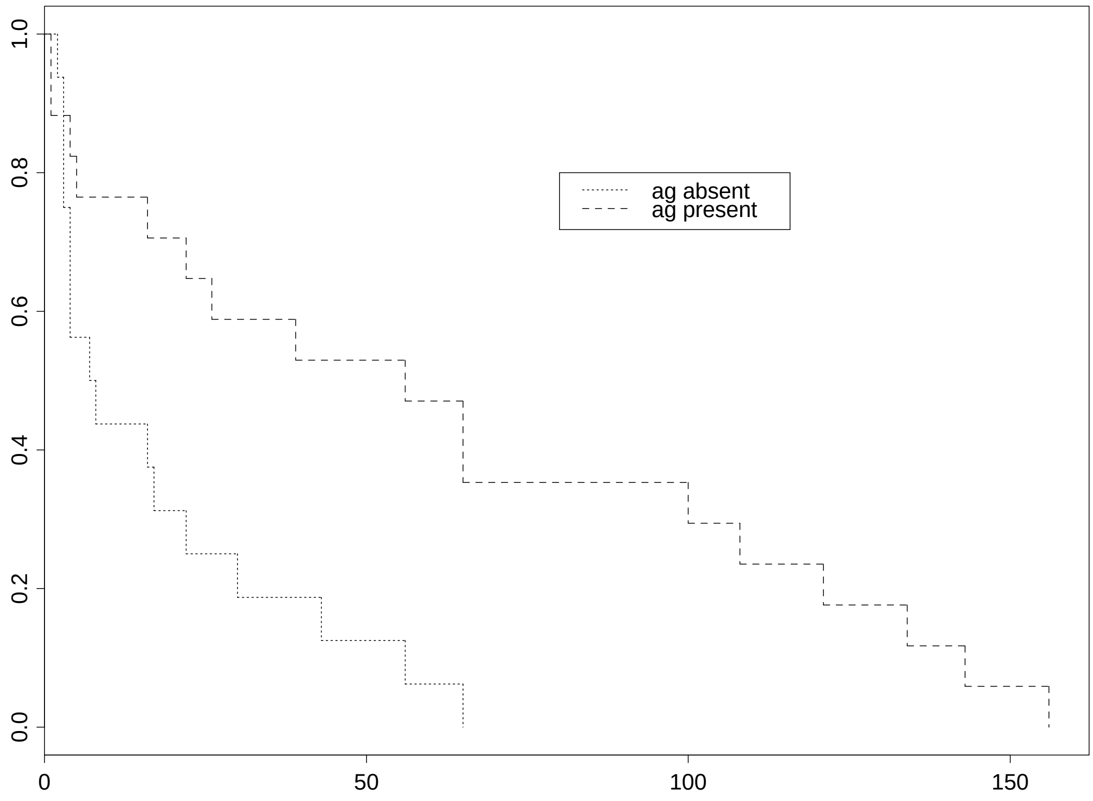
The data are,

```
402 > library(MASS)
402 > leuk
      wbc      ag time
 1   2300 present   65
 2    750 present  156
 3   4300 present  100
 4   2600 present  134
 5   6000 present   16
 6  10500 present  108
 7  10000 present  121
 8  17000 present    4
 9   5400 present   39
10   7000 present  143
11   9400 present   56
12  32000 present   26
13  35000 present   22
14 100000 present    1
15 100000 present    1
16  52000 present    5
17 100000 present   65
18   4400  absent   56
19   3000  absent   65
20   4000  absent   17
21   1500  absent    7
22   9000  absent   16
23   5300  absent   22
24  10000  absent    3
25  19000  absent    4
26  27000  absent    2
27  28000  absent    3
28  31000  absent    8
29  26000  absent    4
30  21000  absent    3
31  79000  absent   30
32 100000  absent    4
33 100000  absent   43
```

The first thing one might wish to do with these data is plot the two survival curves, via

```
402 > attach(leuk)
402 > plot(survfit(Surv(time) ~ ag), lty=c(2,3))
402 > legend(75,.8,c("ag absent","ag present"),lty=2:3)
```

Recall that `surfit` used in this way computes an estimate of a survival curve for censored data using the Kaplan-Meier method (it can also use another method based on the cumulative hazard $\Lambda(t) = -\log S(t)$, called the Fleming-Herrington method) or computes the predicted survivor function for a Cox proportional hazards model. In our case the model is just the two levels of `ag`. The picture of the Kaplan-Meier estimate looks like this:




There are clearly differences in the curves, but whether they are significant or not is not obvious. One could directly proceed with `survdiff` as we have done in the past, or fit a Cox model and look at the likelihood ratio and other tests. We will proceed as follows

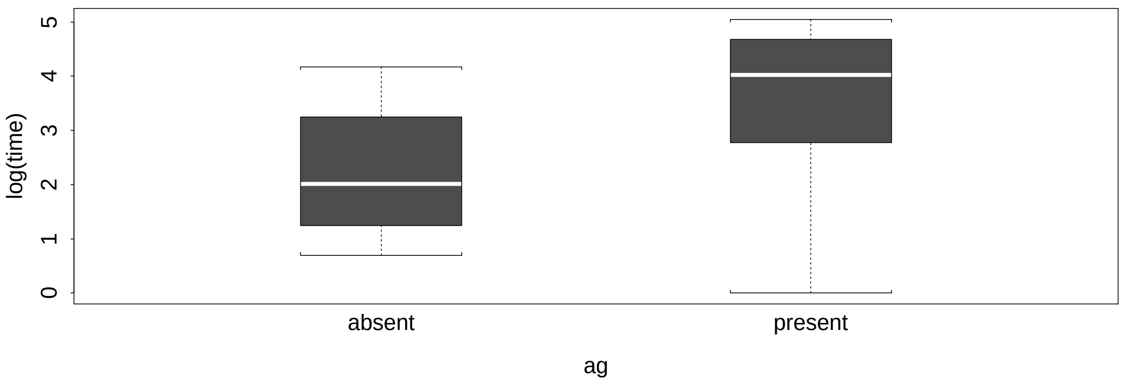
- Parametric models based on Gamma errors and the `glm` function,
- Parametric models, based on the exponential and Weibull distributions using `survreg`, and
- The family of Cox models.

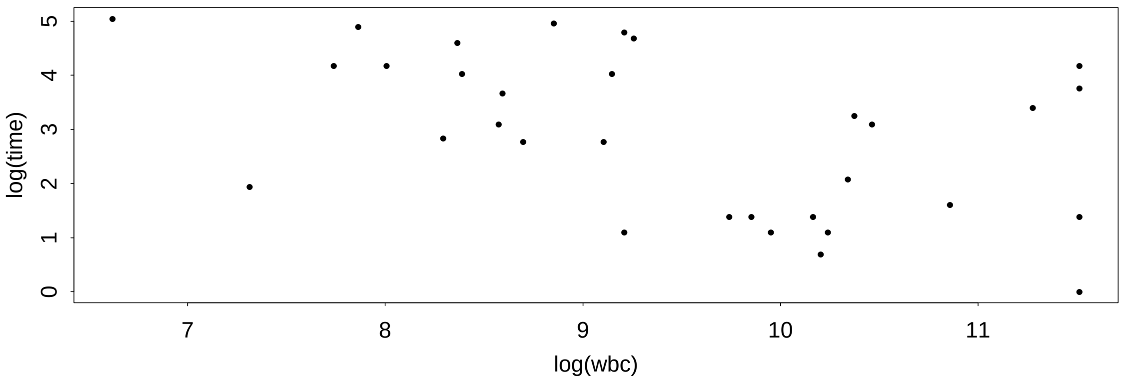
We will **not** do a lot of the theory, but will attempt to push the example through with just enough to make sense of what we are doing.

## Fitting using GLM

If we model the survival times as Gamma, then following our general comments about Gamma models above, we probably want to model the log expected survival time with a linear model. To see what kind of linear model might be appropriate we begin with an exploratory plot.

```
402 > par(mfrow=c(2,1))
402 > plot(log(time) ~ ag + log(wbc)) # cool way to get two different plots!
```




"Ag" looks important, "log(wbc)" less so. Let's see how these come out in a `glm` fit. To make interpreting the parameters slightly easier we will ask for "treatment contrasts" so we can see the effect of a unit change in the covariate.

```
402 > lwbc <- log(wbc)
402 > options(contrasts=c("contr.treatment","contr.poly"))
402 > leuk.glm <- glm(time ~ ag*lwbc, Gamma(link=log))
402 > summary(leuk.glm,dispersion=1) # force exponential model summary

Call: glm(formula = time ~ ag * lwbc, family = Gamma(log))
Deviance Residuals:
       Min       1Q     Median        3Q      Max
 -1.992227 -1.21155 -0.3270025 0.2160396 1.564344
```

```
Coefficients:
                  Value Std. Error   t value
(Intercept)  4.3291270  1.9661806  2.201795
         ag  4.1481711  2.5702903  1.613892
       lwbc -0.1525468  0.2026831 -0.752637
    ag:lwbc -0.3291857  0.2668885 -1.233420

(Dispersion Parameter for Gamma family taken to be 1 )

    Null Deviance: 58.13839 on 32 degrees of freedom

Residual Deviance: 38.55468 on 29 degrees of freedom

Number of Fisher Scoring Iterations: 6

Correlation of Coefficients:
        (Intercept)         ag       lwbc
     ag -0.7649644
   lwbc -0.9918835   0.7587556
ag:lwbc  0.7532658  -0.9907268 -0.7594297
```

Note the high correlations among the parameters!

```
402 >  print(anova(leuk.glm))
Analysis of Deviance Table

Gamma model

Response: time

Terms added sequentially (first to last)
        Df Deviance Resid. Df Resid. Dev
   NULL                    32   58.13839
     ag  1 11.94009        31   46.19829
   lwbc  1  5.87920        30   40.31909
ag:lwbc  1  1.76441        29   38.55468
```

The model appears to be gitting well, but the interaction doesn't look necessary. Surprisingly (recall the plot), "log(wbc)" is hanging in there.

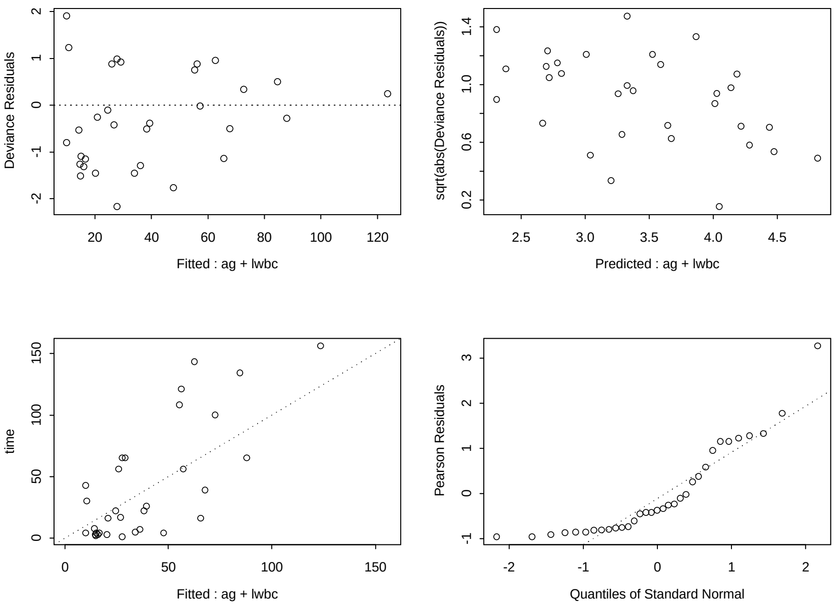
Taking out the interaction we get

```
402 >  leuk.glm <- update(leuk.glm,  ~ .-ag:lwbc)
402 > summary(leuk.glm, dispersion=1)

Call: glm(formula = time ~ ag + lwbc, family = Gamma(log))
```

```
Deviance Residuals:
       Min        1Q     Median        3Q      Max
 -2.174855 -1.266119 -0.4246385 0.4961872 1.904819

Coefficients:
                 Value Std. Error   t value
(Intercept)  5.8134506  1.2931888  4.495438
         ag  1.0182978  0.3492227  2.915898
       lwbc -0.3042296  0.1318634 -2.307157

(Dispersion Parameter for Gamma family taken to be 1 )

    Null Deviance: 58.13839 on 32 degrees of freedom

Residual Deviance: 40.31909 on 30 degrees of freedom

[some output omitted]

402 > 1-pchisq(40.3,30)
[1] 0.09918332
```

Fitting the two main effects did pretty well. Here are the residual plots

```
402 > par(mfrow=c(2,2))
402 > plot(leuk.glm)
```




These don't look bad, but there is some evidence of decreasing variance in the deviance residuals as the fitted value increases. Of course the sample size also decreases as the fitted value increases.

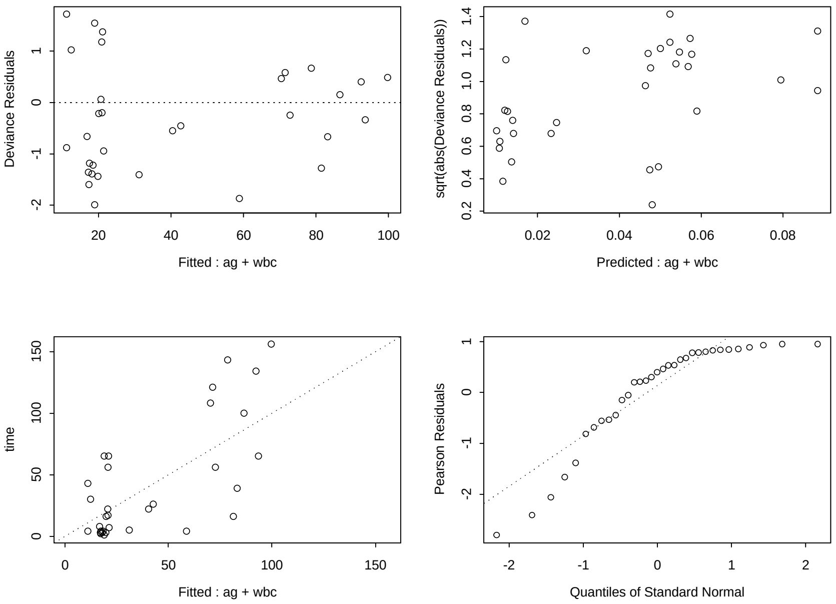
The fitted vs time plot suggests that we have got some of the signal but perhaps not all of it, and the Pearson residuals look a little heavy-tailed.

Recalling that for a Gamma, $E(T) = \alpha/\lambda$, an alternative way to build a model would be to build a linear model for $1/E(T)$—this is called the "inverse link". We can compare our log-linear fit with an inverse-linear fit by changing the "link" function.

```
402 > leuk.glmi <- glm(time ~ ag*wbc, Gamma(inverse))
402 > (summary(leuk.glmi, dispersion=1)
+ )

Call: glm(formula = time ~ ag * wbc, family = Gamma(inverse))
Deviance Residuals:
       Min        1Q     Median        3Q      Max
 -1.890629 -1.299155 -0.4143898 0.3559901 1.909283

Coefficients:
                    Value   Std. Error     t value
(Intercept)  5.635111e-02 1.887885e-02  2.98488018
         ag -4.780969e-02 1.932473e-02 -2.47401586
        wbc -2.030302e-08 4.239835e-07 -0.04788634
     ag:wbc  5.926747e-07 5.232047e-07  1.13277789

(Dispersion Parameter for Gamma family taken to be 1 )

    Null Deviance: 58.13839 on 32 degrees of freedom

Residual Deviance: 39.73216 on 29 degrees of freedom

[some output omitted]

402 > anova(leuk.glmi)
Analysis of Deviance Table

Gamma model

Response: time

Terms added sequentially (first to last)
       Df Deviance Resid. Df Resid. Dev
  NULL                    32   58.13839
    ag  1 11.94009        31   46.19829
   wbc  1  5.31608        30   40.88221
ag:wbc  1  1.15005        29   39.73216
```

The results are essentially the same in this case. Let's fit the additive model and look at its residuals.

```
402 >  leuk.glmi _ update(leuk.glmi, . ~ . - ag:wbc)
402 >  leuk.glmi
Call:
glm(formula = time ~ ag + wbc, family = Gamma(inverse))

Coefficients:
 (Intercept)          ag         wbc
  0.04578319 -0.03608559 4.27565e-007

Degrees of Freedom: 33 Total; 30 Residual
Residual Deviance: 40.88221
402 > 1-pchisq(40.88,30)
[1] 0.0889144
402 > plot(leuk.glmi)
```




The likelihood ratio chi-square (residual deviance) suggests about the same fit as for the log-linear formulation. Looking at the residual plots, the nonconstant variance in the deviance residuals seems to have been cleaned up quite a bit, but (a) there is some oscillation in the residuals, suggesting that a transformation is needed (you can also see this in the fitted vs time plot) and (b) the Pearson residuals look a little light-tailed. These things suggest that we haven't really nailed the model yet. At this point I think I slightly prefer the log-linear formulation, for this data.

## Fitting using `survreg`

For fitting Weibull and even eponential models it may be easiest to use the special function `survreg` built into SPLUS. There is no Weibull family in `glm`, and `survreg` incorporates some details into the fitting of survival curves that are glossed over with the `glm` fit (for example, what to do about censored observations).

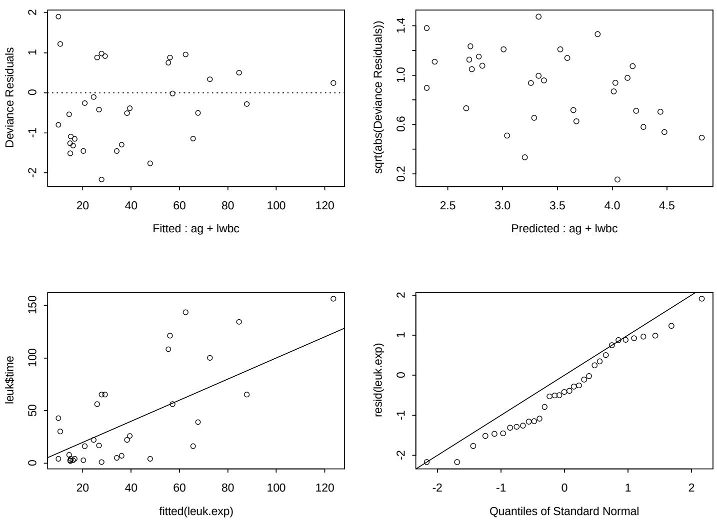
First we re-do the exponential (gamma) analysis, using `survreg`.

```
402 > attach(leuk)
402 > options(contrasts=c("contr.treatment","contr.poly"))
402 > leuk.exp _ survreg(Surv(time) ~ ag*lwbc,dist="exp")
402 > summary(leuk.exp)

[some output omitted]

Coefficients:
             Value Std. Error z value       p
(Intercept)  4.343      1.638   2.651 0.00802
         ag  4.135      2.370   1.745 0.08097
       lwbc -0.154      0.168  -0.915 0.36000
    ag:lwbc -0.328      0.246  -1.332 0.18298

Extreme value distribution: Dispersion (scale) fixed at 1
Degrees of Freedom: 33 Total; 29 Residual
-2*Log-Likelihood: 105

[some output omitted]

402 > leuk.exp _ update(leuk.exp,. ~ . -ag:lwbc)
402 > summary(leuk.exp)

Call:
survreg(formula = Surv(time) ~ ag + lwbc, dist = "exp")
Deviance Residuals:
   Min    1Q Median    3Q Max
 -2.17 -1.27 -0.425 0.496 1.9

Coefficients:
             Value Std. Error z value         p
(Intercept)  5.815      1.263    4.60 4.15e-006
         ag  1.018      0.364    2.80 5.14e-003
       lwbc -0.304      0.124   -2.45 1.44e-002

Extreme value distribution: Dispersion (scale) fixed at 1
Degrees of Freedom: 33 Total; 30 Residual
-2*Log-Likelihood: 106

Number of Newton-Raphson Iterations: 4

Correlation of Coefficients:
     (Intercept)     ag
  ag -0.418
```

```
lwbc -0.980       0.287
```

There seems to be a bug in the "plot" method for `survreg` but the plots that don't get made automatically are easy to reproduce:

```
402 > plot(leuk.exp)
Error in plot: Lengths of x and y must match
Dumped
402 > plot(fitted(leuk.exp),leuk$time)
402 > abline(0,1)
402 > qqnorm(resid(leuk.exp),type="pearson")
402 > abline(0,1)
```

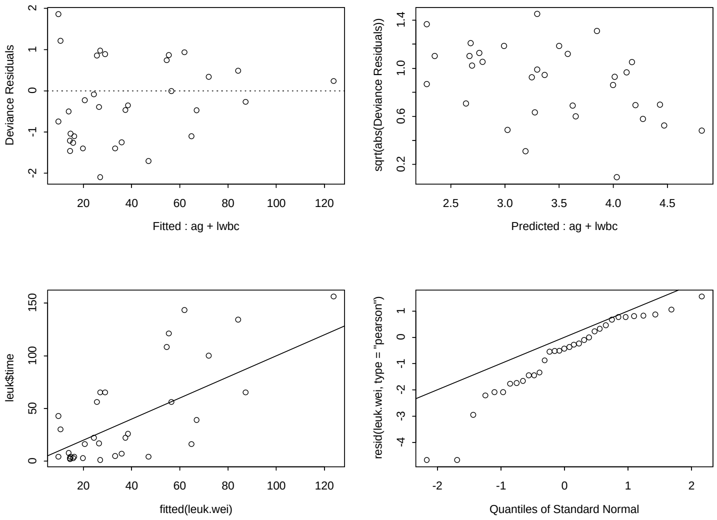



As expected, this looks essentially like our Gamma fit with `glm` earlier.

Now let's look at a Weibull fit:

```
402 > leuk.wei _ survreg(Surv(time) ~ ag*lwbc) # Weibull is the default
402 > summary(leuk.wei)

[some output deleted]

Coefficients:
             Value Std. Error z value       p
(Intercept)  4.342      1.680   2.585 0.00973
         ag  4.174      2.439   1.712 0.08696
       lwbc -0.155      0.173  -0.898 0.36899
    ag:lwbc -0.332      0.253  -1.310 0.19026

[some output deleted]
```

```
402 > leuk.wei _ update(leuk.wei, . ~ . - ag:lwbc)
402 > summary(leuk.wei)

Call:
survreg(formula = Surv(time) ~ ag + lwbc)
Deviance Residuals:
  Min    1Q Median    3Q  Max
 -2.1 -1.21 -0.399 0.483 1.86

Coefficients:
            Value Std. Error z value         p
(Intercept)  5.85      1.323    4.42 9.66e-006
         ag  1.02      0.378    2.70 6.95e-003
       lwbc -0.31      0.131   -2.36 1.81e-002

Extreme value distribution: Dispersion (scale) = 1.040689
Degrees of Freedom: 33 Total; 29 Residual
-2*Log-Likelihood: 106

[some output omitted]

402 >  plot(leuk.wei)
Error in plot: Lengths of x and y must match
Dumped
402 > plot(fitted(leuk.wei),leuk$time)
402 > abline(0,1)
402 > qqnorm(resid(leuk.wei,type="pearson"))
402 > abline(0,1)
```




The general impression one gets is that the Exponential model provided at least as good a fit.

Venables and Ripley (pp. 354–355) give some ideas for testing whether an accellerated lifetime model is appropriate for these data.

## Fitting using Cox's proportional hazards model

Recall the basic form of the hazard function for a Cox model

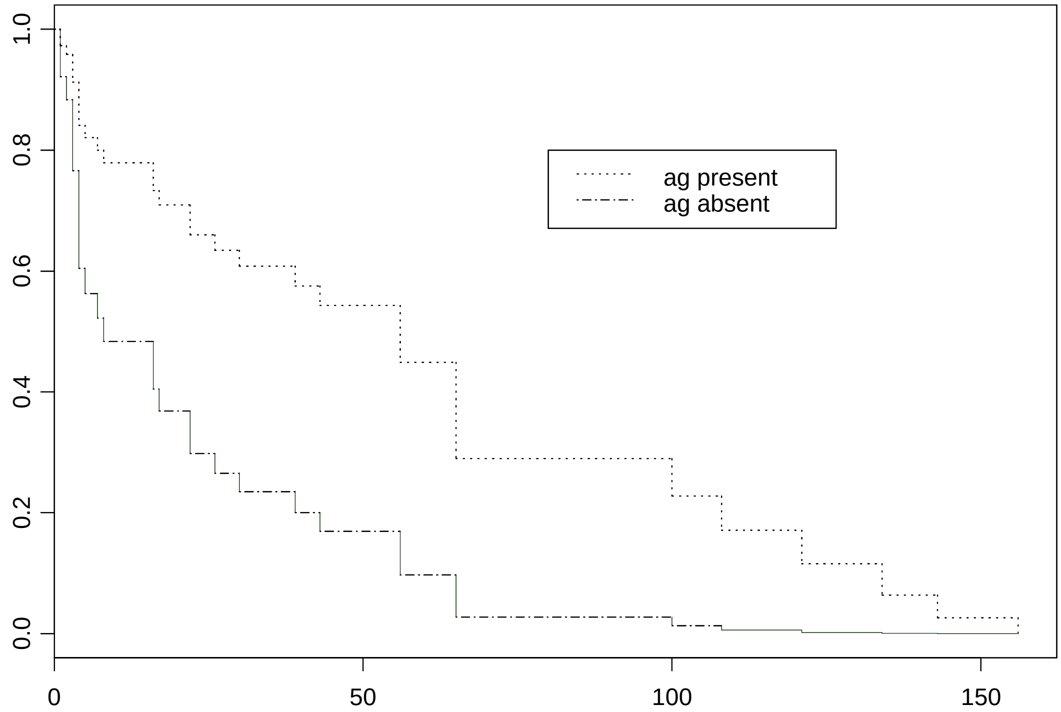
$$\lambda(t; x_i) = \lambda_0(t) \exp\{\beta' z_i\}$$

The dichotomous `ag` variable can be handled in one of two ways in this data set:

- It can be treated as a covariate $z$ that goes into the regression (proportional hazards) part of the model; or
- It can be treated as a "stratum" variable, so two different baseline risk functions $\lambda_0(t)$ is calculated for the two different levels of `ag`.

Venables and Ripley (pp 357–358) illustrate the two approaches. Here I will only show the first approach, and illustrate a function that is useful for testing the assumption of proportional hazards.

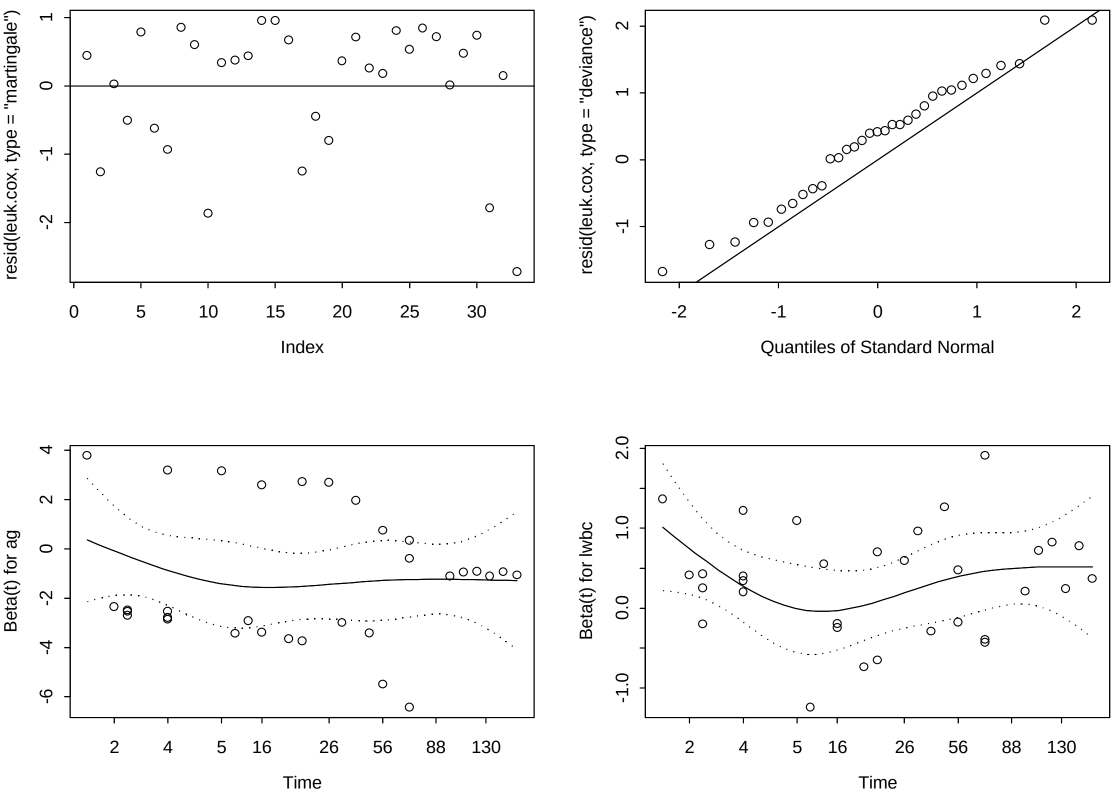
```
402 > leuk.cox _ coxph(Surv(time) ~ ag + lwbc)
402 > summary(leuk.cox)
Call:
coxph(formula = Surv(time) ~ ag + lwbc)

  n= 33

       coef exp(coef) se(coef)     z      p
  ag -1.069     0.343    0.429 -2.49 0.0130
lwbc  0.368     1.444    0.136  2.70 0.0069

     exp(coef) exp(-coef) lower .95 upper .95
  ag     0.343      2.913     0.148     0.796
lwbc     1.444      0.692     1.106     1.886

Rsquare= 0.377   (max possible= 0.994 )
Likelihood ratio test= 15.6  on 2 df,   p=0.000401
Wald test            = 15.1  on 2 df,   p=0.000537
Efficient score test = 16.5  on 2 df,   p=0.000263

402 > par(mfrow=c(1,1))
402 > plot(survfit(leuk.cox,newdata=data.frame(ag=c("present","absent"),
+ lwbc=c(9.5,9.5))),lty=2:3)
402 > legend(80,.8,c("ag present","ag absent"),lty=2:3)
```




It is interesting to compare the shape of these curves with the shapes of the curves on p 358 of Venables and Ripley, where `ag` is treated as a stratum variable (and hence can influence the hazard non-proportionally).

SPLUS has a function, `cox.zph` which can test for whether the coefficients $\beta$ in $\exp\{\beta' z_i\}$ should be replaced with $\beta + \theta g(t)$, for some coefficient $\theta$ and some predetermined function $g(t)$. In addition to plots of the martingale and deviance residuals below, I have illustrated using the `cox.zph` function.

```
402 > par(mfrow=c(2,2))
402 > plot(resid(leuk.cox,type="martingale"))
402 > abline(0,0)
402 > qqnorm(resid(leuk.cox,type="deviance"))
402 > abline(0,1)
402 > tmp _ cox.zph(leuk.cox)
402 > print(tmp)
           rho  chisq     p
    ag -0.1317 0.6910 0.406
  lwbc -0.0442 0.0492 0.824
GLOBAL      NA 0.7097 0.701
402 > plot(tmp)
```




The martingale and deviance residual plots actually look pretty good!

The two plots from `cox.zph` suggest that the coefficient of `lwbc` may be time-varying (horizontal fitted curves would indicate no time-varying component) but the $p$-values do not seem to support that.

Venables and Ripley, pp. 363–380, have several extended examples and these are well worth perusing.