UNIVERSITY OF ESSEX
INSTITUTE FOR SOCIAL AND ECONOMIC RESEARCH
Professor Stephen P. Jenkins <stephenj@essex.ac.uk>

Essex Summer School course 'Survival Analysis'
and
EC968. Part II: Introduction to the analysis of spell duration data

Lesson 2. The shapes of hazard, survivor and related functions

Contents

1 Aims

The aims of this lesson are

- to gain more familiarity with the functional forms typically used for hazard, survivor, and related, functions in both continuous and discrete time modelling, and the derivation of key statistics such as the median duration.
- to gain more familiarity with Stata and its capabilities (especially generate, display, and graph commands).

If time is short, many of the details of this Lesson can be skipped (e.g. draw only a few graphs once you know the principles involved).

We first look at (1) continuous time models and then (2) discrete time models. In each case, we will examine the shapes of hazard and survivor functions and derive summary statistics such as the median duration for different parameter values.

Think of what we do here as illustrating chapter 2 of *Survival Analysis*. We take a series of duration models, each of which is characterized by a set of parameters (e.g. in the Weibull model, the shape parameter α and the index function λ_i, which depends on regression coefficients β and the characteristics of person i, X_i). We consider what the shapes of the hazard and survivor functions are for particular values of those parameters; it is as if we are *simulating* values for a hypothetical individual i. In later chapters we consider how to *estimate* the parameters.

2 Continuous time models

I will first work through the Weibull model case, and then ask you, as an exercise, to repeat the analysis for the log-logistic model.

2.1 Weibull model

Recall that for this model:

Hazard function	$h(t) = \alpha\lambda t^{\alpha-1}$
Survivor function	$S(t) = \exp(-\lambda t^{\alpha})$
Failure function	$F(t) = 1 - S(t)$
Density function	$f(t) = \alpha\lambda t^{\alpha-1} \exp(-\lambda t^{\alpha})$
Integrated hazard function	$H(t) = -\ln[S(t)] = \lambda t^{\alpha}$
	where $\lambda = \exp(\beta'X)$, $\exp(x) = e^{x}$, and $\alpha > 0$

The Exponential model is the case when $\alpha = 1$. Beware that alternative parameterisations of this model exist in the literature! Remember too that the Weibull model can also be characterized as an Accelerated Failure Time model as well as a Proportional Hazard model (it is the PH representation used above).

The median duration t' satisfies $S(t') = 0.5$.

The median can be derived in several ways empirically.
(i) simply read off the value of t from the graph of the survivor function at $S(t') = 0.5$.
(ii) print out combinations of t and $S(t)$ and interpolate using this. (Use the **list** command.)
(iii) Derive the exact value using the closed form expression for the median which one can derive by inverting the survivor function. From above

$$t' = [-\ln(0.5) /\lambda]^{(1/\alpha)} = [\ln(2)/\lambda]^{(1/\alpha)}$$

which can calculated using the **display** command. For other quanttiles in addition to median which is the fiftieth percentile, *p*50), one would substitute the relevant quantile in the formula above in place of 0.5. For example, the lower quartile (*p*25) of the duration distribution would be given by $[-\ln(0.25) /\lambda]^{(1/\alpha)} = [\ln(4) /\lambda]^{(1/\alpha)}$

The mean duration t^* is given by

$$t^* = (1/\lambda)^{(1/\alpha)}\Gamma(1+(1/\alpha))$$

where $\Gamma(x)$ is the Gamma function, calculated in Stata using the expression **exp(lngamma(*x*))**. Again beware that different parameterisations lead to different expressions! The ratio of the mean duration to the median duration is $\Gamma(1+(1/\alpha)) / [\ln(2)]^{(1/\alpha)}$.

Use Stata to look at how the Weibull hazard varies with α and λ. First I give you some sample code and then show the graphs which Stata produces. The code is a little more fancy than the minimum required to do the exercise, but should give you an idea of the capabilities for labelling output.

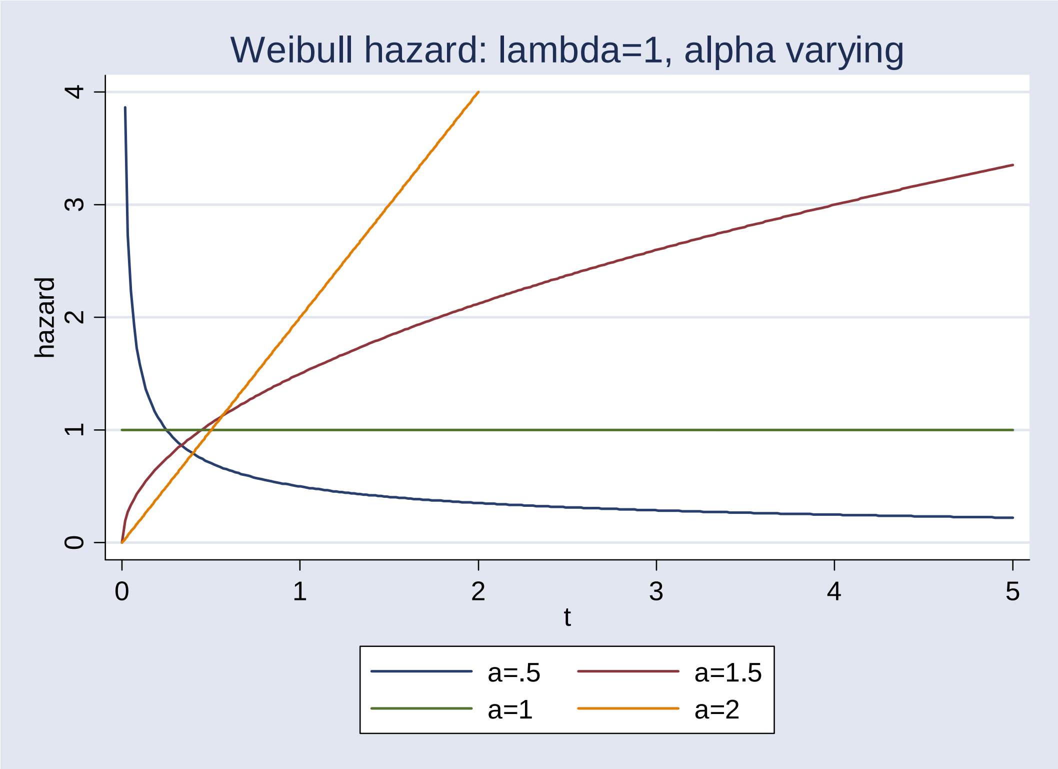
Use the output to remind yourself how the hazard and survivor function shapes change as α is varied for given λ. What would the hazard function look like for α > 2? Note too how the functions vary with changes in λ for given α. Of the values of α and λ used in the example, which combination would give the longest and the shortest durations? For each set of parameter values, verify that the mean duration is always larger than the median duration.

To examine the shapes of the curve, I use the fact that **graph twoway** can plot a function *y* = *f*(*x*), without any data for *y* or *x* having to be in memory: you just provide the functional form (and e.g. the range over which *x* lies). One simple example of the command being used to display the shape of the Weibull hazard rate is as follows.

```
* Weibull hazard: lambda = 1, alpha = .5
twoway  function y = .5*x^(-.5), range(0 5)
```

A fancier version of the command which overlays several curves on one graph, and adds labelling and titles is the following. This shows different Weibull hazard rates for λ = 1, and different values for shape parameter α.

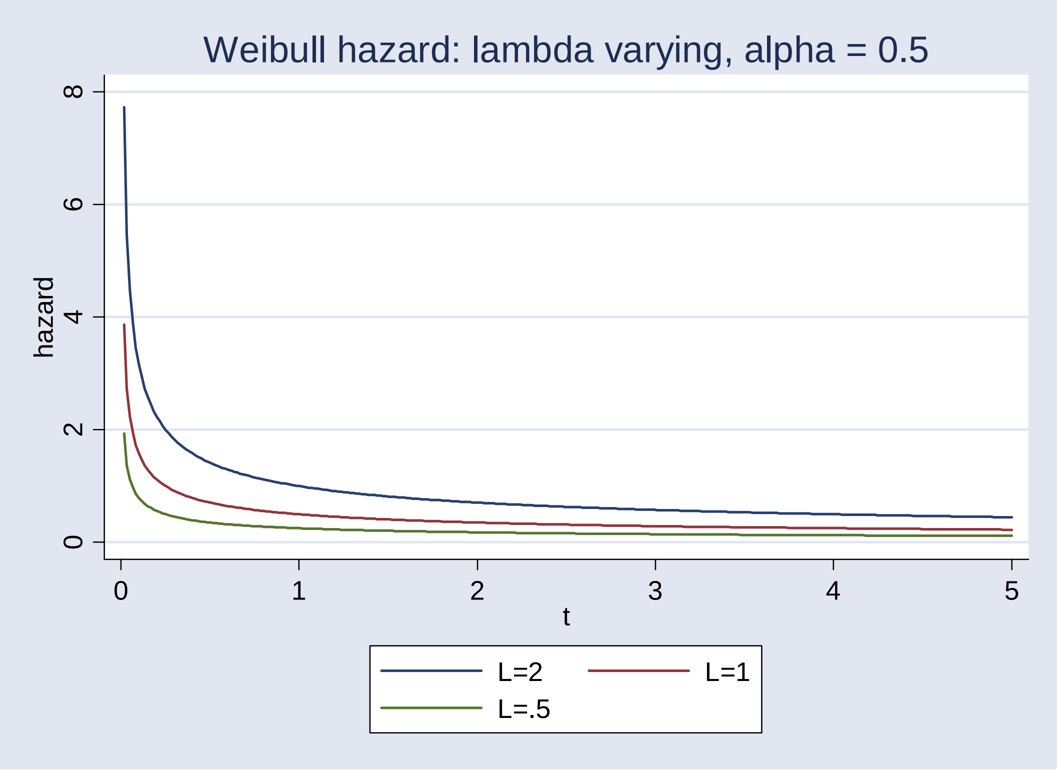
```
twoway  (function y = .5*x^(-.5), range(0 5) yvarlab("a=.5") )   ///
       ( function y = 1.5*x^(.5), range(0 5) yvarlab("a=1.5") ) ///
       ( function y = 1*x^(0), range(0 5) yvarlab("a=1") )  ///
       ( function y = 2*x, range(0 2) yvarlab("a=2") )                     ///
       , saving(weib1, replace)                             ///
       title("Weibull hazard: lambda=1, alpha varying")     ///
       ytitle(hazard) xtitle(t)
```



The graphs forming each part of the overlay are referred to in the statements enclosed by (.), e.g. `( function y = x^(-.5), range(0 5) yvarlab("L=2") ).` (The ‘///’ marker means the command line continues to the next line.) The additional options, after the ‘,’ in command line 4 save the graph to a file, and provide titles and labels. (Many other formatting possibilities are available: see the Stata Graphics manual.)

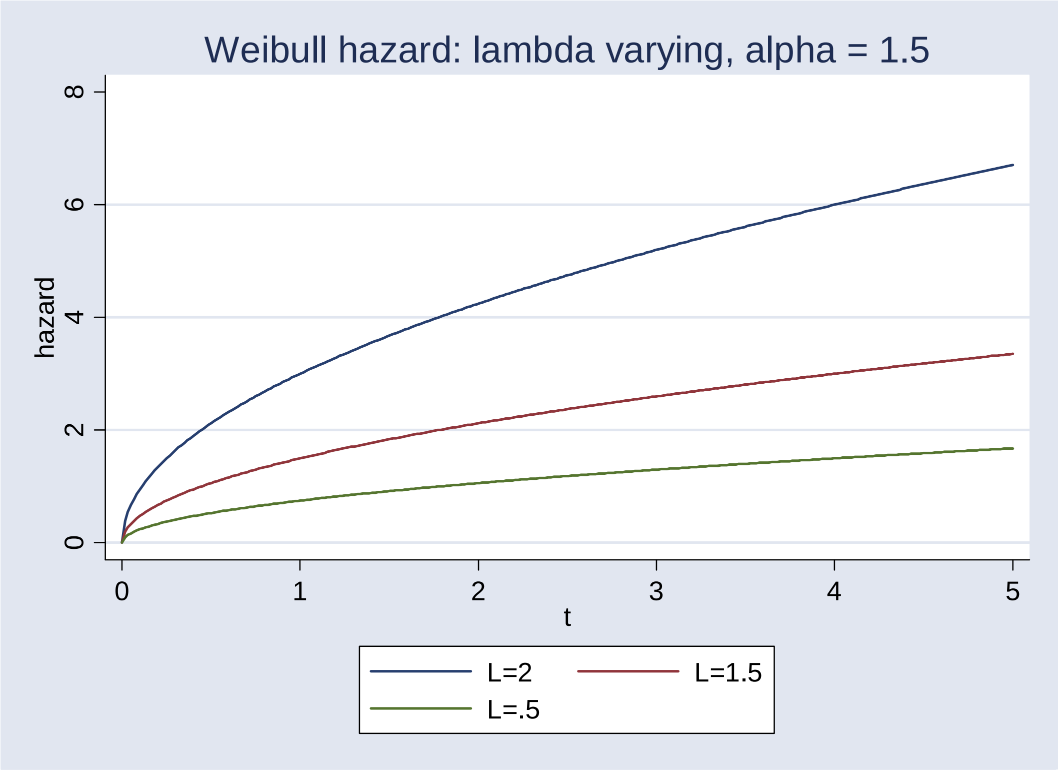
Now let’s look at the Weibull hazard keeping α fixed at 0.5, but varying λ:

```
twoway  ( function y = x^(-.5), range(0 5) yvarlab("L=2") )  ///
       ( function y = .5*x^(-.5), range(0 5) yvarlab("L=1") ) ///
       ( function y = 0.25*x^(-.5), range(0 5) yvarlab("L=.5") )   ///
       , saving(weib2, replace)                            ///
       title("Weibull hazard: lambda varying, alpha = 0.5") ///
       ytitle(hazard) xtitle(t)
```



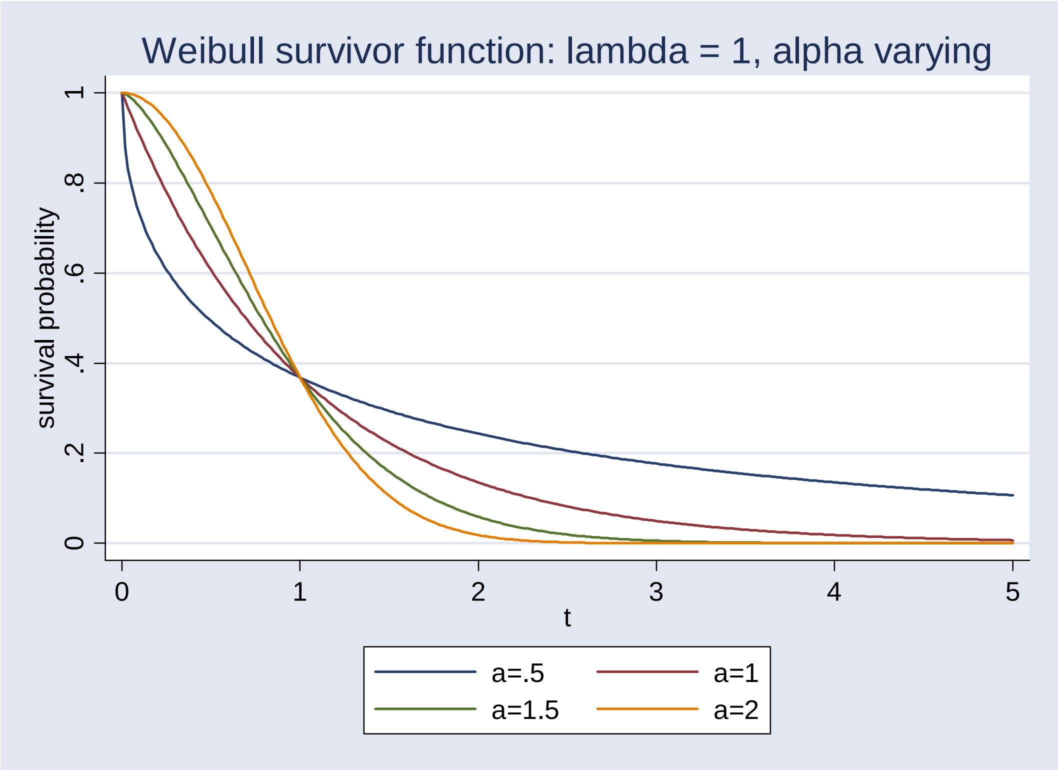
Here's the graph for the Weibull hazard keeping α fixed at 1.5, but varying λ:

```
twoway  ( function y = 2*1.5*x^(.5), range(0 5) yvarlab("L=2") ) ///
       ( function y = 1.5*x^(1.5), range(0 5) yvarlab("L=1.5") )  ///
       ( function y = .5*1.5*x^(.5), range(0 5) yvarlab("L=.5")) ///
       , saving(weib3, replace)                             ///
       title("Weibull hazard: lambda varying, alpha = 1.5") ///
       ytitle(hazard) xtitle(t)
```



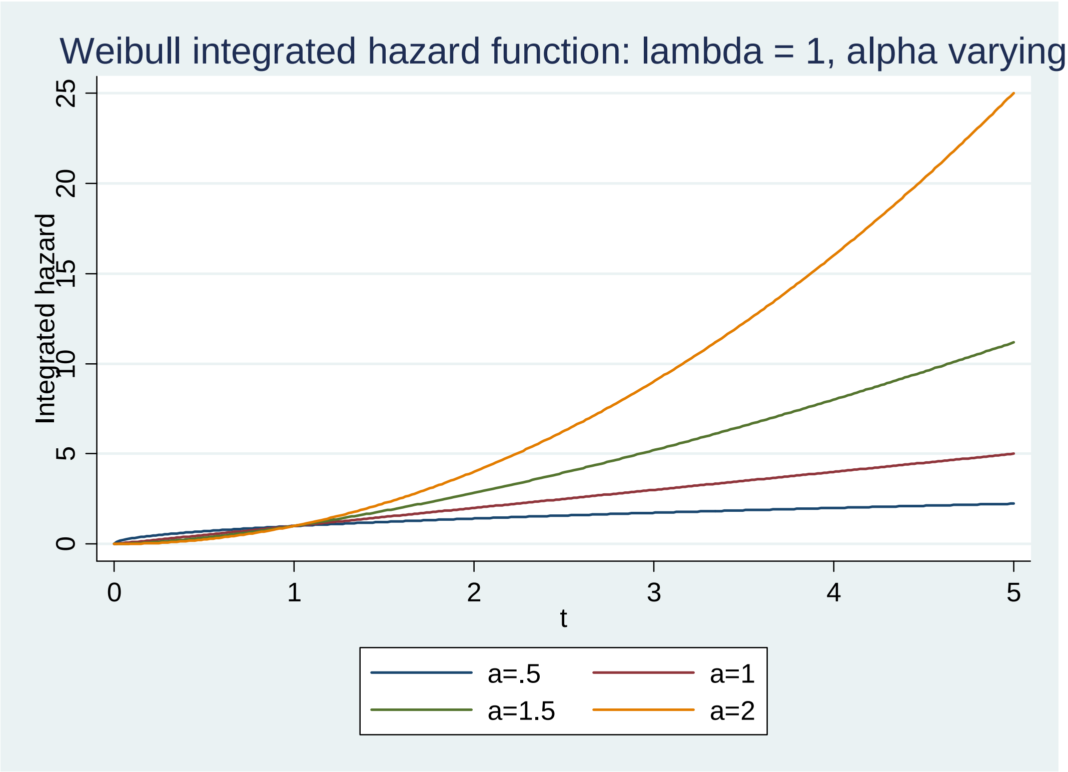
Now let’s consider the graphs for the Weibull survivor function.

```
twoway  ( function y = exp(-x^.5), range(0 5) yvarlab("a=.5") )             ///
        ( function y = exp(-x), range(0 5) yvarlab("a=1") )                 ///
        ( function y = exp(-x^1.5), range(0 5) yvarlab("a=1.5") )           ///
        ( function y = exp(-x^2), range(0 5) yvarlab("a=2") )               ///
        , saving(weib4, replace)                            ///
        title("Weibull survivor function: lambda = 1, alpha varying")       ///
        ytitle(survival probability) xtitle(t)
```



Now, here's the integrated hazard function for the Weibull model, derive fixing λ = 1, but with different values for shape parameter α.

```
twoway  ( function y = x^.5, range(0 5) yvarlab("a=.5") )           ///
       ( function y = x, range(0 5) yvarlab("a=1") )                       ///
       ( function y = x^1.5, range(0 5) yvarlab("a=1.5") )          ///
       ( function y = x^2, range(0 5) yvarlab("a=2") )                     ///
       , saving(weib5, replace)                             ///
       title("Weibull integrated hazard function: lambda = 1, alpha varying")     ///
       ytitle("Integrated hazard") xtitle(t)
```



Remember the integrated hazard function cumulates hazard values as survival time increases. Since the hazard rate is always non-negative, then so too is the integrated hazard.

In the Exponential case, when the hazard rate is constant, the integrated hazard function increases linearly (the slope with respect to *t* is constant). If α >1, then the slope of the integrated hazard increases; if α < 1, then the slope is positive but not increasing with *t*. (Recall that the slope of the integrated hazard at a particular *t* is simply the hazard rate at that *t*.)

What would the curves look like if we allowed λ to vary, but fixed α at some particular value?

In Chapter 4, we consider how one might use as a test for the Weibull model, graphs against the log of time of the log of non-parametric estimates of the integrated hazard function for different subgroups of the population. To anticipate this, and to help motivate it, look at what happens if you repeat the code exactly as before except that you add `xscale(log) yscale(log)` to the options after `xtitle(t)`.

Mean and median durations

The output from the calculations of the median and mean durations below can be derived with **do ex2_1.do** and looking at the resulting log file. Check that answers from alternative various methods correspond as they should.

For derivations by *interpolation*, we simply list the values of survival time *t* and the survivor function *S*(*t*) in the neighbourhood of the median. To do this we first have to create an artificial data sets containing the values of *S*(*t*) that are implied at each *t*, with a different data set for each particular combination of λ and α.

```
set obs 100
ge t = (_n-1)/20
su t
compress
* Weibull survivor function: lambda=1, alpha varying
ge sw1 = exp(-t^.5)
lab var sw1 "lambda=1,alpha=.5"
ge sw2 = exp(-t^1.5)
lab var sw2 "lambda=1,alpha=1.5"
ge sw3 = exp(-t)
lab var sw3 "lambda=1,alpha=1"
ge sw4 = exp(-t^2)
lab var sw4 "lambda=1,alpha=2"
```

The variables sw1, sw2, and sw3 are the values of *S*(*t*) that are implied at each *t*. Now we can simply `list` the data and look for the median:

```
. * Median: derive by interpolation
. di "lambda=1,alpha=.5 (sw1)"
lambda=1,alpha=.5 (sw1)

. list t sw1 if abs(sw1-.5) < .1

     +----------------+
     |   t        sw1 |
     |----------------|
  7. |  .3   .5782652 |
  8. | .35   .5534366 |
  9. |  .4   .5312856 |
 10. | .45   .5112889 |
 11. |  .5   .4930687 |
     |----------------|
 12. | .55   .4763417 |
 13. |  .6   .4608896 |
 14. | .65   .4465402 |
 15. |  .7   .4331549 |
 16. | .75     .42062 |
     |----------------|
 17. |  .8   .4088417 |
     +----------------+
```

The ‘if abs(sw1-.5) < .1’ is the trick by which we list only values in the neighbourhood of the median defined here as those for which the absolute difference between the value of the simulated survivor function and the median value (0.5) is less than 10%. (abs(.) is the absolute value function.) Now repeat the exercise for the other simulated survivor functions:

```
. di "lambda=1,alpha=1.5 (sw2)"
lambda=1,alpha=1.5 (sw2)

. list t sw2 if abs(sw2-.5) < .1

     +-----------------+
     |   t         sw2 |
     |-----------------|
 14. | .65   .5921196 |
 15. |  .7   .5567372 |
 16. | .75   .5222969 |
 17. |  .8   .4889272 |
 18. | .85   .4567307 |
     |-----------------|
 19. |  .9   .4257875 |
     +-----------------+

. di "lambda=1,alpha=1 (sw3)"
lambda=1,alpha=1 (sw3)

. list t sw3 if abs(sw3-.5) < .1

     +-----------------+
     |   t         sw3 |
     |-----------------|
 12. | .55   .5769498 |
 13. |  .6   .5488116 |
 14. | .65   .5220458 |
 15. |  .7   .4965853 |
 16. | .75   .4723665 |
     |-----------------|
 17. |  .8    .449329 |
 18. | .85   .4274149 |
 19. |  .9   .4065697 |
     +-----------------+

. di "lambda=1,alpha=2 (sw4)"
lambda=1,alpha=2 (sw4)

. list t sw4 if abs(sw4-.5) < .1

     +-----------------+
     |   t         sw4 |
     |-----------------|
 16. | .75   .5697829 |
 17. |  .8   .5272924 |
 18. | .85   .4855369 |
 19. |  .9   .4448581 |
 20. | .95   .4055545 |
     +-----------------+
```

Now see what happens if we use the *exact formula* to derive the median. (You don’t have to replicate all these examples: just do enough to understand the principles involved.)

```
. *  Median: derive using exact formula
. di "median duration for lambda=1, alpha=.5 is "  (ln(2))^(1/.5)
median duration for lambda=1, alpha=.5 is .48045301

. di "median duration for lambda=1, alpha= 1 is "  (ln(2))^(1/1)
median duration for lambda=1, alpha= 1 is .69314718

. di "median duration for lambda=1, alpha= 1.5 is "  (ln(2))^(1/1.5)
median duration for lambda=1, alpha= 1.5 is .78321977

. di "median duration for lambda=1, alpha= 2 is "  (ln(2))^(1/2)
median duration for lambda=1, alpha= 2 is .83255461

. di "median duration for lambda=1, alpha= 4 is "  (ln(2))^(1/4)
median duration for lambda=1, alpha= 4 is .91244431

. di "median duration for lambda=.5, alpha=.5 is "  (2*ln(2))^(1/.5)
median duration for lambda=.5, alpha=.5 is 1.9218121

. di "median duration for lambda=.5, alpha= 1 is "  (2*ln(2))^(1/1)
median duration for lambda=.5, alpha= 1 is 1.3862944

. di "median duration for lambda=.5, alpha= 1.5 is "  (2*ln(2))^(1/1.5)
median duration for lambda=.5, alpha= 1.5 is 1.2432839

. di "median duration for lambda=.5, alpha= 2 is "  (2*ln(2))^(1/2)
median duration for lambda=.5, alpha= 2 is 1.17741

. di "median duration for lambda=.5, alpha= 4 is "  (2*ln(2))^(1/4)
median duration for lambda=.5, alpha= 4 is 1.0850853

.
. di "median duration for lambda=2, alpha=.5 is "  (.5*ln(2))^(1/.5)
median duration for lambda=2, alpha=.5 is .12011325

. di "median duration for lambda=2, alpha= 1 is "  (.5*ln(2))^(1/1)
median duration for lambda=2, alpha= 1 is .34657359

. di "median duration for lambda=2, alpha= 1.5 is "  (.5*ln(2))^(1/1.5)
median duration for lambda=2, alpha= 1.5 is .49339754

. di "median duration for lambda=2, alpha= 2 is "  (.5*ln(2))^(1/2)
median duration for lambda=2, alpha= 2 is .58870501

. di "median duration for lambda=2, alpha= 4 is "  (.5*ln(2))^(1/4)
median duration for lambda=2, alpha= 4 is .76727115
```

Here are the corresponding calculations for the *mean*:

```
. *  Mean: derive using formula
.
. di "mean duration for lambda=1, alpha=.5 is "  exp(lngamma(3))
mean duration for lambda=1, alpha=.5 is 2

. di "mean duration for lambda=1, alpha= 1 is "  exp(lngamma(2))
mean duration for lambda=1, alpha= 1 is 1

. di "mean duration for lambda=1, alpha= 1.5 is " exp(lngamma(1+(1/1.5)))
mean duration for lambda=1, alpha= 1.5 is .90274529

. di "mean duration for lambda=1, alpha= 2 is "  exp(lngamma(1.5))
mean duration for lambda=1, alpha= 2 is .88622693

. di "mean duration for lambda=1, alpha= 4 is "  exp(lngamma(1.25))
mean duration for lambda=1, alpha= 4 is .90640248

. di "mean duration for lambda=.5, alpha=.5 is " exp(lngamma(3))*(1/.5)^(1/.5)
mean duration for lambda=.5, alpha=.5 is 8

. di "mean duration for lambda=.5, alpha= 1 is "  exp(lngamma(2))*(1/.5)^(1/1)
mean duration for lambda=.5, alpha= 1 is 2

. di "mean duration for lambda=.5, alpha= 1.5 is "  exp(lngamma(1+(1/1.5)))*(1/
> .5)^(1/1.5)
mean duration for lambda=.5, alpha= 1.5 is 1.4330188

. di "mean duration for lambda=.5, alpha= 2 is "  exp(lngamma(1.5))*(1/.5)^(1/2
> )
mean duration for lambda=.5, alpha= 2 is 1.2533141

. di "mean duration for lambda=.5, alpha= 4 is "  exp(lngamma(1.25))*(1/.5)^(1/
> 4)
mean duration for lambda=.5, alpha= 4 is 1.0779003

. di "mean duration for lambda=2, alpha=.5 is "  exp(lngamma(3))*(1/2)^(1/.5)
mean duration for lambda=2, alpha=.5 is .5

. di "mean duration for lambda=2, alpha= 1 is "  exp(lngamma(2))*(1/2)^(1/1)
mean duration for lambda=2, alpha= 1 is .5

. di "mean duration for lambda=2, alpha= 1.5 is "  exp(lngamma(1+(1/1.5)))*(1/2
> )^(1/1.5)
mean duration for lambda=2, alpha= 1.5 is .5686939

. di "mean duration for lambda=2, alpha= 2 is "  exp(lngamma(1.5))*(1/2)^(1/2)
mean duration for lambda=2, alpha= 2 is .62665707

. di "mean duration for lambda=2, alpha= 4 is "  exp(lngamma(1.25))*(1/2)^(1/4)
mean duration for lambda=2, alpha= 4 is .76219059
```

2.2 Log-logistic model

Recall that for the log-logistic model

Hazard function	$h(t) = \{\psi^{(1/\gamma)} t^{[(1/\gamma)-1]}\} / \{ \gamma[1 + (\psi t)^{(1/\gamma)}] \}$
Survivor function	$S(t) = [1 + (\psi t)^{(1/\gamma)}]^{-1}$
Failure function	$F(t) = 1 - S(t)$
Density function	$f(t) = \{\psi^{(1/\gamma)} t^{[(1/\gamma)-1]}\} / \{ \gamma[1 + (\psi t)^{(1/\gamma)}] \}^2$
Integrated hazard function	$H(t) = -\ln S(t) = \log(1 + (\psi t)^{(1/\gamma)})$
	where $\psi = \exp(-\beta^{*\prime}X)$ and $\gamma > 0$.

Beware of alternative parameterisations of the same model in the literature.

As for the Weibull distribution, one can derive the median survival time in several ways:
(i) simply read the value of t from the graph of the survivor function at $S(t') = 0.5$.
(ii) print out combinations of t and $S(t)$ and interpolate using this.
(iii) derive the exact value using the closed form expression for the median implied by inverting the survivor function. Since t' satisfies $S(t') = 0.5$, then, from above:

$$t' = \psi^{-1}[(1/0.5) - 1]^{(1/\gamma)} = 1/\psi.$$

The mean survival time t^* only has a closed form when $\gamma < 1$. (Look at the graph for the hazard function for the case $\gamma \geq 1$ and consider why this is.) If $\gamma < 1$, then

$$t^* = \psi^{-1}(\gamma\pi)/\sin(\gamma\pi).$$

The ratio of the mean duration to the median duration is $(\gamma\pi)/\sin(\gamma\pi)$.

In Exercise 2.1, I ask you to repeat for the log-logistic model, the derivations for the Weibull model that were undertaken in the previous section.

3 Discrete time models

We work with the two leading cases, the logistic and the complementary log-log (cloglog) hazard functions. Recall that the cloglog model can be interpreted as the discrete time model corresponding to an underlying continuous time Proportional Hazards model. The logistic model is not a PH model, but if the hazard is sufficiently small then it approximates one quite closely. The logistic model can be interpreted as a proportional odds model however (see Lectures).

3.1 The logistic and complementary loglog models

Let $z(t) = c(t) + \beta'X$ for a representative person in month t, where $c(t)$ is the baseline hazard function and $\beta'X$ includes an intercept term. Observe that t now represents discrete time rather than continuous time and take integer values only.

Logistic discrete time hazard function: $p(t)$, where

$$\log[p(t)/(1–p(t))] = z(t)$$

$$\Rightarrow \quad p(t) \; = [1 + \exp(–z(t))]^{-1}$$

Complementary log-log (‘cloglog’) discrete time hazard function: $p(t)$, where

$$\log[–\log(1–p(t))] = z(t)$$

$$\Rightarrow \quad p(t) \; = 1 – \exp[–\exp(z(t))]$$

You can compare how the shapes of two discrete functions vary with z using the following code. The first part creates z and the two artificial variables containing logit(z) and cloglog(z):

```
clear
set obs 101
ge z = ((_n-1)-50)/10
ge logitp  = 1/(1 + exp(-z) )
ge clogp  = 1-exp(-exp(z) )
twoway connect logitp clogp z, yline(.5) xline(0)
drop z
```

In this artificial data set, time z varies from –5 to 5. The key Stata trick is the reference to ‘_n’, which is an integer built in to Stata equal to the value of the current observation number. We set the number of observations to 101 initially; the first observation (in row 1) has _n = 1, the second has _n = 2, and so on.

Observe the symmetry around 0 in the logit function that is not present for the cloglog function.

Let us now re-write the hazard functions in terms of the baseline hazard function $c(t)$ and $\lambda = \exp(\beta'X)$.

Logistic hazard function:

$$p(t) \; = [1 + (1/\lambda)\exp(–c(t))]^{-1}$$

Complementary log-log (‘cloglog’) hazard function:

$$p(t) \; = \; 1 – \exp[–\lambda\exp(c(t))]$$

The discrete time survival function $S(t)$

$$S(t) = [1–p(1)][1–p(2)][1–p(3)..[1–p(t)]$$

NB if $p(t)$ does not depend on t (constant hazard), i.e. $p(t) = p$, then

$$S(t) = (1–p)^t.$$

This is a Geometric duration distribution, which is the discrete time analogue of the Exponential distribution implied by the continuous time model with a duration-invariant hazard.

To calculate $S(t)$, we can use Stata's built-in functions. To do this, note that we can re-write $S(t)$ as

$$S(t) = \exp\{ \sum_{s=1}^{t} \ln[1–p(s)] \}.$$

First we can calculate ln[1–$p(s)$] for each observation. Then we can use the **sum(.)** function combined with **generate** to compute the cumulative sum, and then finally we can exponeniate the result. In fact, Stata's functions work in such a way that we can combine all these operations into one line of code, as shown below.

It is harder to derive closed form solutions for the mean duration and median duration in discrete time models compared to continuous time ones.

The median duration t' is defined implicitly by $S(t') = 0.5$.

The mean duration $t^* = \sum_{t=0}^{\max(t)} t.p(t)$.

Moreover we have not yet specified the shape of the baseline hazard function, $c(t)$.

Consider the case

$$c(t) = (q–1).\ln(t)$$

This is the discrete time analogue to the Weibull model (defined for $t > 0$). Look at the shape of the discrete hazard functions below to see why.

In this case, the logistic hazard simplifies to:

$$p(t) = [1 + \lambda^{–1}t^{1–q}))]^{–1}$$

and the cloglog hazard to:

$$p(t) = 1 – \exp[–\lambda t^{q–1}].$$

Since closed form expressions for the discrete time survivor function are not generally available, typically one interpolates from the estimated survivor function (looking at its graph, or at the underlying data).

Closed forms for the survivor functions are available, however, in the case when the hazard does not depend on time, i.e. $p(t) = p$, in particular when $q = 1$ in the case where the baseline hazard function $c(t) = (q–1)\ln(t)$. In this case we have

$$S(t) = (1–p)^t \Rightarrow t = \log[S(t)]/\log(1–p)$$

$$=> \quad t' = \log(0.5)/\log(1–p)$$

Hence, in the logistic hazard case,

$$t' = –\log(0.5)/\log(1+\lambda) = \log(2)/\log(1+\lambda)$$

and in the cloglog hazard case,

$$t' = –\log(0.5)/\lambda = \log(2)/\lambda.$$

Now let us look at the hazard and survival functions for specific values of q and λ.

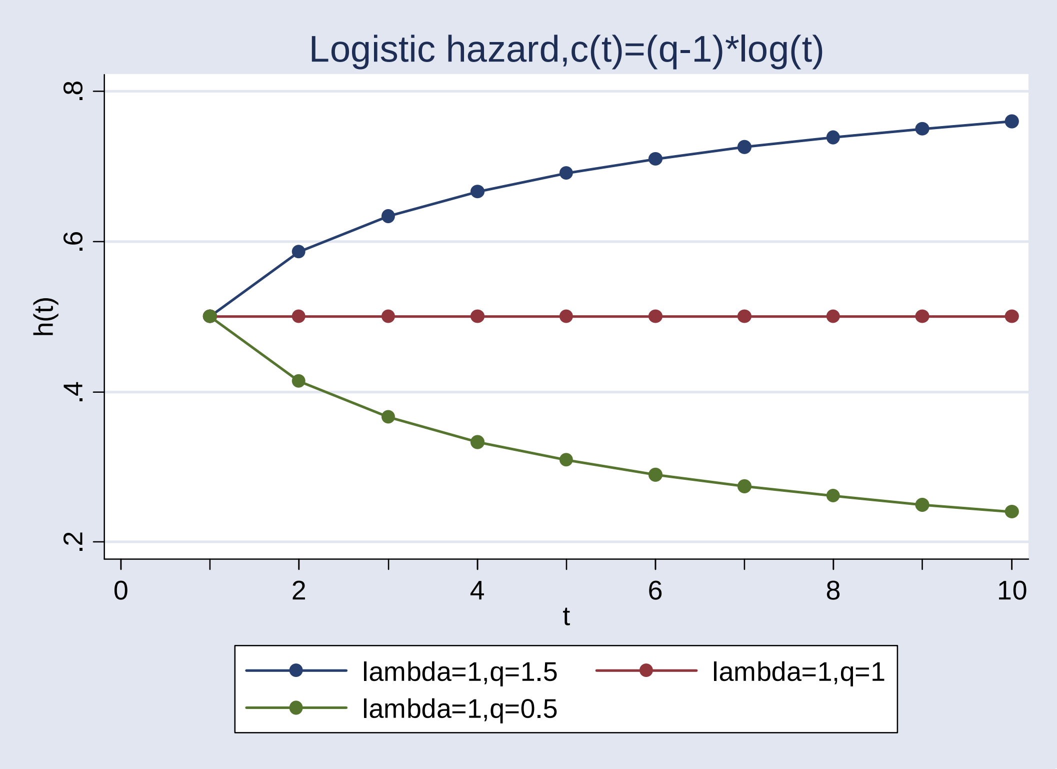
I will first work through the logistic hazard case assuming baseline hazard $c(t) = (q–1).\ln(t)$, and then later I shall ask you to repeat the analysis for the cloglog model, as an exercise.

I first create an artificial data set to represent ‘time’. Note that t now refers to intervals of time (‘months’) rather specific points in time. Index each month by integers 1,..., 10.

```
clear
set obs 10
ge t = _n
compress
```

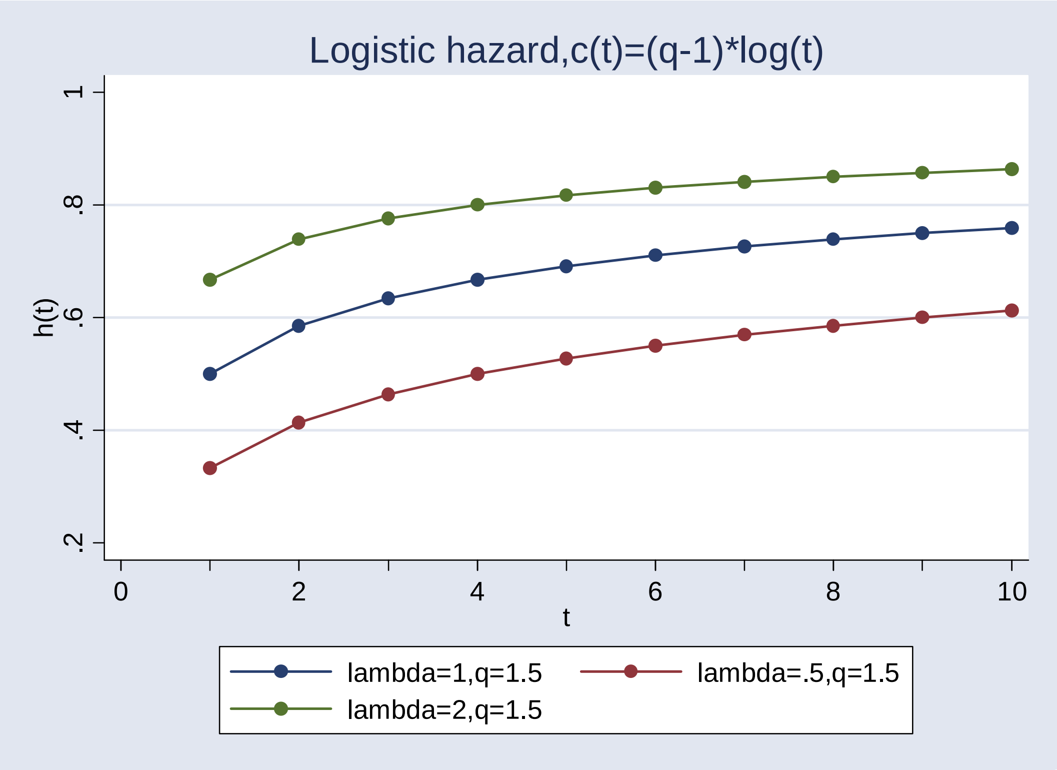
Now look at the hazard and survivor functions for the logistic hazard function, using the following code. One of the tricks used is to label the variables being graphed: these labels then get used in the graph legend of **twoway connect** automatically. The `xtick(0(1)10)` is what is used to ensure that there are ticks on the horizontal axis at each integer. Observe that the hazard is not defined at $t = 0$ if $c(t) = (q–1).\ln(t)$ because ln(0) is not defined. Hence no value is calculated for the survivor function.

```
ge hlog1 =  1/(1 + t^(-.5))
lab var hlog1 "lambda=1,q=1.5"
ge hlog2 =  1/2
lab var hlog2 "lambda=1,q=1"
ge hlog3 =  1/(1 + t^.5)
lab var hlog3 "lambda=1,q=0.5"
twoway connect hlog1 hlog2 hlog3 t ///
       , title("Logistic hazard,c(t)=(q-1)*log(t)")  ///
       ytitle("h(t)") xtick(0(1)10) saving(logist1,replace)
```



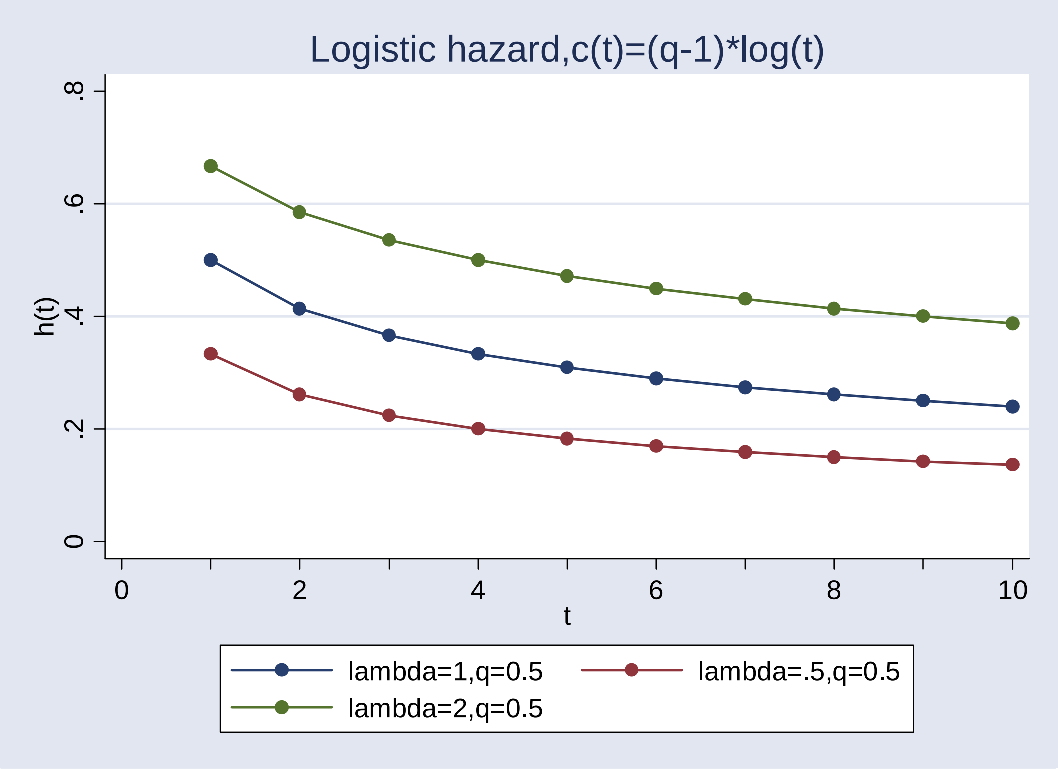
Now let’s try keeping q constant, but vary λ:

```
ge hlog4 =  1/(1 + 2*(t^(-.5)))
lab var hlog4 "lambda=.5,q=1.5"
ge hlog5 =  1/(1 + .5*(t^(-.5)))
lab var hlog5 "lambda=2,q=1.5"
twoway connect hlog1 hlog4 hlog5 t ///
       , title("Logistic hazard,c(t)=(q-1)*log(t)")  ///
       ytitle("h(t)") xtick(0(1)10) saving(logist2,replace)
```


Logistic hazard,c(t)=(q-1)*log(t)
h(t)
.2
.4
.6
.8
1
0
2
4
6
8
10
t
lambda=1,q=1.5
lambda=.5,q=1.5
lambda=2,q=1.5

Here's the same thing again, but now with q = 0.5.

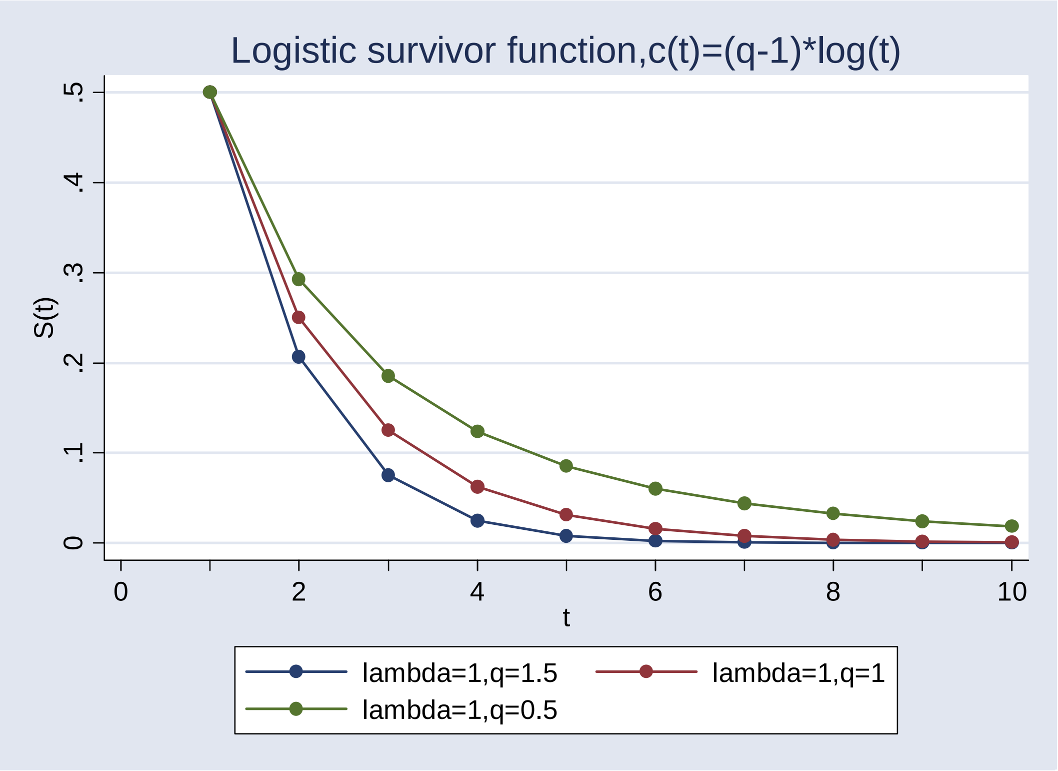
```
ge hlog6 =  1/(1 + 2*(t^(.5)))
lab var hlog6 "lambda=.5,q=0.5"
ge hlog7 =  1/(1 + .5*(t^(.5)))
lab var hlog7 "lambda=2,q=0.5"
twoway connect hlog3 hlog6 hlog7 t ///
       , title("Logistic hazard,c(t)=(q-1)*log(t)")  ///
       ytitle("h(t)") xtick(0(1)10) saving(logist3,replace)
```



Now here are some survivor functions. Notice the use of the cumulative sum function, **sum(.)**, combined with **generate**, as explained earlier.

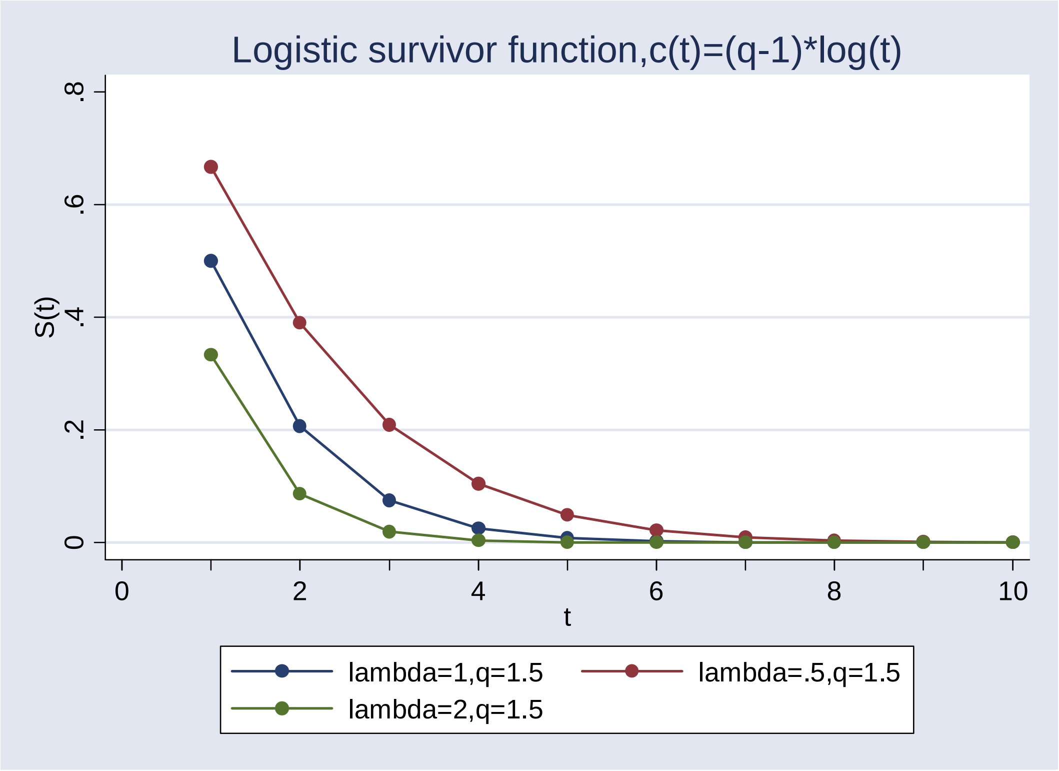
```
sort t
ge slog1 =  exp(sum(ln(1-hlog1)))
lab var slog1 "lambda=1,q=1.5"
ge slog2 =  exp(sum(ln(1-hlog2)))
lab var slog2 "lambda=1,q=1"
ge slog3 =  exp(sum(ln(1-hlog3)))
lab var slog3 "lambda=1,q=0.5"
ge slog4 =  exp(sum(ln(1-hlog4)))
lab var slog4 "lambda=.5,q=1.5"
ge slog5 =  exp(sum(ln(1-hlog5)))
lab var slog5 "lambda=2,q=1.5"

twoway connect slog1 slog2 slog3 t ///
       , title("Logistic survivor function,c(t)=(q-1)*log(t)")  ///
       ytitle("S(t)") xtick(0(1)10) saving(logist4,replace)
```



To draw the graph for the varying λ,

```
twoway connect slog1 slog4 slog5 t ///
       , title("Logistic survivor function,c(t)=(q-1)*log(t)")  ///
       ytitle("S(t)") xtick(0(1)10) saving(logist5,replace)
```



We can calculate the median durations in the special case q = 1:

```
. di "Median (logistic hazard, lambda=.5,q=1) = " ln(2)/ln(1.5)
Median (logistic hazard, lambda=.5,q=1) = 1.7095113

. di "Median (logistic hazard, lambda=1,q=1) = " ln(2)/ln(2)
Median (logistic hazard, lambda=1,q=1) = 1

. di "Median (logistic hazard, lambda=2,q=1) = " ln(2)/ln(3)
Median (logistic hazard, lambda=2,q=1) = .63092975
```

More generally we can try and derive the median using interpolation:

```
. di "lambda=1,q=1.5 (slog1 case)"
lambda=1,q=1.5 (slog1 case)

. list t slog1 if abs(slog1-.5) < .1

             t      slog1
  1.         1         .5

. di "lambda=1,q=1 (slog2 case)"
lambda=1,q=1 (slog2 case)

. list t slog2 if abs(slog2-.5) < .1

             t      slog2
  1.         1         .5

. di "lambda=1,q=0.5 (slog3 case)"
lambda=1,q=0.5 (slog3 case)

. list t slog3 if abs(slog3-.5) < .1

             t      slog3
  1.         1         .5

. di "lambda=0.5,q=1.5 (slog4 case)"
lambda=0.5,q=1.5 (slog4 case)

. list t slog4 if abs(slog4-.5) < .1

             t      slog4
```

This crude search method clearly does not work well in every case. Look at the survivor function graphs and you can see why! What would have happened if, instead, you had sought to derive the lower quartile, *l*, defined implicitly by $S(l) = 0.25$.

4 Exercise 2.1

1. Repeat the Weibull model derivations undertaken in Section 2 using instead the log-logistic hazard model, with parameter values $\psi = 1, 2$ and $\gamma = 0.5, 2$.
2. Use your results to comment on whether the mean survival time is always greater than the median survival time.
3. Repeat the discrete time logistic model derivations undertaken in Section 3 using instead the cloglog hazard function (and the same parameters). To what extent do the logistic and cloglog specifications provide similar or different results? What relationships do you see between the discrete time models with the baseline hazard specification $c(t) = (q–1).\ln(t)$, and the continuous time Weibull model?
4. Repeat some of the graphs of the hazard functions for the discrete-time logistic or cloglog models based on the log(time) duration dependence specification, but this time with a value of $q = 4$. Can you explain the shape of the hazard function in this case?
5. In this chapter, we have focused simulations on the hazard rate and survivor functions. You can use the same principles to graph the corresponding density and integrated hazard functions. The formulae are given in the Lecture Notes book.