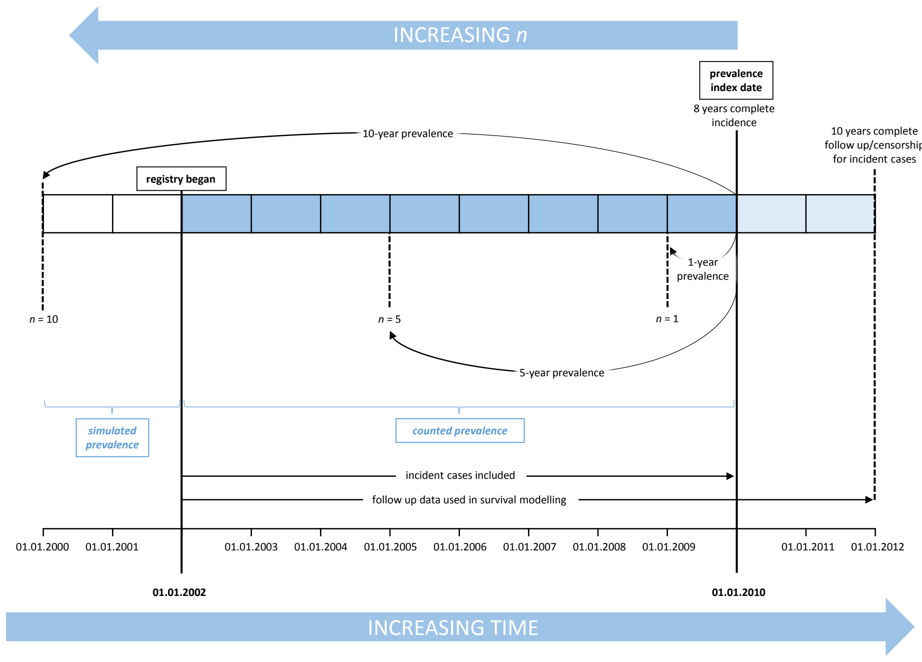
prevR Vignette (Short)

S J Lax

25 May 2016

Introduction

In this vignette we are going to demonstrate how predictions of prevalence are calculated from a synthetic patient dataset using `prevR`. We define the term **prevalence** as **"point estimates of disease prevalence at a specific index date"**. This is typically estimated from incidence and modelled survival probability using registry data. We calculate prevalence for n years, with n defined as the number of years prior to the index date for which we have included incident cases in the prevalent pool. In other words, prevalence estimates consider those diagnosed in the preceding n years. Therefore, the higher the value of n, the more incident cases are included. If n is greater than the number of registry years for which we have real patient data, then the contribution for the years preceding the registry back to n years is estimated using Monte Carlo simulation. This is illustrated in the diagram below:



As prevalence is estimated for the simulated years using probability of survival from the disease modelled on the available registry data, an appropriate value of n should be selected. If n is too low, the prevalence prediction will be an underestimate, as long-surviving cases may exist that were incident before n years preceding the index date. If n is too high relative to the length of the registry, the longer the simulation and the greater the reliance on extrapolation outside the registry data. If the disease of interest is rapidly fatal or the registry is sufficiently long that survival is accurately modelled on the available data, larger values of n might be appropriate. It is necessary to keep in mind that, as survival is presently modelled with an asymptotic Weibull distribution, some long-surviving cases may appear to be immortal. To overcome the problem of extrapolating the survival of these patients, we recommend implementing our so-called *cure* model, which we will discuss in more detail later in this vignette. In short, the higher the value of n, the more accurate the prevalence prediction, as long as survival can be adequately modelled over n years from the available registry data.

In the following sections we aim to take the user through the successive stages of the modelling process, outlining the correct use of the basic functions, illustrating how the diagnostics provided are used to check the validity of the assumptions required for simulating years of interest before the registry, and checking the agreement between observed and simulated prevalence estimates. First, we will inspect the consistency of incidence data between years of the registry and determine if it can be appropriate modelled as an homogeneous Poisson process. Second, we will look at the consistency of the survival data between years of the registry and check the adequacy of a parametric model of survival. We would emphasise that the user should check that their registry data does meet the required assumptions and if not, should understand that estimates of prevalence made using simulation based on that data may not be correct. Then, we will also introduce the population survival data and explain how it integrates with patient survival to produce a *cure* model, where it is desired. Lastly, we will introduce our functions for estimating prevalence, demonstrate how to check the resulting estimates are reasonable, and harvest the posterior distribution of a modelled covariate.

Basic setup and summary of data

```
library(prevR)
library(survival)
data(prevsim)
```

prevsim is a dataset that has been synthesised to resemble disease registry data. Incident cases are recorded from 2003-01-07 to 2013-01-30, and events occur between 2003-01-09 and 2015-03-17. It has 6 columns and is organised in a fashion typical to that found in real registry datasets. Patient data includes the date of both entry into the registry and last known event (be it death or a censored event time), survival time and an event indicator (*status*) along with both age and sex. The latter two are incorporated into the prevalence model at several levels, from being used as covariates in the survival modelling, and marking the incidence Poisson process. Currently `prevR` does not support the inclusion of additional variables in either the survival or incidence models, although this feature will be introduced in a later release.

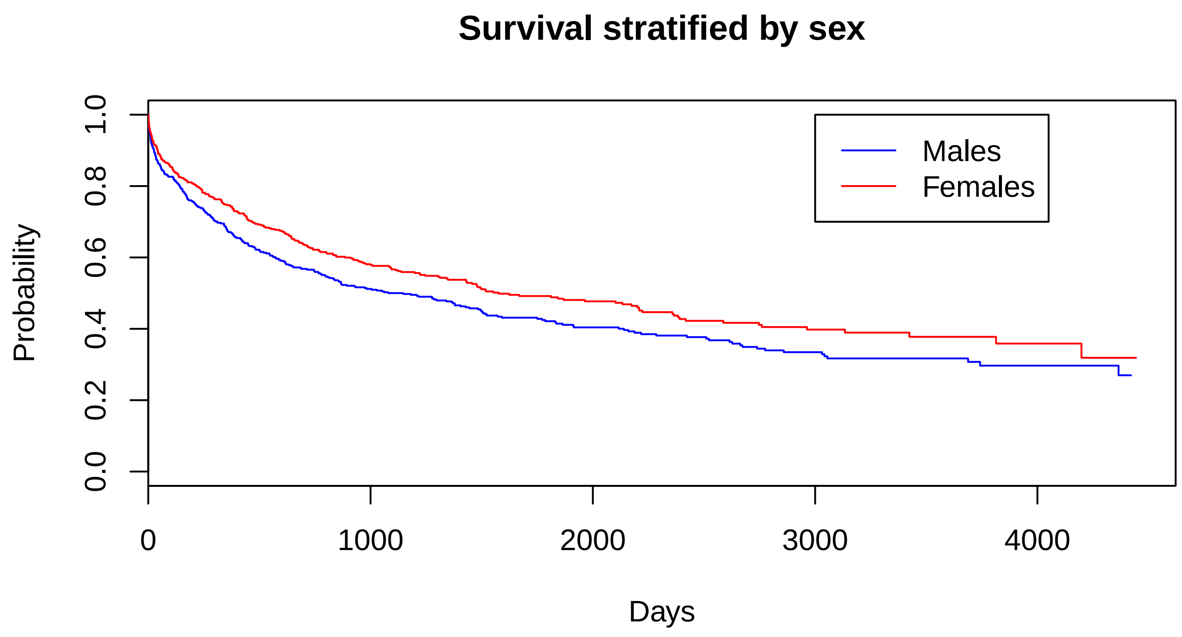
```
summary(prevsim)
```

```
##       time            status           age         sex
##  Min.   :   1.0   Min.   :0.000   Min.   :36.43   0:508
##  1st Qu.: 235.0   1st Qu.:0.000   1st Qu.:58.16   1:492
##  Median : 893.5   Median :1.000   Median :64.56
##  Mean   :1236.0   Mean   :0.558   Mean   :64.95
##  3rd Qu.:1868.2   3rd Qu.:1.000   3rd Qu.:71.88
##  Max.   :4444.0   Max.   :1.000   Max.   :95.32
##   entrydate          eventdate
##  Length:1000        Length:1000
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The following Kaplan-Meier plot shows the survival probabilities associated with the included dataset, *prevsim*, where, typical of many diseases, males have poorer survival outcomes than females. It also highlights that survival starts to level off after 2000 days. As our registry is short relative to the duration of survival from this disease, we do not have much information about long-term survival after this point in our patient population in order to base a model. In this scenario, we recommend using general population survival data to model long-term survivors, and we refer to this as a *cure model*. In our example, a patient is considered *cured* after five years survival with the disease.

Until the point in time that the cure model takes effect, survival is modelled on disease registry data; after, survival probability for the surviving cases is modelled using population mortality rates. `prevR` includes data from the Office of National Statistics (ONS) to describe UK mortality rates (in dataset `UKmortality`), however, the user may supply their own population mortality data. In addition to generating a better informed model, this approach also minimises the limitation of using a Weibull model, which tends towards predicting infinite survival times for long-term survivors, which will obviously not be correct.

```
survf <- survfit(Surv(time, status) ~ sex, data=prevsim)
plot(survf, lwd=1, col = c("blue","red"), main="Survival stratified by sex", xlab="Days",
     ylab="Probability")
legend(3000, 1, c("Males", "Females"), lty = 1, col=c("blue","red"))
```



Incidence

The simulation of incident cases in the years for which registry data is not available relies upon the assumption of a homogeneous Poisson process. The package includes several functions for inspecting the incidence data, as well as diagnosing the validity of the Poisson fit.

The `incidence` function calculates absolute disease incidence for each selected year of the registry; the optional arguments *start* and *num_reg_years* allow for the specification of certain years. These are useful when there are doubts about the accuracy of the incidence data, e.g. during the first year of operation of the registry. The output is a vector of *num_reg_years* length, detailing the number of incident cases in the registry at yearly intervals starting at `start` date. For example, in the following output, the incidence between 2004-01-30 and 2005-01-30 is 107.

```
raw_incidence <- incidence(prevsim$entrydate, start="2004-01-30", num_reg_years=9)
raw_incidence
```

```
## [1] 107  97  97  88  83 100 111 109 107
```

It can be concluded that the disease has around 100 new cases each year. To assess consistency of the incidence data with an homogeneous Poisson process, the `test_poisson_fit`, `inspect_incidence`, and `poisson_incidence_sim` functions can be used.

`test_poisson_fit` outputs the p-values for testing whether the process is over or underdispersed:

```
test_poisson_fit(raw_incidence)
```

```
## [1] 0.47945 0.52055
```

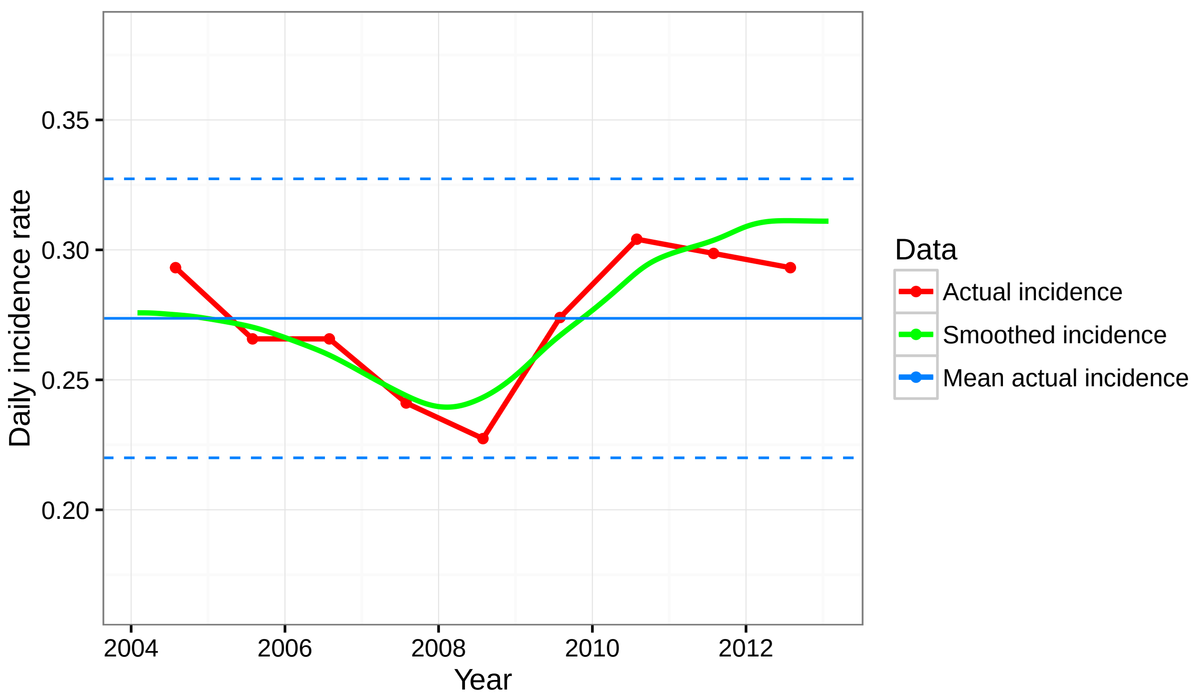
`inspect_incidence` and `poisson_incidence_sim` are both graphical means of validating the Poisson fit, and are methods of a `cincidence` object, for which the constructor is `cumulative_incidence`. This latter function calculates cumulative incidence for the registry data and fits a spline to it.

```
c_inc <- cumulative_incidence(prevsim$entrydate, start = "2004-01-30", num_reg_years = 9)
summary(c_inc)
```

```
## ~~~~~~~~~~~~~~
## Registry Data
## ~~~~~~~~~~~~~~
## Number of years: 9
## ~~~~~~~~~~~~~~
## Incidence
## ~~~~~~~~~~
## Known incidence by year: 107 97 97 88 83 100 111 109 107
## Diagnoses (time since registry began):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   820.5  1723.0  1674.0  2516.0  3279.0
## ~~~~~~~~~~~~~~
## Fitted smooth:
## ~~~~~~~~~~~~~~
## Call:
## smooth.spline(x = diags, y = seq(length(diags)), df = df)
##
## Smoothing Parameter  spar= 0.9987817  lambda= 0.02237235 (13 iterations)
## Equivalent Degrees of Freedom (Df): 5.99928
## Penalized Criterion: 11271.49
## GCV: 12.79195
```

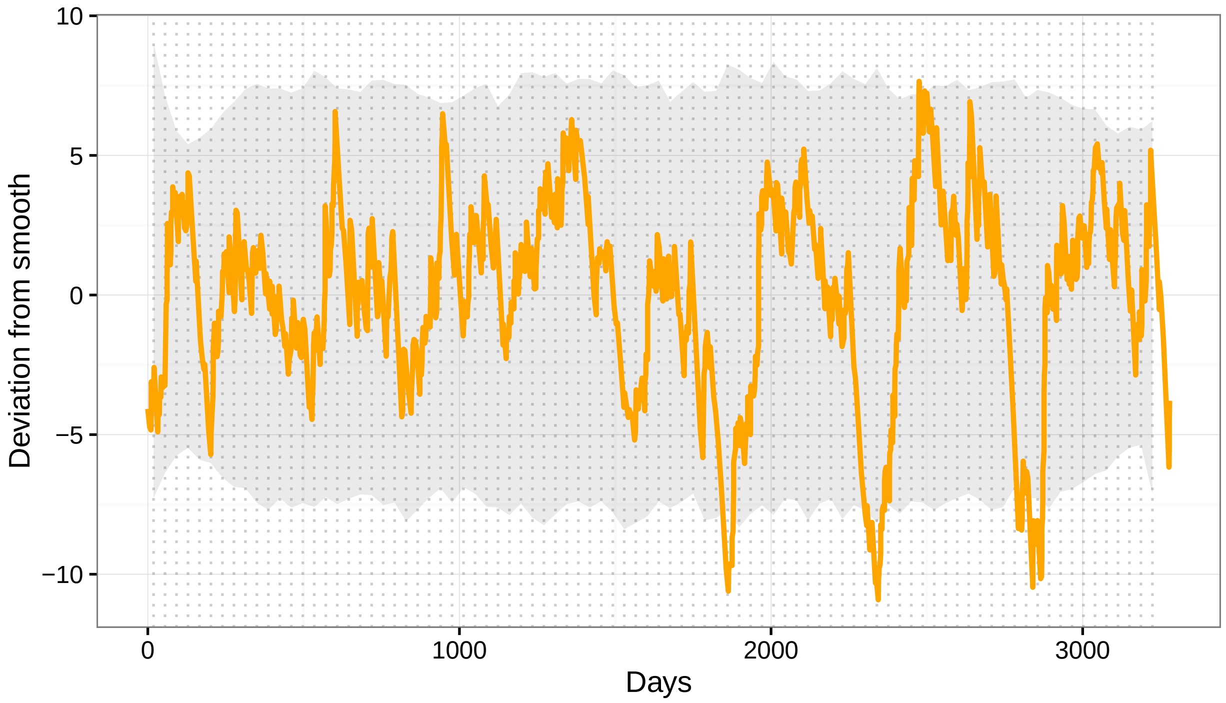
`inspect_incidence` displays a plot of the actual incidence rate as recorded in the registry, with the smooth overlaid. If incidence is an homogeneous Poisson process, both the smooth (green) and incidence process (red) should remain within the 95% confidence interval (dashed blue) and be evenly distributed about the mean (blue line). The actual incidence values should also be distributed in such a way as to resemble Brownian motion around the smooth. **Clarify this is correct with Simon**

```r
inspect_incidence(c_inc)
```



`poisson_incidence_sim` and `poisson_incidence_sim_gg` simulate a large number of incident case datasets, where the incident cases are randomly uniformly distributed between the start and end date of the original registry. Smooths are then fitted to these datasets. For both the smooth from the `cincidence` object and the simulated smooths, the deviations from the actual data are calculated. The deviation of the actual incidence from it's corresponding smooth is displayed in orange, with the confidence intervals derived for each bin from the simulations in grey. If the incidence process is consistent with an homogeneous Poisson process, the deviation from the smooth will be normally distributed and the actual data should fall within the 95% confidence interval. This plot assists the user to decide if the assumption is met:

```r
poisson_incidence_sim_gg(c_inc)
```

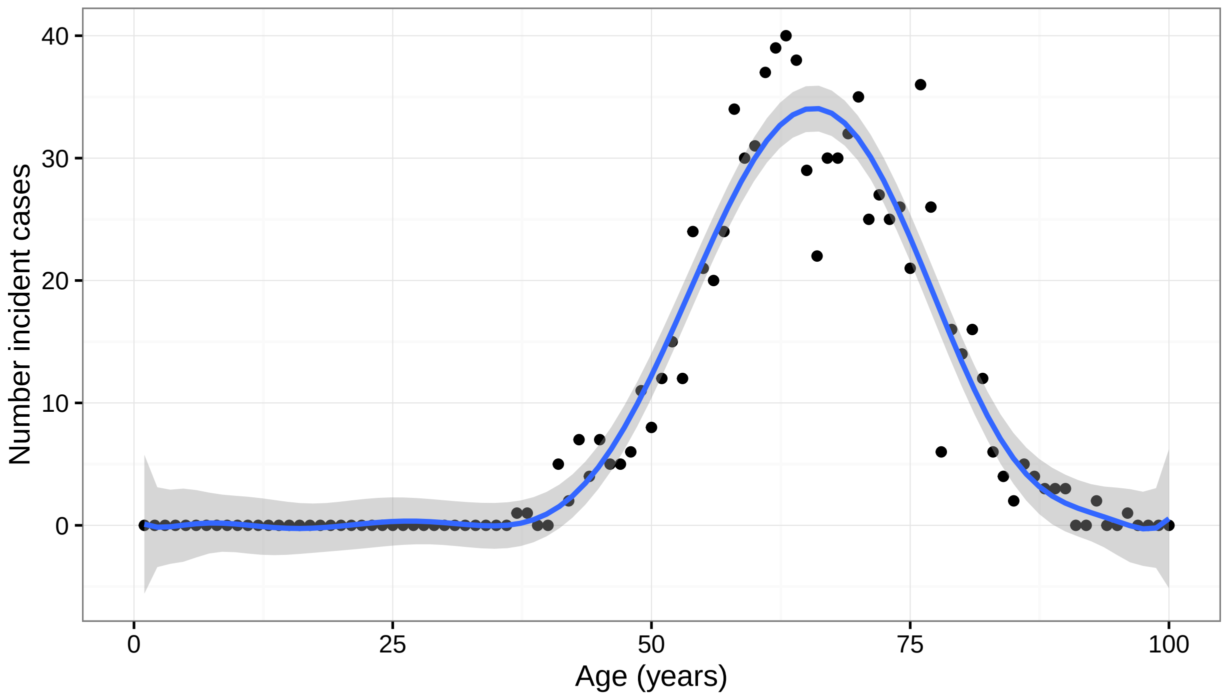


Currently, the incidence process can only be modelled as a Poisson homogeneous process; if the diagnostics indicate this is not a suitable fit then the prevalence estimates will be inaccurate. Future releases of `prevR` will allow for custom incidence distributions.

There are other functions to describe the incident cases, such as `incidence_age_distribution` and `mean_incidence_rate`. The former simply plots the distribution of incident cases by age:

```
prevsim_r <- prevsim[prevsim$entrydate >= "2004-01-30", ]

incidence_age_distribution(prevsim_r$age)
```



To calculate the mean incidence rate per 100,000 within the study population `mean_incidence_rate` is used, which requires an estimate of the size of the population at risk. The function provides confidence intervals at the specified level (default is 95%):

```
mean_incidence_rate(prevsim$entrydate, population_size = 3.5e6,
                    start = "2004-01-30", num_reg_years = 9)
```

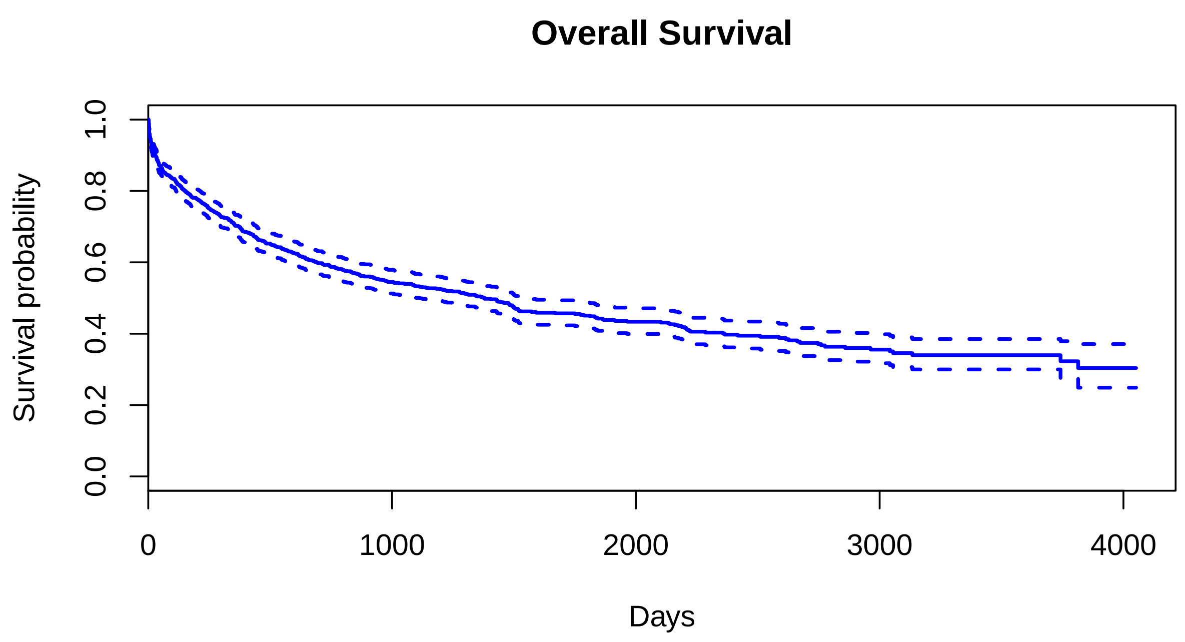
```
## $absolute
## [1] 99.89
##
## $per100K
## [1] 2.85
##
## $per100K.lower
## [1] 2.79
##
## $per100K.upper
## [1] 2.92
```

Survival modelling

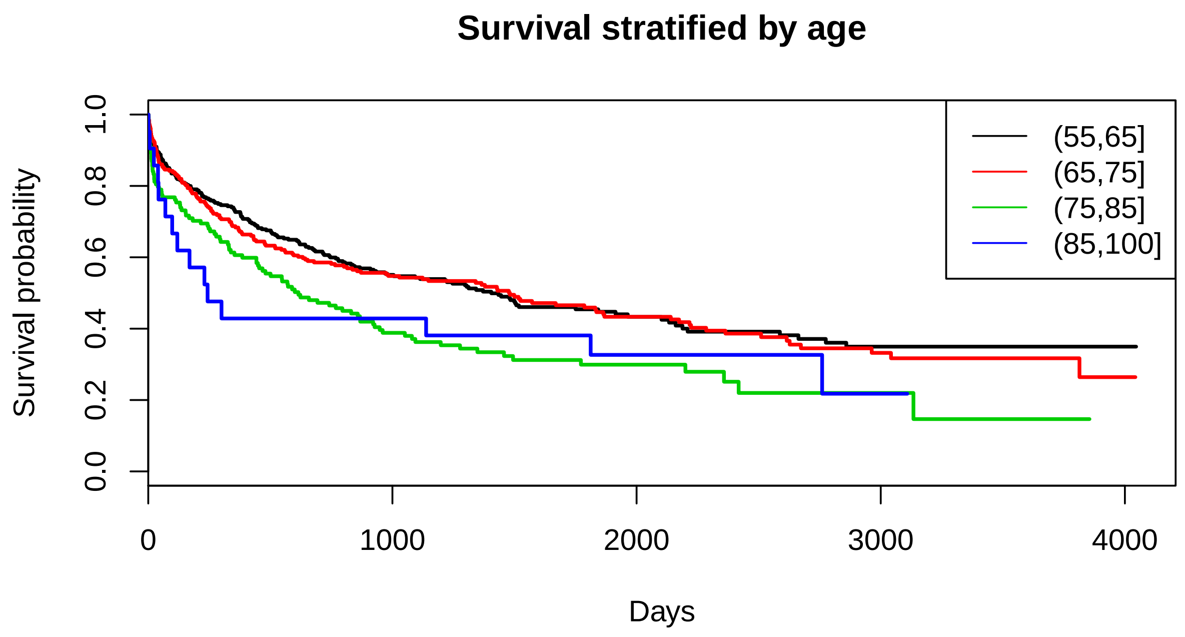
The second component to the prevalence estimation is the survival modelling process. It is recommended that the user inspects the survival data for consistency with a Cox Proportional Hazards model and between years of the registry as follows:

Firstly, it is always useful to plot the Kaplan-Meier estimator of the data, both as a whole and stratified by age to visually inspect for any inconsistencies:

```r
km <- survfit(Surv(time, status) ~ 1, data=prevsim_r)
plot(km, lwd=2, col="blue", main="Overall Survival", xlab="Days",
     ylab="Survival probability")
```

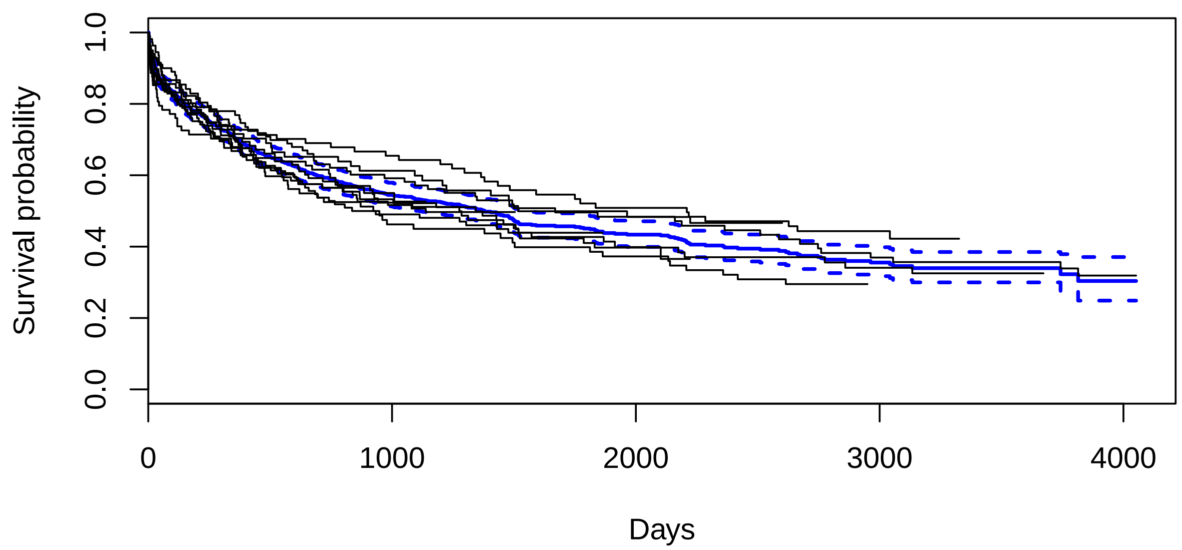


```r
ages = c(55, 65, 75, 85, 100)
km2 <- survfit(Surv(time, status) ~ cut(age, breaks=ages), data=prevsim_r)
plot(km2, lwd=2, col=1:length(ages), main="Survival stratified by age", xlab="Days",
     ylab="Survival probability")
legend("topright", legend=substring(names(km2$strata), 25, 32), lty = 1,
       col=1:length(ages))
```



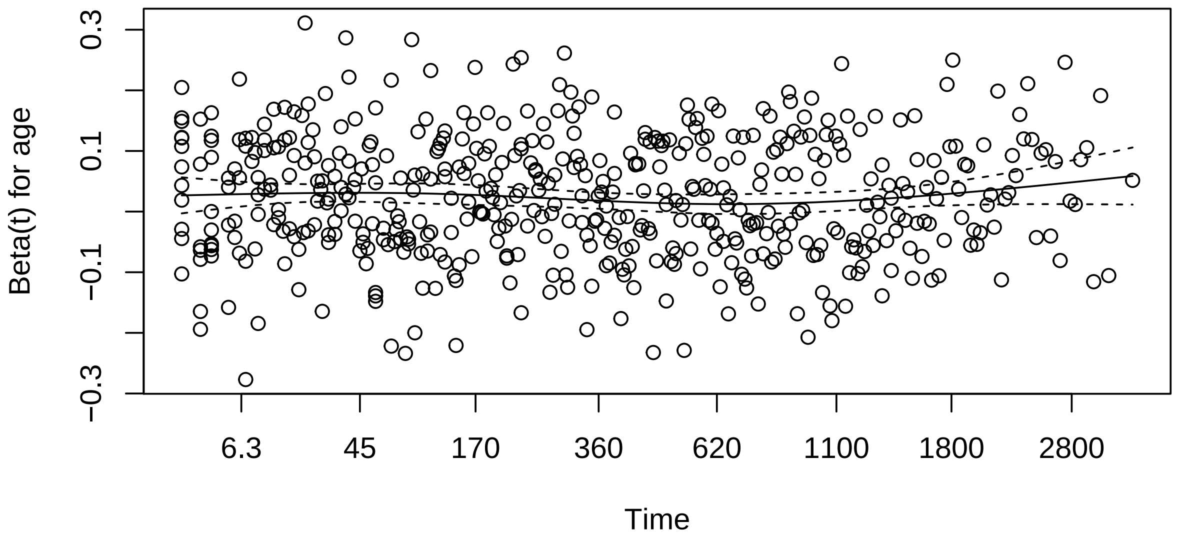
It is also a useful diagnostic aid to plot the survival curve for each year of the registry to determine whether there are any patterns:

```
plot(km, lwd=2, col="blue", mark.time=F, conf.int=T, xlab="Days",
     ylab="Survival probability")
num_reg_years <- 9
registry_years <- determine_registry_years(start='2004-01-30',
                                           num_reg_years=num_reg_years)
sapply(seq(num_reg_years),
       function(i) lines(survfit(Surv(time, status) ~ 1,
                                 data=prevsim_r[prevsim_r$entrydate >=
                                                  registry_years[i] &
                                                  prevsim_r$entrydate <
                                                  registry_years[i + 1], ]),
                         mark.time = F, conf.int = F))
```



The effect of age on hazard can be visualised to determine if there are any non-linear effects, which is easily done using the `cox.zph` function from the `survival` package. If the proportional hazards assumption holds, the Schoenfeld residuals should be approximately normally distributed around the beta(t) line, which should be horizontal: **Is this correct? Also, why is Cox model fit here, not Weibull since that's what we use?**

```
cx <- coxph(Surv(time, status) ~ age, data=prevsim_r)
cxp <- survfit(cx,
                newdata=data.frame(age=sapply(seq(length(ages) - 1),
                                              function(i) mean(c(ages[i], ages[i + 1])))))
plot(cox.zph(cx))
lines(cxp, lwd=2, col=1:length(ages), lty=2, mark.time=F)
```



It is also essential to test the proportional hazards assumption: **Does the Weibull AFT parameterisation in `survreg` actually need the PH assumption to be met?**

```r
cox.zph(cx)
```

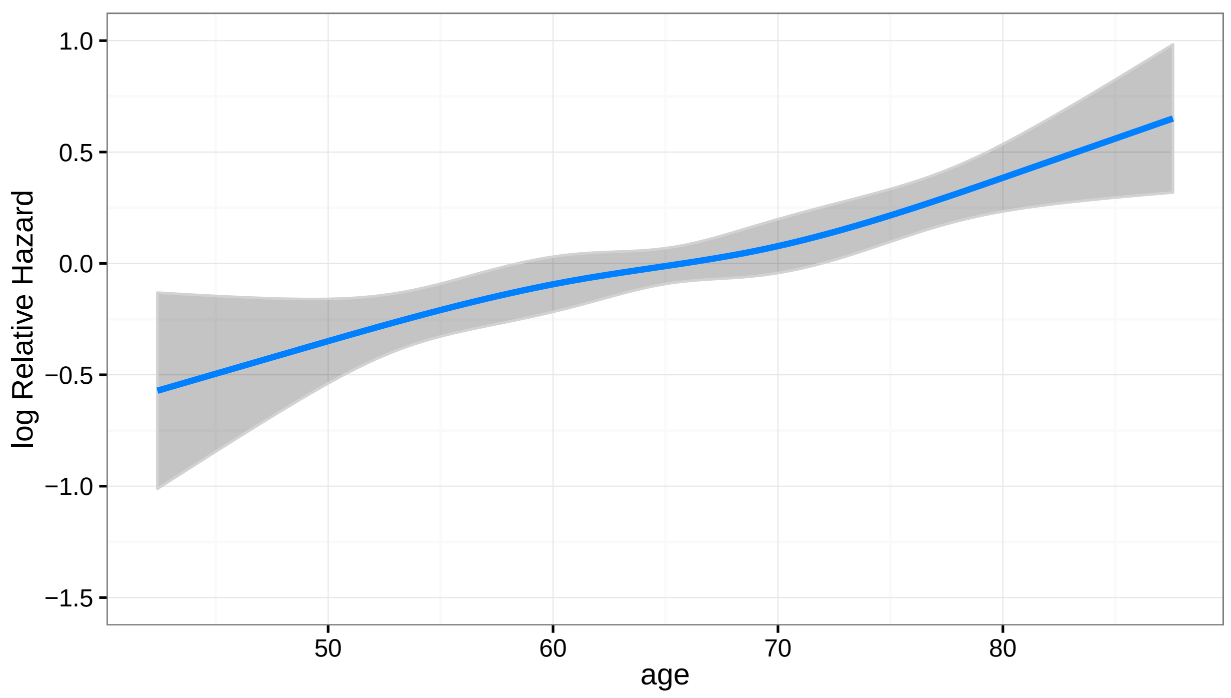
```
##          rho chisq    p
## age -0.0142 0.102 0.75
```

Functional form of age

`functional_form_age()` is provided so that the user can further investigate if modelling the effect of age as linear is appropriate. This function fits a Cox model to the data with a restricted cubic spline with the specified degrees of freedom. The model is returned by the function as an `rms::cph object`, allowing the user to inspect the fit. The function can also plot relative hazard as a function of age with the `plot_fit` argument.

Is this what is being plotted? Never used Predict function before. How is this different to the third plot in survival modelling above?

```r
functional_form_age(Surv(time, status) ~ age, prevsim_r, df=4, plot_fit=T)
```



```
##
## Cox Proportional Hazards Model
##
## rms::cph(formula = myform, data = mydf, x = TRUE, y = TRUE, surv = T,
##     time.inc = 1)
##                     Model Tests       Discrimination
##                                          Indexes
## Obs        900    LR chi2     29.24    R2       0.032
## Events     496    d.f.            3    Dxy      0.136
## Center 1.8153    Pr(> chi2) 0.0000    g        0.266
##                   Score chi2  30.92    gr       1.305
##                   Pr(> chi2) 0.0000
##
##       Coef    S.E.   Wald Z Pr(>|Z|)
## age    0.0293 0.0189  1.55  0.1208
## age'  -0.0252 0.0485 -0.52  0.6037
## age''  0.1165 0.1843  0.63  0.5275
```

Weibull model

Survival is modelled in the prevalence estimations using a Weibull distribution. Currently, there is no flexibility to how this model is fitted, it defaults to using both age and sex. It is therefore important to manually verify that this is an appropriate model.

```
wb <- survreg(Surv(time, status) ~ age + sex, data=prevsim_r)
summary(wb)
```

```
##
## Call:
```

```
## survreg(formula = Surv(time, status) ~ age + sex, data = prevsim_r)
##               Value Std. Error     z        p
## (Intercept) 10.7691    0.59009 18.25 2.07e-74
## age         -0.0479    0.00873 -5.49 4.04e-08
## sex1         0.5254    0.17214  3.05 2.27e-03
## Log(scale)   0.6422    0.03967 16.19 6.09e-59
##
## Scale= 1.9
##
## Weibull distribution
## Loglik(model)= -4087.9   Loglik(intercept only)= -4107.2
##  Chisq= 38.48 on 2 degrees of freedom, p= 4.4e-09
## Number of Newton-Raphson Iterations: 5
## n= 900
```

Population survival

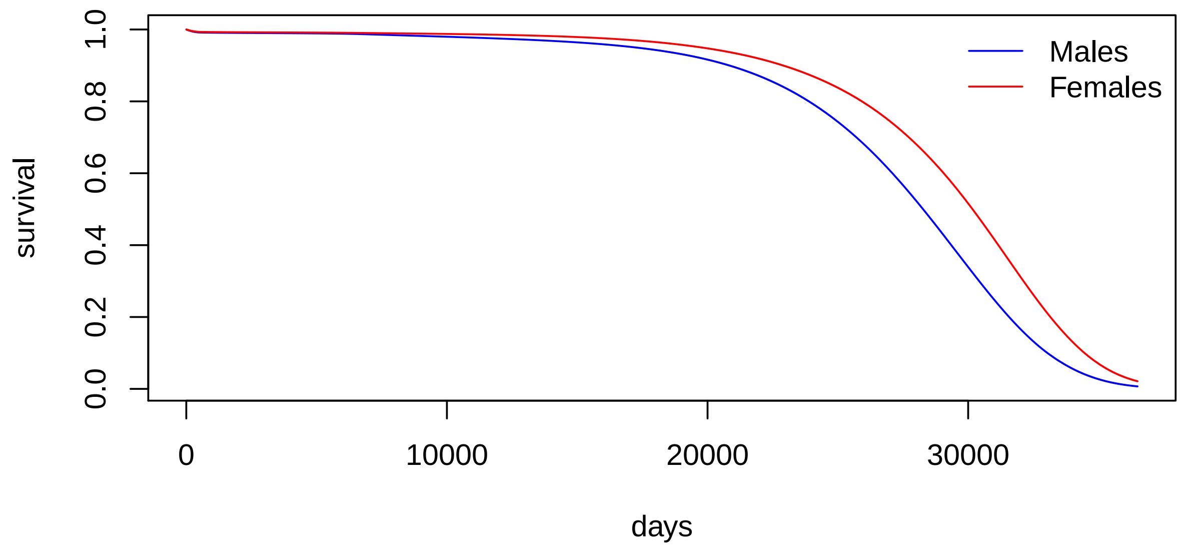
For modelling long-term survivors in registry data, we have implemented a *cure* model option. For example, the default `cure = 10` in the following functions for estimating prevalence, means a patient can be considered "cured" if they have survived until 10 years after diagnosis, and therefore from this point onwards their survival is modelled using population data rather than patient data.

General population period survival data for the UK is loaded from the supplied `UKmortality` dataset, and yearly rates are translated into daily rates by linear interpolation using `population_survival_rate()`.

```r
data(UKmortality)

daily_survival_males <- population_survival_rate(rate ~ age, subset(UKmortality,
                                                                    sex == 0))
daily_survival_females <- population_survival_rate(rate ~ age, subset(UKmortality,
                                                                      sex == 1))

plot(daily_survival_males, type="l", col="blue", xlab="days", ylab="survival")
lines(daily_survival_females, col="red")
legend("topright", legend = c("Males", "Females"),
               bty = "n", lty = 1, col = c(4,2))
```



Prevalence estimates

As a reminder, prevalence is estimated using incidence and survival data from n years, the larger n, the more accurate the estimate. However, registry data (and thus known incidence and survival) data may only be known for r years, where $r <= n$. If $r < n$, the remaining n-r years of incidence and survival are simulated using Monte Carlo techniques.

Counts

In the first instance, prevalence can be estimated by counting from the registry data. It is imperative that the number of incident cases is complete for estimating prevalence, and that follow-up data for all cases is complete at least until after the index date. In the case of the provided *prevsim* dataset there is a discrepency between the last recorded incident case and the last date of follow-up owing to a systematic difference in methods of incidence and survival data collection, as is typically the case in registry datasets where incidence and follow-up data are provided by different sources.

`prevalence_counted` calculates prevalence contributions for each year of the provided registry data, specified using the `start` and `num_reg_years` arguments. The index date, automatically calculated to be the last date of the last year of registry data included by the user, is 2013-01-30 in our example. In `prevalence_counted()`, all cases alive after the index date are censored.

NB: all follow-up data for the available incident cases is still used in survival modelling.

```
prevalence_counted(prevsim$entrydate,
                   prevsim$eventdate,
                   prevsim$status,
                   start="2004-01-30",
                   num_reg_years=9)
```

```
## [1] 44 41 50 35 44 52 57 66 85
```

The function returns the prevalence contributions by year, in ascending order (i.e there are 44 contributions from the year between 2004-01-30 and 2005-01-30).

Simulation

Prevalence is estimated using simulation by calling `prevalence()`. `num_years_to_estimate` corresponds to the required number of years preceding the index date that contribute incident cases. If any values are larger than `num_reg_years` then the remainder of years have their incidence and survival characteristics simulated. By passing a vector to `num_years_to_estimate`, multiple estimates of prevalence at the index date can be calculated with their own confidence intervals. The `population_size` variable is used to estimate prevalence per one hundred thousand (or at any rate specified by the `proportion` argument).

```r
prevalence_total <- prevalence(Surv(time, status) ~ sex(sex) + age(age) +
                                   entry(entrydate) + event(eventdate),
                               prevsim, num_years_to_estimate=c(3, 5, 10),
                               population_size=1e6, cure=5,
                               start='2004-01-30', num_reg_years=9)
```

Printing the `prevalence` object returned by the function of the same name displays the point estimate of prevalence at the index date using `num_years_to_estimate` years of data:

```r
prevalence_total
```

```
## Estimated prevalence per 1e+05 at 2013-01-30
## 3 years: 20.8
## 5 years: 30.4
## 10 years: 51.19
```

More detail from the `prevalence` object can be harvested using `summary`, including the p-value from a chi squared test of the difference between the predicted and counted prevalence for the available years of registry data:

```r
summary(prevalence_total)
```

```
## Registry Data
## ~~~~~~~~~~~~~
## Index date: 2013-01-30
## Start year: 2004-01-30
## Number of years: 9
## Known incidence rate:
## 107 97 97 88 83 100 111 109 107
## Counted prevalent cases:
## 44 41 50 35 44 52 57 66 85
##
## Bootstrapping
## ~~~~~~~~~~~~~
## Iterations: 1000
## Posterior age distribution summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
```

```
##      36      61      67      67      74     100 9492873
## Average simulated prevalent cases per year:
## 38 43 39 40 38 38 51 64 71 85
## P-value from chi-square test: 0.8007592
```

Calling the prevalence object's `estimates` attribute displays the point prevalence estimate along with rates per one hundred thousand and confidence intervals. As n increases, the prevalence estimate increases:

```
prevalence_total$estimates
```

```
## $y3
## $y3$absolute.prevalence
## [1] 208
##
## $y3$per100K
## [1] 20.8
##
## $y3$per100K.upper
## [1] 17.97
##
## $y3$per100K.lower
## [1] 23.63
##
##
## $y5
## $y5$absolute.prevalence
## [1] 304
##
## $y5$per100K
## [1] 30.4
##
## $y5$per100K.upper
## [1] 26.98
##
## $y5$per100K.lower
## [1] 33.82
##
##
## $y10
## $y10$absolute.prevalence
## [1] 511.93
##
## $y10$per100K
## [1] 51.19
##
## $y10$per100K.upper
## [1] 46.75
##
## $y10$per100K.lower
## [1] 55.63
```

Inspecting bootstrapped survival modelling

To inspect the distribution of the bootstrapped survival models, a `survfit.prev` object can be constructed using the usual `survfit` call, accepting a data frame of new data (with identical column names to those found in the original dataset). In the example below, survival probability is estimated for a 60 year old male:

```
prevsurv <- survfit(prevalence_total, newdata=list(age=60, sex=0))
prevsurv
```

```
## Survival probability calculated at 4444 timepoints, across 1000 bootstraps.
```

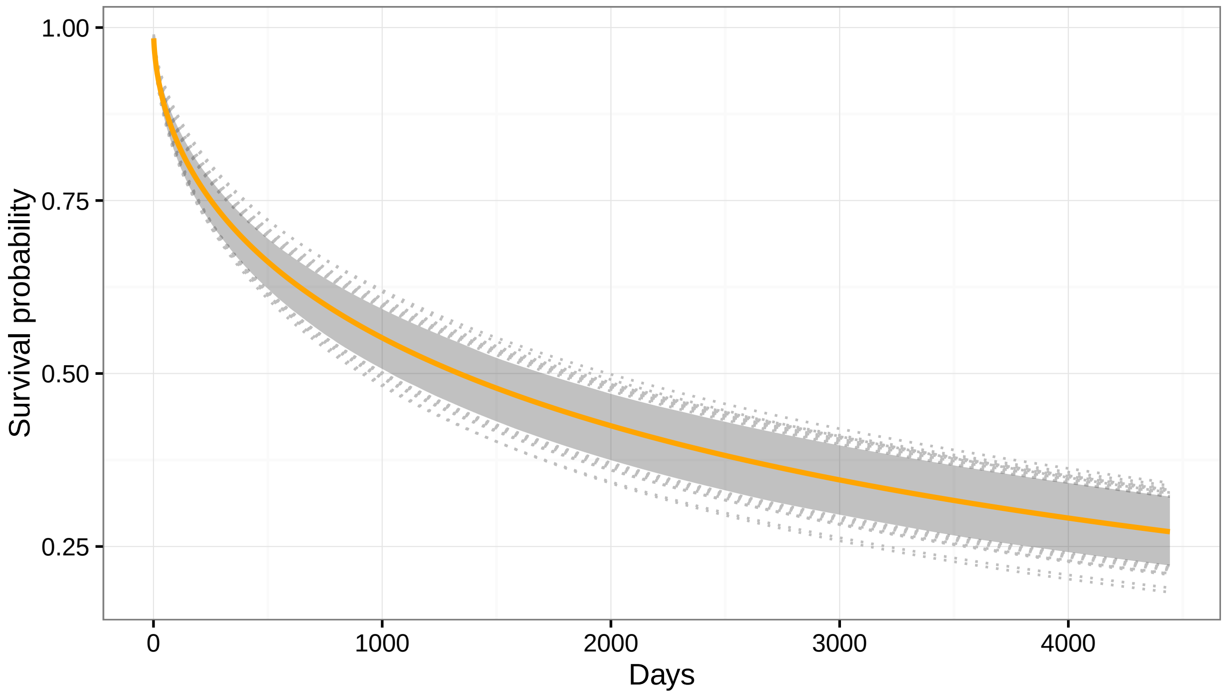
The `summary.survfit.prev` method provides N-year survival probabilities, with N specified as an argument vector:

```
summary(prevsurv, years=c(1, 3, 5, 10))
```

```
## Survival probability estimated using 1000 bootstrap survival curves:
## 1 year survival: 0.705 (0.668 - 0.736)
## 3 year survival: 0.535 (0.49 - 0.578)
## 5 year survival: 0.441 (0.393 - 0.488)
## 10 year survival: 0.308 (0.258 - 0.359)
```

Plotting the `survfit.prev` object displays the survival curve of a Weibull model using the original dataset (orange), along with 95% confidence intervals derived using the bootstrapped models shaded in light grey, while outlier curves are individually plotted. Outliers are defined as those survival curves where at least `pct_show` proportion of their point predictions lie outside the 95% confidence interval.

```
plot(prevsurv, pct_show=0.90)
```



Comparison between simulated and counted prevalence

The test whether the model is predicting reasonable values of prevalence, we can use the fact that we can directly measure the discrepancy between the predicted and actual prevalence for the available registry years. This difference can be formally tested using a chi-square test; the resulting p-value resulting is returned as an attribute of a `prevalence` object, called `pval`.

```
prevalence_total$pval
```

```
## [1] 0.8007592
```

For this model, there is no evidence to reject the null hypothesis.

This can also be calculated manually with the `test_prevalence_fit` function.

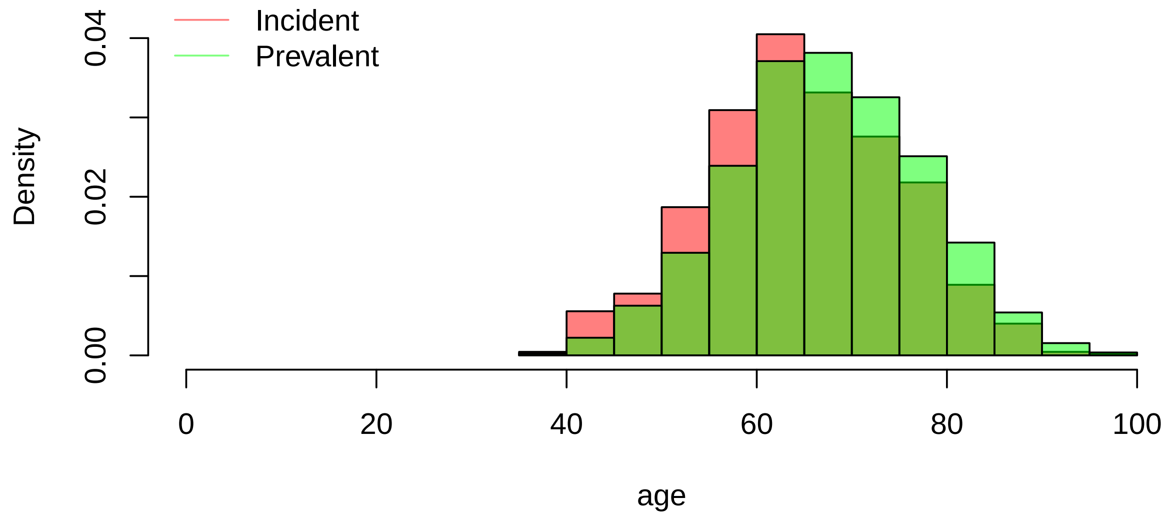
```
test_prevalence_fit(prevalence_total)
```

```
## [1] 0.8007592
```

Age distribution of prevalent cases

The age distribution of simulated prevalent cases can be viewed as a histogram and compared with the age distribution of incident cases:

```
hist(prevsim$age[prevsim$entrydate >= min(registry_years) &
                     prevsim$entrydate < max(registry_years)],
     col=rgb(1,0,0, alpha=0.5), xlim=c(0,100), ylim=c(0,0.045),
     main = "", xlab = "age", prob = TRUE)
hist(prevalence_total$simulated$posterior_age,
     col=rgb(0,1,0, alpha=0.5), prob = TRUE, add=T)
legend("topleft", legend = c("Incident", "Prevalent"), bty = "n", lty = 1,
       col = c(rgb(1,0,0, alpha=0.5),
               rgb(0,1,0, alpha=0.5)))
```


Incident
Prevalent
Density
0.00
0.02
0.04
0
20
40
60
80
100
age