SimRVPedigree

Christina Nieuwoudt, Jinko Graham

2016-10-10

Table of contents

Introduction

The `SimRVPedigree` package provides methods and functions to randomly simulate pedigrees likely to be asertained for multiple affected members under a particular study design. Pedigrees include individual specific variables such as birth year, and, when applicable, onset year, and/or death year. Additionally, the transmission of a rare variant, assumed to increase disease susceptibility, is simulated according to Mendel's laws.

In addition to randomly simulating pedigrees, `SimRVPedigree` also includes functions for: simulating waiting times associated with a non-homogeneous Poisson process, randomly simulating life events contingent on user-provided age-specific hazards, and functions to reassign the generation number of affected members (?? There's gotta be a better way to say this...).

Basic Usage

After installing the `SimRVPedigree` package, load it by typing the command:

```
library(SimRVPedigree)
```

Example Data

`SimRVPedigree` includes 2 example datasets:

1. `AgeSpecific_Hazards`: A dataset containing example age-specific hazards.
2. `exp_peds`: A dataset of five example pedigrees.

We begin the vignette by illustrating how the data in `AgeSpecific_Hazards` may be used to: simulate the waiting time to an event associated with a non-homogeneous Poisson process, randomly simulate all life events for an individual, starting at birth and terminating with death, and in conjunction with the `sim_RVpedigree` function to randomly simulate a pedigree with multiple affected members. We will use the `exp_peds` data to illustrate the effects of trimming pedigrees based on a proband's recall probability of a relative, and reassignment of affected generation number to reflect minimum carrier status. For more information on either of these datasets please type the command `help(AgeSpecific_Hazards)` and/or `help(exp_peds)` in the `R` console.

Simulating Waiting Times to Life Events

At the heart of the pedigree simulator is the simulation of individual life events starting with birth and ending with death. In simulation, we consider three competing age-specific life events: disease onset, reproduction, and death. We will now discusss the simulation of the waiting times to each of these life events in turn.

Simulating the Waiting Time to Disease Onset

First, consider the waiting time time to disease onset. Given an individual's current age t, we model the waiting time to disease onset, $w_{o|t}$, as the waiting time associated with a non-homogeneous Poisson process. The age-specific hazard associated with this process is determined by the individual's rare variant status, x_{RV}. In what follows we adopt the notation, $x_{RV} = 1$, if the individual has inherited the rare variant, and $x_{RV} = 0$, otherwise.

We model the age-specific hazard for disease onset according to the following Cox proportional hazards model:

$$\lambda_o(t|x_{RV}) = \begin{cases} \lambda_0(t), & \text{if } x_{RV} = 0; \\ \kappa * \lambda_0(t), & \text{if } x_{RV} = 1, \end{cases}$$

for $\kappa \geq 1$, where κ represents the relative risk for individuals who have inherited the rare variant.

We note that, in using this procedure to simulate the waiting time to disease onset, not all individuals will necessarily experience disease onset. However, we impose the restriction that those who do may only do so once. That is, if an individual has already experienced disease onset, we do not consider them for further onset events. Assuming that disease onset has not occurred by the individual's current age we use the `SimRVPedigree` package to simulate the waiting time to onset as follows:

```
#load example hazards
data("AgeSpecific_Hazards")
colnames(AgeSpecific_Hazards)
```

```
## [1] "pop_onset_hazard"      "pop_death_hazard"      "affected_death_hazard"
```

```
#the first column of the AgeSpecific_Hazards dataset provides the age-specific
#onset hazard for the population
my_onset_hazard = AgeSpecific_Hazards[,1]

#We must specify a partition of ages over which to apply the age-specifc
#hazards.  Note that a valid partition must contain 1 element more that the
#age-specific hazards. Assuming that the age-specific hazards are specified
#in 1 year increments starting at birth we can specify the partition of ages
#as follows.
age_part = seq(0, length(my_onset_hazard), by = 1)
```

```r
#Let's assume that the individual has inherited a rare variant which has an
#associated relative risk of disease onset of 10, and that the indiviual is
#currently 45 years old.  To simulate the waiting time to onset we would type
#the following command:
set.seed(1)
Time_to_onset <- get_WaitTime(p = runif(1), last_event = 45,
                 hazard = my_onset_hazard*10,
                 part = age_part)
Time_to_onset
```

```
## [1] 29.25
```

```r
#Note that in some instances, no waiting time to onset is simulated
Time_to_onset <- get_WaitTime(p = 0.99, last_event = 45,
                 hazard = my_onset_hazard*10,
                 part = age_part)
Time_to_onset
```

```
## [1] NA
```

Simulating the Waiting Time to Death

Simulating the waiting time to death given the current age t, $w_{d|t}$, is highly analogous, with a few caveats.

The first is that instead of considering an individual's rare variant status, we instead consider their disease status. If the individual for whom we are simulatind the waiting time to death has experienced disease onset, we use the affected age-specific mortality rate to simulate the waiting time to death, otherwise we use the unaffected age-specific mortaility rate.

The second caveat, is that unlike disease onset, which may or may not occur, all individuals must eventually die. However, since we cannot simulate death past the greatest age to which the age-specific hazards apply, we simulate the waiting time to death under the assumption that all indivdiuals will die by the maximum age specified. For example, suppose that the maximum age specifed is 100. Then, given an individual's current age t, we model the waiting time to death, $w_{d|t}$, as the waiting time associated with a non-homogeneous Poisson process conditioned on death occuring by age 100. The process of conditioning on the occurance of an event has an effect that is analogous to sampling from a truncated exponential distribution.

In simulation this is accomplished by setting `scale = TRUE` in the `get_WaitTime` function.

```r
#the second and third columns of the AgeSpecific_Hazards dataset, respectively,
#provide the age-specific death hazards for the unaffected and affected
#populations.  Note: if the disease of interest is sufficiently rare the
#population death hazard may be used as an estimate for the unaffected death
#hazard.
my_death_hazard <- AgeSpecific_Hazards[, c(2,3)]

#Note the effect of the different death hazards:
#First we simulate the waiting time to death for a 45 year old who has NOT
#experienced onset.
set.seed(123)
Time_to_death_unaffected <- get_WaitTime(p = runif(1), last_event = 45,
                                         hazard = my_death_hazard[,1],
                                         part = age_part, scale = TRUE)
Time_to_death_unaffected
```

```
## [1] 33.75
```

```
#Next, we simulate the waiting time to death for a 45 year old who HAS
#experienced disease onset.
set.seed(123)
Time_to_death_affected <- get_WaitTime(p = runif(1), last_event = 45,
                                       hazard = my_death_hazard[,2],
                                       part = age_part, scale = TRUE)
Time_to_death_affected
```

```
## [1] 24.25
```

Simulating the Waiting Time to Reproduction

To accommodate extra-Poisson variability observed in the number of offspring of humans, Kojima and Kelleher (1962) have proposed a negative binomial model with number of trials $n \approx 2$ and success probability $p \approx 4/7$. We adopt this negative binomial model of offspring number in `SimRVPedigree`.

We use an equivalent Poisson-Gamma mixture model (Zhou_2015) to obtain the negative-binomial offspring number and to simulate the waiting time to reproduction as follows.

Given an individuals current age t, let $w_{r|t}$ denote the conditional waiting time to reproduction. Next, assume that individuals are only able to reproduce from age a_1 to age a_2.

For each individual:

1. Draw $\gamma \sim Gamma(\alpha = 2, \beta = 4/3)$, their lifetime birthrate.
2. Simulate the unconditional waiting time to reproduction by drawing $w_r \sim Exp(\frac{\gamma}{a_2 - a_1})$.
3. Condition on their current age and generate $w_{r|t}$ as follows:

$$w_{r|t} = \begin{cases} a_1 + w_r & \text{if } t < a_1 \text{and } (a_1 + w_r) < a_2; \\ t + w_r & \text{if } t \in [a_1, a_2) \text{and } (t + w_r) < a_2; \\ \infty & \text{otherwise.} \end{cases}$$

Note that individuals with large simulated values of γ will have high birth rates and many children, whereas individuals with small simulated values of γ will have low birth rates and few or no children.

Note that with this simulation procedure we simulate the waiting time to birth according to a homogeneous Poisson process during an individual's reproductive years.

Simulating Life Events

Starting at birth, we simulate waiting times to each of the three events and choose as the winning life event the event with the shortest waiting time. We continue this process from the age of the previous winning event, until death occurs at which point the life events simulation process terminates. Again, we make note that the `SimRVPedigree` package only allows for disease onset to occur once. In the event that onset has occured, we simply choose between death and reproduction.

The `get_lifeEvents` function may be used to simulate all life events starting with birth and ending with death. The user must simply specify the individual's rare variant status, `RV_status`, the appropriate hazards for onset and death, `onset_hazard` and `death_hazard`, the year of birth, `YOB`, the age span for reproduction, `birth_range`, the parameters of the negative binomial distribution used to model the number of offspring per household, `NB_params`, and the relative risk of disease onset for individuals who have inherited the rare variant, `RR`. Note, all individuals with `RV_status = 0` are assumed to have `RR = 1`.

```r
# First, we illustrate how all life events are simulated for an individual who
# has inherited the rare variant
set.seed(2)
Life_Events <- get_lifeEvents(RV_status = 1,
                              onset_hazard = my_onset_hazard,
                              death_hazard = my_death_hazard,
                              part = age_part,
                              birth_range = c(18, 45),
                              NB_params = c(2, 4/7),
                              RR = 10, YOB = 1900)

Life_Events
```

```
## Start Birth Birth Onset Death
##  1900  1923  1930  1967  1985
```

```r
# Note the effect of keeping all factors constant except for rare variant #status.
set.seed(2)
Life_Events <- get_lifeEvents(RV_status = 0,
                              onset_hazard = my_onset_hazard,
                              death_hazard = my_death_hazard,
                              part = age_part,
                              birth_range = c(18, 45),
                              NB_params = c(2, 4/7),
                              RR = 10, YOB = 1900)

Life_Events
```

```
## Start Birth Birth Death
##  1900  1923  1930  1995
```

Simulating Pedigrees

Simulation of a full pedigree is accomplished as follows:

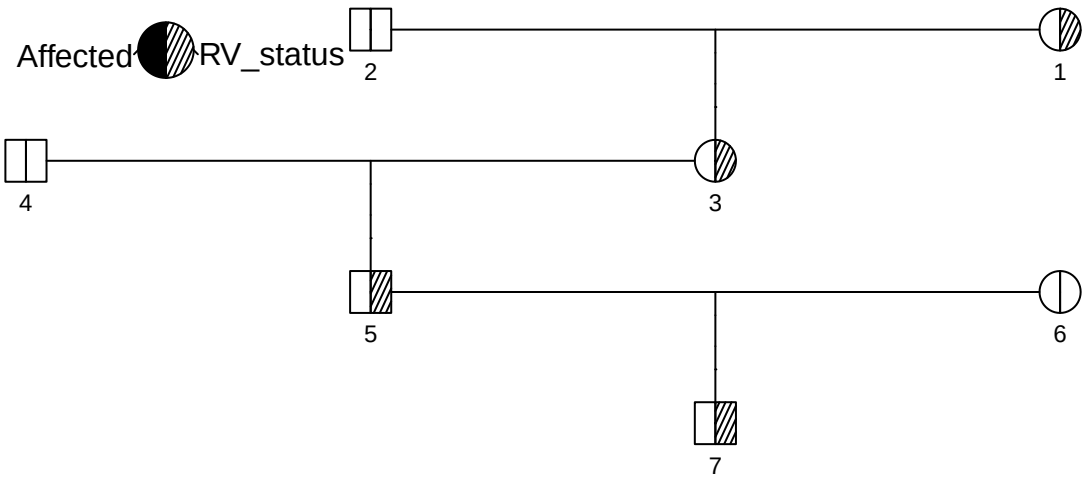
1. Simulate the founder's year of birth uniformly from the span of ages specified in `founder_byears`. We assume that this founder has introduced the rare variant into the pedigree.
2. Simulate all life events for the founder who has introduced the rare variant.
3. Transmit the rare variant to any of the founder's offspring according to Mendel's laws.
4. Simulate life events for all offspring recursively until the `stop_year` or death is reached. The `stop_year` argument represents the end of the study. All information that occurs after the stop year is censored.

Random simulation of a pedigree may be achieved using the `sim_ped` function as illustrated below.

```r
#Simulate a random pedigree
set.seed(6)
ex_ped <- sim_ped(onset_hazard = my_onset_hazard,
                  death_hazard = my_death_hazard,
                  part = age_part,
                  RR = 5, FamID = 1, stop_year = 2015,
                  founder_byears = c(1900, 1910))
```

```
# Pedigrees may be plotted usign the kinship2 package.  Assuming that this
# package has been installed it is loaded by executing the command:
library(kinship2)
```

```
# Define pedigree to use kinship2's plot function
 ex_pedigree <- pedigree(id = ex_ped$ID,
                         dadid = ex_ped$dad_id,
                         momid = ex_ped$mom_id,
                         sex = (ex_ped$gender + 1),
                         affected = cbind(Affected = ex_ped$affected,
                                          RV_status = ex_ped$DA1 +
                                                      ex_ped$DA2),
                         famid = ex_ped$FamID)['1']
 plot(ex_pedigree, cex = 0.75)
 pedigree.legend(ex_pedigree, location = "topleft",  radius = 0.25)
```



Is it critical to note that the `sim_ped` function does not simulate pedigrees with a mimimum number of affected individuals. Furthermore, if the pedigree founder does not produce any offspring `sim_ped` will return a ped file with a single individual. To ensure that a simulated pedigree contains a minimum number of affected individuals please use the `sim_RVpedigree` function.

Trimming Pedigrees

One of the features included in the the `SimRVPedigree` package is the option to trim pedigrees based on a proband's ability to recall relatives. This option is included to allow researchers to model the ascertainment bias that occurs when individuals either cannot provide a complete family history or they explicity request that particular family members not be contacted.

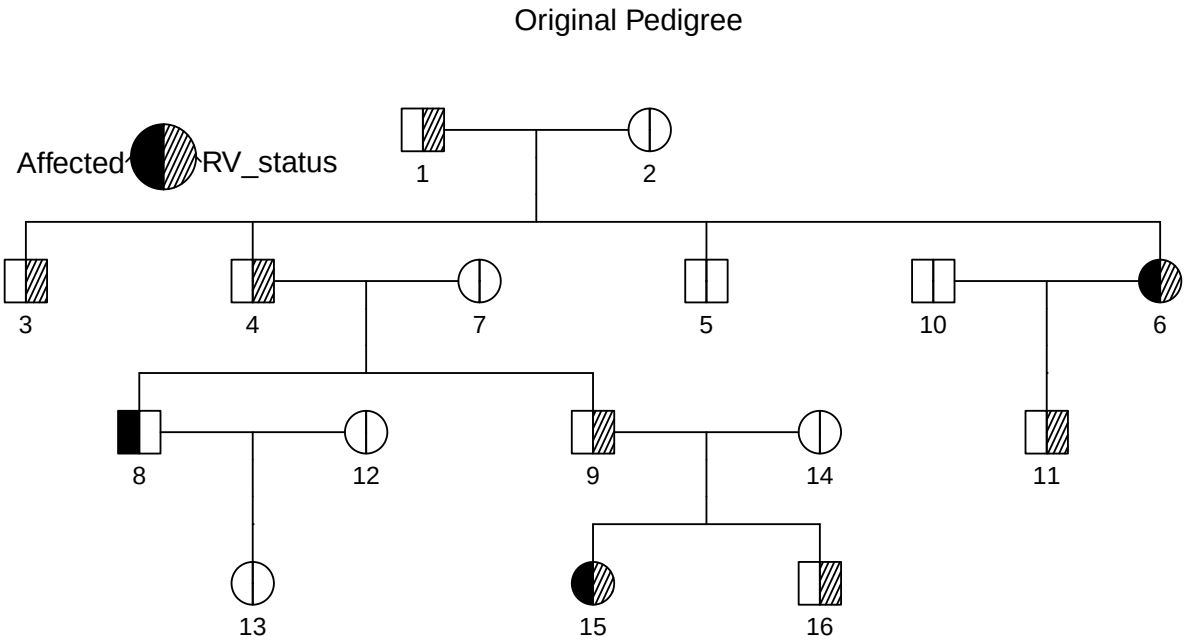
To trim a pedigree the user must specify:

1. `recall_probs`: the proband's relative recall probabilities, $p = (p_1, p_2, ..., p_n)$ where

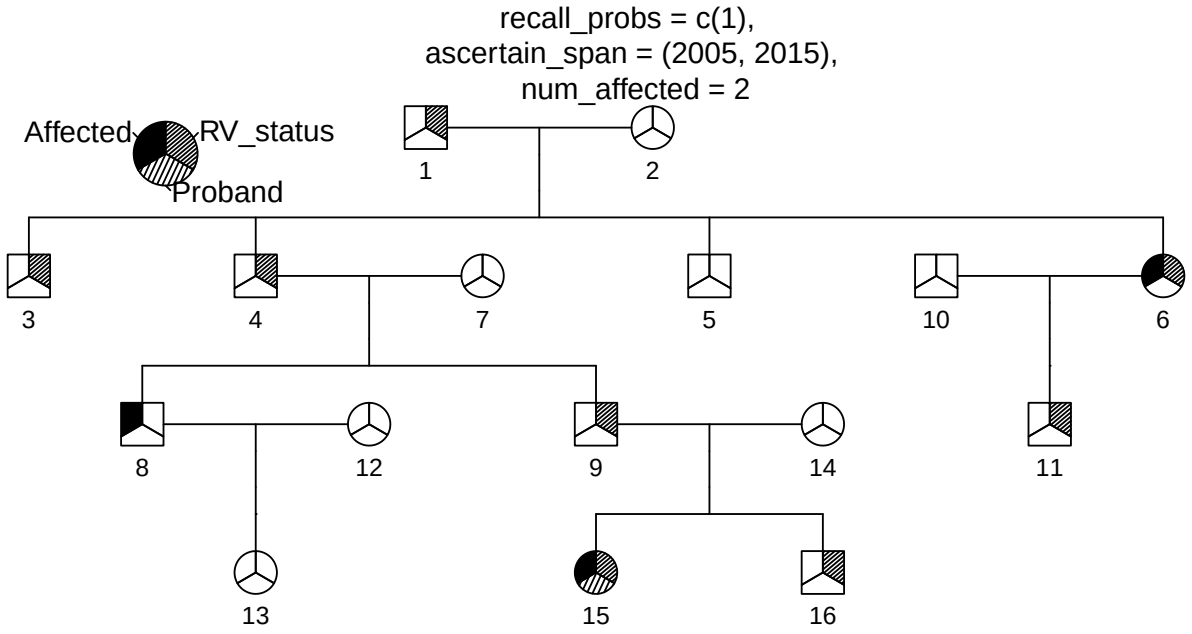
$$\rho_i = \begin{cases} \text{proband recall probability for a relative of degree } i & \text{for } i < n; \\ \text{proband recall probability for a relative of degree } n \text{ or greater} & \text{for } i = n. \end{cases}$$

   To simulate fully ascertained families simply specify `recall_probs = c(1)` so that the proband's recall probability of all relatives is 1.
2. `ascertain_span`: the ascertainment period in years. This period represents the range of years during which the proband developed disease and the family would have been ascertained for multiple affecteds.
3. `num_affected`: the minimum number of affected individuals in the pedigree at the time of ascertainment. NOTE: this does not ensure that after trimming the pedigree will contain at least `num_affected` individuals, it only ensures that the proband is chosen so that at least `num_affected`- 1 indivdiuals in the family were affected before the proband experienced onset.

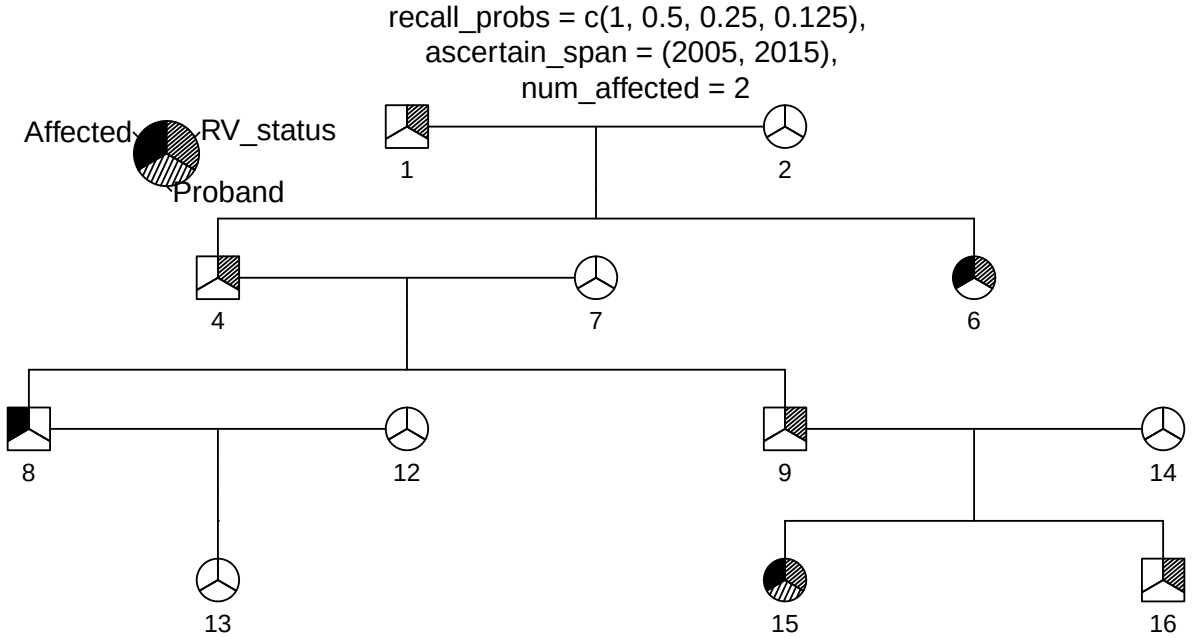
The following results illustrate how various arguement settings affect the `trim_ped`. Let's begin the example by considering the following pedigree prior to the application of `trim_ped`:



Notice that, with the exception of proband selection, the following trimmed pedigree is identical to the original pedigree. This is accomplished by setting `recall_probs = c(1)`.



In the next example the original pedigree has been trimmed after setting the proband recall probabilities to `recall_probs = c(1, 0.5, 0.25, 0.125)`.



If `recall_probs` is left unspecified, the `trim_ped` function will automatically use the 4*kinship coeffient between the proband and his or her relatives as the recall probability. This has the effect of retaining all first degree relatives with probability 1.

Simulating Pedigrees with Multiple Affecteds

Simulation of pedigrees likely to be ascertained according to a particular study design is the primary purpose of the `SimRVPedigree` package and is accomplished with the `sim_RVpedigree` function.

To simulate a pedigree with `sim_RVpedigree` the user must specify:

1. `onset_hazard`: the population age-specific disease onset hazard.
2. `death_hazard`: a data frame with:
    1. column 1: the unaffected age-specific death hazard,
    2. column 2: the affected age-specific death hazard. If the disease of interest is sufficiently rare, the unaffected age-specific death hazard my be approximated by the population age-specific death hazard.
3. `part`: a partition of ages over which to apply the age-specific hazard in `onset_hazard` and `death_hazard`.
4. `RR`: the relative risk of diesase onset for indivdiuals who have inherited the rare variant.
5. `FamID`: the family ID to assign to the simuated pedigree.
6. `founder_byears`: the span of years from which to simulate, uniformly, the founder's birth year.
7. `ascertain_span`: the ascertainment period in years. This period represents the range of years during which the proband developed disease and the family would have been ascertained for multiple affecteds.
8. `num_affected`: the minimum number of affected individuals in the pedigree.

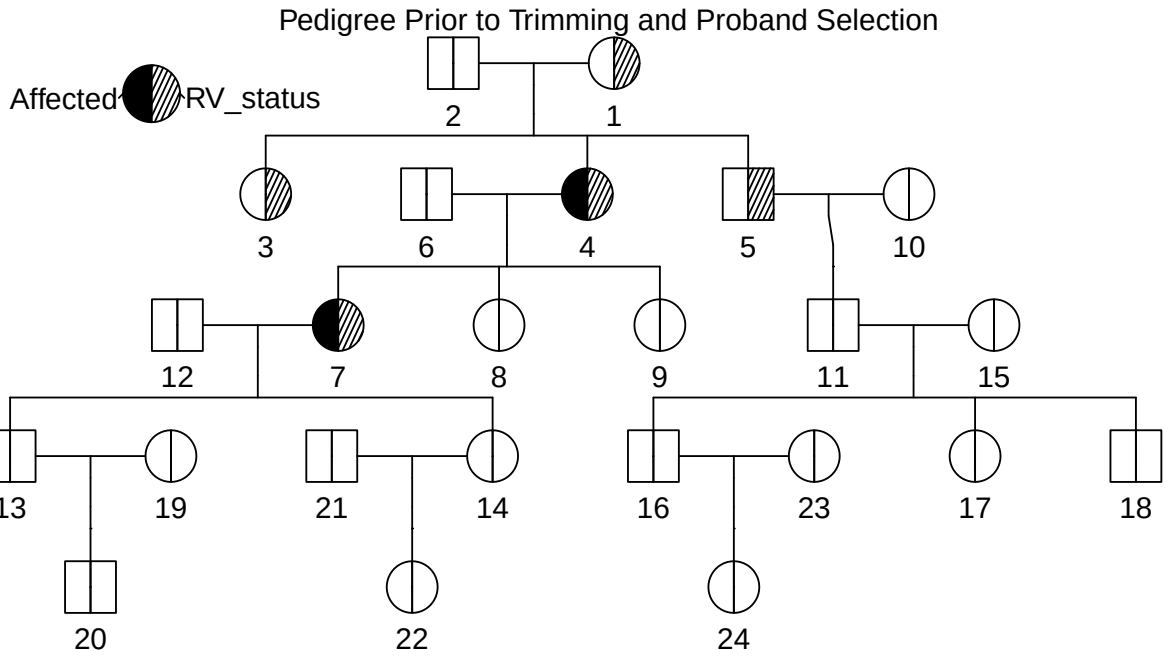
Optional arguments:

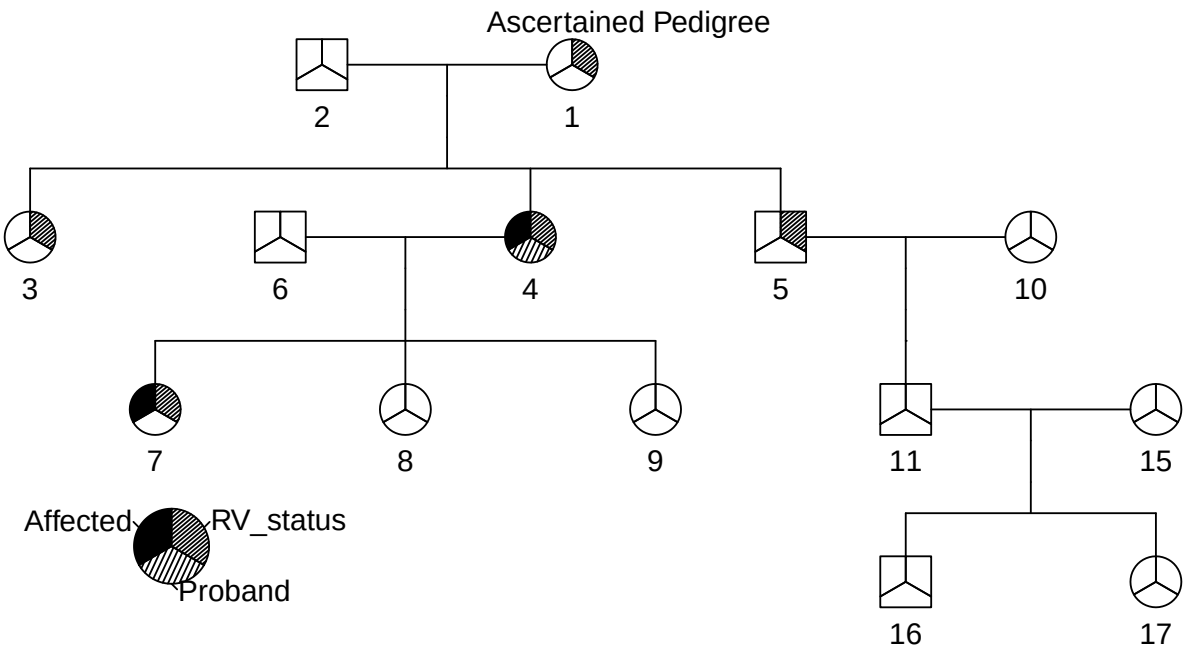
9. `recall_probs`: the proband's relative recall probabilities, if unspecified 4*kinship coeffient between the proband and the relative will be used.
10. `birth_range`: the minimum and maximum allowable ages indivduals will be able to reproduce. If unspecified it is assumed that `birth_range = c(18, 45)`.
11. `NB_params`: the size and probability parameters of the negative binomial distribution used to model the number of children per household. If unspecifed it is assumed that `NB_params = c(2, 4/7)`.
12. `stop_year`: the last year of study. If unspecified the current year will be used.

```
#Load hazard data
data(AgeSpecific_Hazards)

#specify onset hazard, death hazard, and age parition over which to apply
#hazards.
my_onset_hazard <- AgeSpecific_Hazards[,1]
my_death_hazard <- AgeSpecific_Hazards[,c(2,3)]
age_part <- seq(0, 100, by = 1)

 #Simulate a random pedigree
 set.seed(6)
 ex_RVped <- sim_RVpedigree(onset_hazard = my_onset_hazard,
                            death_hazard = my_death_hazard,
                            part = age_part,
                            RR = 5, FamID = 1, stop_year = 2015,
                            founder_byears = c(1900, 1910),
```

```
                        ascertain_span = c(1900, 2015),
                        num_affected = 2,
                        recall_probs = c(1, 0.5, 0.25, 0.125))
```

The `sim_RVpedigree` returns two data frames, which are plotted below. The first is the ped file of the original pedigree prior to trimming. The second is the ascertained pedigree, with proband selected and unrecalled relative removed. If a relative cannot be recalled by the proband but they are required to create a full pedigree structure they will be readded to the pedigree, but all of their information will be censored.





Parallel Processing Example

It is importatant to note that the processing time required to simulate a sample of pedigrees ascertained for multiple affecteds is directly related to (1) the rarity of the disease, and (2) the arguments specified by the user. In particular, we expect to see increases in the required computation time as:

1. as the relative risk, `RR`, specifed by the user approaches 1, particularly for rare diseases,
2. as ascertainment period, `ascertain_span`, narrows,
3. and as the proband's relative recall probabilities, `recall_probs`, become smaller.

Given the vast number of study designs that can be specified, it is likely that there are particular combinations of simulation setting may also lead to an increase in processsing time. For this reason we HIGHLY recommend the use of parallel processing when simulating a large set of pedigrees ascertained for multiple affected members.

The following parallel processing example uses the `doParallel` and `doRNG` packages to simulate a study sample of 100 pedigrees. In this simulation we make the following model assumptions:

1. Pedigrees are ascertained on the basis of containing at least 2 affected members: `num_affected = 2`.
2. Set the relative risk of disease onset for individuals who have inherited the rare variant of 1.5 : `RR = 1.5`.
3. Assume the birth year for the founder who introduced the rare variant is distributed uniformly from 1900 to 1980: `founder_byears = c(1900, 1980)`
4. Set the ascertainment period, i.e. the years the proband experienced onset and the pedigree was ascertained for multiple affecteds, from 2000 to 2016: `ascertain_span = c(2000, 2015)`.

5. Assume fully ascertained pedigrees, by assuming that the proband can recall all relatives: `recall_probs = c(1)`.
6. Assume that the study ended in 2015: `stop_year = 2015`

We will use the example age-specific hazards provided in the `AgeSpecific_Hazards` dataset. Notice that since the last age-specific hazard in this dataset applies to individuals between the ages of 99 and 100, the largest age possible in this simulation will be 100.

The following example was intended for Windows and slight modifications may be necessary for Unix-like systems.

```
#SITATION FOR DOPARALLEL VIGNETTE?
#Note that: while parallel processsing may acheived using only the doParallel
#package, to ensure that simulations are reproducible we must also incorporate
#the doRNG package.

#assuming they have been installed the required packages are loaded using the
#commands:
library(doParallel)
library(doRNG)

# Before we create our cluster, let's determine how many processors are
#currently in use using the getDoParWorkers function.  Since we have not created
#a cluster yet, this function should return 1.
getDoParWorkers()

#The number of cores available for parallel processing will depend on the the
#computer  being used.  To determine how many cores are available for parallel
#processing on your computer, simply execute the following command:
detectCores()

#To run simulations in parallel we must create a cluster and then register the
#cluster. The following code illustrates how to create and register a cluster
#that will simulate pedigrees in parallel on 2 cores.
cl <- makeCluster(2)       # create cluster
registerDoParallel(cl)     # register cluster

#Note that getDoParWorkers() should now return 2 instead of 1
getDoParWorkers()

#To avoid problems, after you are finished using the cluster, you will want to
#stop it. This can be acheived by executing:
on.exit(stopCluster(cl))
#which will stop the cluster when you end the R session, or by executing
#stopCluster(cl) after the simulation is complete.

#to ensure reproducibility, we make use of the %dorng% operator provided by the
#doRNG package, in the foreach loop, by specifing a seed after .option.RNG.
npeds <- 8    #set the number of pedigrees to generate

RV_peds = foreach(i = seq(npeds),
                  .combine = rbind,
                  .packages = c("kinship2", "SimRVPedigree"),
                  .options.RNG = 1984
```

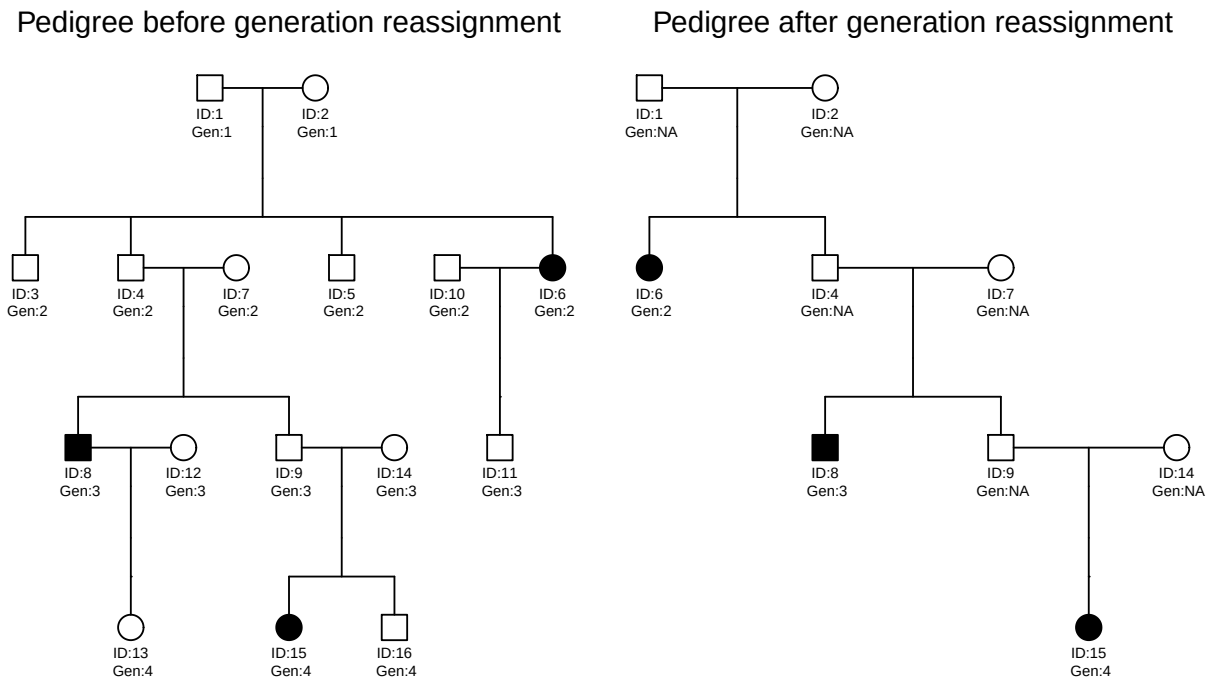
```
  ) %dorng% {
    sim_RVpedigree(onset_hazard = my_onset_hazard,
                   death_hazard = my_death_hazard,
                   part = age_part, RR = 1.5,
                   FamID = i, stop_year = 2015,
                   founder_byears = c(1900, 1980),
                   ascertain_span = c(2000, 2015),
                   recall_probs = c(1),
                   num_affected = 2)[[2]]
  }
```

Reassigning Affected Generation

Upon simulating a pedigree, occasionally researchers have reason to reassign generation number based on affected status.

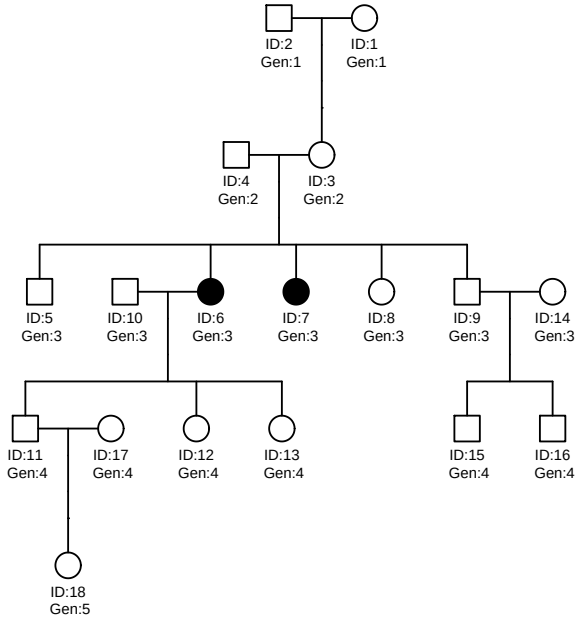
This task can be achieved using the `assign_affectedGen` function. The `assign_affectedGen` function assigns generation 1 to the first reported affected or the the first obligate carrier. The following examples illustrate the nature of this function.

In following example we see that since the first obligate carrier is one of the two individuals in generation 1, the generation assignment does not change.



Notice that in this example, since only individuals 6 and 7 are affected the first obligate carrier is individual 3 or 4, hence the affecteds are assigned generation 2.

Pedigree before generation reassignment



Pedigree after generation reassignment