kinship2 package vignette
Version 1.3.7

Jason Sinnwell and Beth Atkinson

October 29, 2012

1 Introduction

This document is a brief tutorial for the kinship2 package, with examples of creating pedigree objects and kinship matrices, and other pedigree utilities. If the kinship2 package is not loaded, we load it now.

```
> library(kinship2)
```

2 Basic Usage

Example Data

Two datasets are provided within the kinship2 package:

- *breast:* 17 families from a breast cancer study
- *sample.ped:* two sample pedigrees, with 41 and 14 subjects

This vignette uses the two pedigrees in *sample.ped*. For more information on these datasets, see *help*(*breast*) and *help*(*sample.ped*).

Pedigree

First, we load *sample.ped* and look at some of the values in the dataset, and create a *pedigreeList* object using the *pedigree*() function. We use the required arguments *id*, $father$, *mother*, and sex. The $famid$ argument is required to make a *pedigreeList* object, but not for a single *pedigree* object.

```
> data(sample.ped)
> sample.ped[1:10,]
```

```
   ped  id father mother sex affected avail
1    1 101      0      0   1        0     0
2    1 102      0      0   2        1     0
3    1 103    135    136   1        1     0
4    1 104      0      0   2        0     0
5    1 105      0      0   1       NA     0
6    1 106      0      0   2       NA     0
7    1 107      0      0   1        1     0
8    1 108      0      0   2        0     0
9    1 109    101    102   2        0     1
10   1 110    103    104   1        1     1
```

```
> pedAll <- pedigree(id=sample.ped$id,
+                 dadid=sample.ped$father, momid=sample.ped$mother,
+                 sex=sample.ped$sex, famid=sample.ped$ped)
> print(pedAll)
```

```
Pedigree list with 55 total subjects in 2 families

>
```

The *pedigreeList* object can be subset to individual pedigrees by their family id. The pedigree object has a print and plot method, which we show below. The print method prints a short summary of the pedigree, while the plot in Figure 1 displays the smaller pedigree.

```
> ped1basic <- pedAll['1']
> ped2basic <- pedAll['2']
> print(ped1basic)

Pedigree object with 41 subjects, family id= 1
Bit size= 46

> print(ped2basic)

Pedigree object with 14 subjects, family id= 2
Bit size= 16

> plot(ped2basic)
> # plot(ped1basic)
```

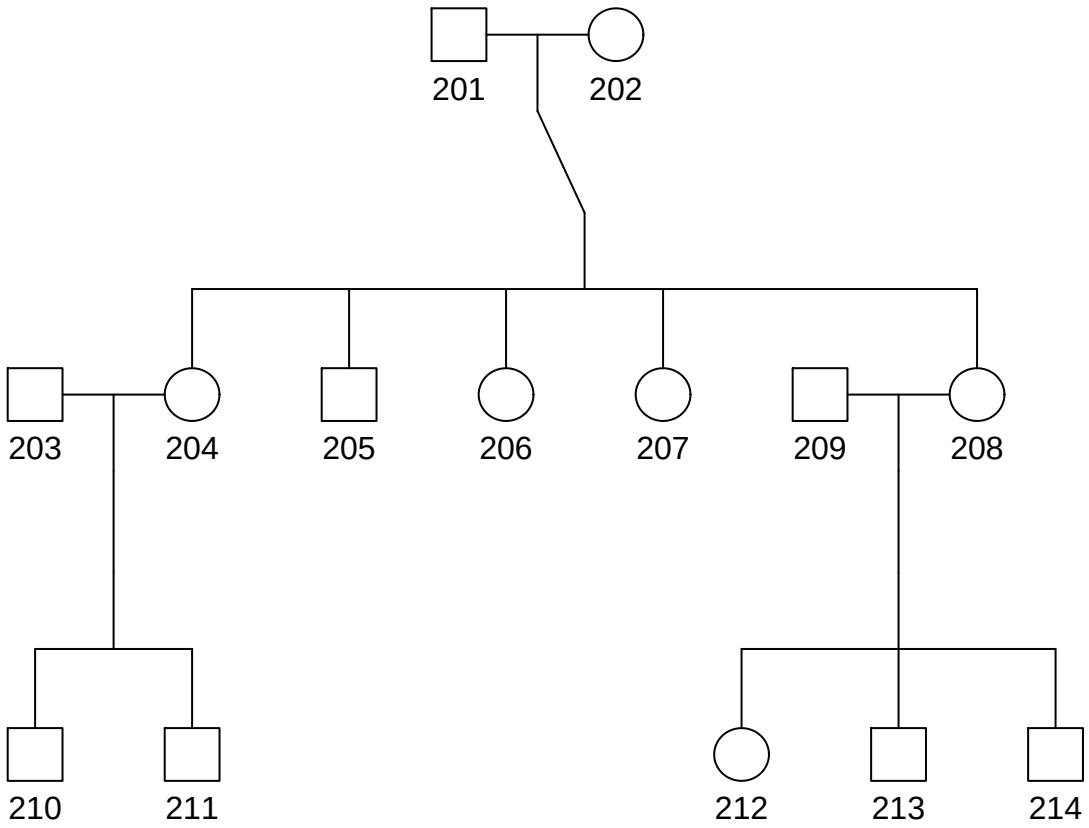


Figure 1: Basic Plot: pedigree 2

Kinship

A common use for pedigrees is to make a matrix of kinship coefficients that can be used in mixed effect models. A kinship coefficient is the probability that a randomly selected allele from two people at a given locus will be identical by descent (IBD), assuming all founder alleles are independent. For example, we each have two alleles per autosomal marker, so sampling two alleles with replacement from our own DNA has only $p = 0.50$ probability of getting the same allele twice.

Kinship for pedigree object

We use *kinship* to calculate the kinship matrix for *ped2basic*. The result is a special symmetrix matrix class from the *Matrix* R package[1], which is stored efficiently to avoid repeating elements.

```
> kin2 <- kinship(ped2basic)
> kin2
```

```
      201   202  203   204   205   206   207   208  209    210    211    212    213
201 0.500 0.000 0.00 0.250 0.250 0.250 0.250 0.250 0.00 0.1250 0.1250 0.1250 0.1250
202 0.000 0.500 0.00 0.250 0.250 0.250 0.250 0.250 0.00 0.1250 0.1250 0.1250 0.1250
203 0.000 0.000 0.50 0.000 0.000 0.000 0.000 0.000 0.00 0.2500 0.2500 0.0000 0.0000
204 0.250 0.250 0.00 0.500 0.250 0.250 0.250 0.250 0.00 0.2500 0.2500 0.1250 0.1250
205 0.250 0.250 0.00 0.250 0.500 0.250 0.250 0.250 0.00 0.1250 0.1250 0.1250 0.1250
206 0.250 0.250 0.00 0.250 0.250 0.500 0.250 0.250 0.00 0.1250 0.1250 0.1250 0.1250
207 0.250 0.250 0.00 0.250 0.250 0.250 0.500 0.250 0.00 0.1250 0.1250 0.1250 0.1250
208 0.250 0.250 0.00 0.250 0.250 0.250 0.250 0.500 0.00 0.1250 0.1250 0.2500 0.2500
209 0.000 0.000 0.00 0.000 0.000 0.000 0.000 0.000 0.50 0.0000 0.0000 0.2500 0.2500
210 0.125 0.125 0.25 0.250 0.125 0.125 0.125 0.125 0.00 0.5000 0.2500 0.0625 0.0625
211 0.125 0.125 0.25 0.250 0.125 0.125 0.125 0.125 0.00 0.2500 0.5000 0.0625 0.0625
212 0.125 0.125 0.00 0.125 0.125 0.125 0.125 0.250 0.25 0.0625 0.0625 0.5000 0.2500
213 0.125 0.125 0.00 0.125 0.125 0.125 0.125 0.250 0.25 0.0625 0.0625 0.2500 0.5000
214 0.125 0.125 0.00 0.125 0.125 0.125 0.125 0.250 0.25 0.0625 0.0625 0.2500 0.2500
       214
201 0.1250
202 0.1250
203 0.0000
204 0.1250
205 0.1250
206 0.1250
207 0.1250
208 0.2500
209 0.2500
210 0.0625
211 0.0625
212 0.2500
213 0.2500
214 0.5000
```

```
>
```

For family 2, see that the row and column names match the id in Figure 1, and see that each person's coefficient with themselves is 0.50, siblings are 0.25 (e.g. $204 - 205$), and pedigree marry-ins only share alleles IBD with their children with coefficient 0.25 (e.g. $203 - 210$). The plot can be used to verify other kinship coefficients.

Kinship for pedigreeList object

The kinship function also works on a *pedigreeList* object. We show how to create the kinship matrix, then show a snapshot of them for the two families, where the row and columns names are the ids of the subject.

```
> pedAll <- pedigree(id=sample.ped$id,
+                 dadid=sample.ped$father, momid=sample.ped$mother,
+                 sex=sample.ped$sex, famid=sample.ped$ped)
> kinAll <- kinship(pedAll)
> kinAll[1:14,1:14]

14 x 14 sparse Matrix of class "dsCMatrix"

101 0.50 .    .    .    .    .    .    0.25 .    .    .    .    .
102 .    0.50 .    .    .    .    .    0.25 .    .    .    .    .
103 .    .    0.50 .    .    .    .    .    0.25 0.25 0.25 .    0.25
104 .    .    .    0.50 .    .    .    .    0.25 0.25 0.25 .    0.25
105 .    .    .    .    0.5  .    .    .    .    .    .    .    .
106 .    .    .    .    .    0.5  .    .    .    .    .    .    .
107 .    .    .    .    .    .    0.5  .    .    .    .    .    .
108 .    .    .    .    .    .    .    0.5  .    .    .    .    .
109 0.25 0.25 .    .    .    .    .    0.50 .    .    .    .    .
110 .    .    0.25 0.25 .    .    .    .    .    0.50 0.25 0.25 .    0.25
111 .    .    0.25 0.25 .    .    .    .    .    0.25 0.50 0.25 .    0.25
112 .    .    0.25 0.25 .    .    .    .    .    0.25 0.25 0.50 .    0.25
113 .    .    .    .    .    .    .    .    .    .    .    .    0.5  .
114 .    .    0.25 0.25 .    .    .    .    .    0.25 0.25 0.25 .    0.50

> kinAll[40:43,40:43]

4 x 4 sparse Matrix of class "dsCMatrix"
     140  141 201 202
140 0.50 0.25 .   .
141 0.25 0.50 .   .
201 .    .    0.5 .
202 .    .    .   0.5

> kinAll[42:46, 42:46]

5 x 5 sparse Matrix of class "dsCMatrix"
     201  202 203  204  205
201 0.50 .    .   0.25 0.25
202 .    0.50 .   0.25 0.25
203 .    .    0.5 .    .
204 0.25 0.25 .   0.50 0.25
205 0.25 0.25 .   0.25 0.50

>
```

Note that subject 113 is not in pedigree 1 because they are a marry-in without children in the pedigree. Subject 113 is in their own pedigree of size 1 in the *kinAll* matrix at index 41. We later show how to handle such marry-ins for plotting.

3 Optional Pedigree Features

We use pedigree 2 from *sample.ped* to sequentially add optional information to the pedigree object.

Status

The example below shows how to specify a *status* indicator, such as vital status. The *sample.ped* data does not include such an indicator, so we create one to indicate that the first generation of pedigree 2, subjects 1 and 2, are deceased.

```
> df2 <- sample.ped[sample.ped$ped==2,]
> names(df2)

[1] "ped"      "id"       "father"   "mother"   "sex"      "affected" "avail"

> df2$censor <- c(1,1, rep(0, 12))
> ped2 <- pedigree(df2$id, df2$father, df2$mother,
+                  df2$sex, status=df2$censor)
>
```

Affected Indicators

We show how to specify affected status with a single indicator and multiple indicators in a matrix. First, we use the affected indicator from *sample.ped*, which contains 0/1 indicators and NA as missing, and let's imagine it indicates blue eyes. Next, we create a matrix to contain the affected indicator from *sample.ped* and a second indicator that we create, imagine as an indicator for baldness.

```
> ped2 <- pedigree(df2$id, df2$father, df2$mother,
+                  df2$sex, affected=df2$affected,
+                  status=df2$censor)
> aff2 <- data.frame(blue=df2$affected,
+                    bald=c(0,0,0,0,1,0,0,0,0,1,1,0,0,1))
> ped2 <- pedigree(df2$id, df2$father, df2$mother,
+                  df2$sex, affected=as.matrix(aff2),
+                  status=df2$censor)
>
```

Special Relationships

Special pedigree relationships can be specified in a matrix as the *relation* argument. There are 4 relationships that can be specified by numeric codes: 1=Monozygotic twins, 2=Dizygotic twins, 3=Twins of unknown zygosity, and 4=Spouse. The spouse relationship can indicate a marry-in when a couple does not have children together.

Below, we create a matrix of relationships for monozygotic and unknown-zygosity twins in the most recent generation of pedigree 2.

```
> ## create twin relationships
> relate2 <- matrix(c(210,211,1,
+                     212,213,3), nrow=2, byrow=TRUE)
> ped2 <- pedigree(df2$id, df2$father, df2$mother,
+                  df2$sex, affected=as.matrix(aff2),
+                  status=df2$censor,
+                  relation=relate2)
```

4 Pedigree Plot Details

The plot method does an admirable job plotting pedigrees within the standard R plotting paradigm. It attempts to adhere to many standards in pedigree plotting, as presented in Bennet et al., 2008[2].

We show in Figure 2 the plot of the updated pedigree 2. The plot shapes for each subject are divided into two equal parts and shaded differently to indicate the two affected indicators. Also, the two deceased subjects are displayed with a diagonal line through the shape. The twin relationships are both represented with diverging lines from a single point. The monozygotic twins have an additional line connecting the diverging lines, while the other twins have a question mark to indicate unknown zygosity.

We also show how the subjects can be colored individually, where we color a subject's shape red if their *avail* indicator is 1, which can represent their DNA availability, a useful indicator in genetic studies. Lastly, we show how to use the *id*

argument in the plot method to add additional information under each subject. In the example below, we add names to the existing *id* vector using the newline character as the *sep* argument in *paste*(). As space permits, more lines and characters per line can be made using the *id* argument.

Finally, we show how a pedigree.legend can place a simple legend in one of the corners of the pedigree plot to show the sections of the plot symbols corresponding to the multiple affected indicators.

```
> id2 <- paste(df2$id, c("John", "Linda", "Jack", "Rachel", "Joe", "Deb",
+                        "Lucy", "Ken", "Barb", "Mike", "Matt",
+                        "Mindy", "Mark", "George"), sep="\n")
> plot(ped2, col=ifelse(df2$avail, 2, 1),
+      id=id2)
> pedigree.legend(ped2, location="topright", radius=.3)
```

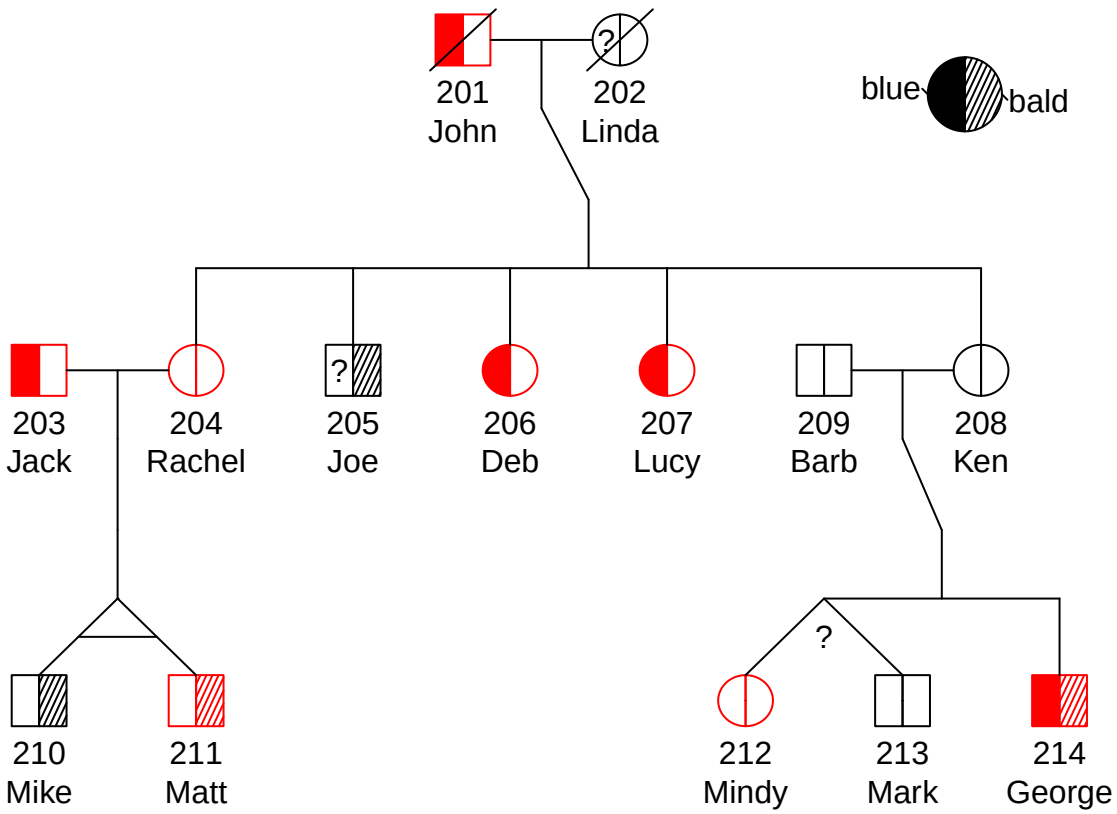


Figure 2: Updated plot: pedigree 2

To show some other tricks with pedigree plotting, we use pedigree 1 from *sample.ped*, which has 41 subjects in 4 generations, including a generation with double first cousins. After the first marriage of 114, they remarried subject 113 without children between them. If we do not specify the marriage with the *relation* argument, the plot method excludes subject 113 from the plot. The basic plot of pedigree 1 is shown in Figure 3, where the subjects are colored red if their *avail* indicator is 1.

```
> df1<- sample.ped[sample.ped$ped==1,]
> relate1 <- matrix(c(113, 114, 4), nrow=1)
> ped1 <- pedigree(df1$id, df1$father, df1$mother,
+        df1$sex, affected=df1$affected,
+                  relation=relate1)
> print(ped1)

Pedigree object with 41 subjects
Bit size= 46

> plot(ped1, col=df1$avail+1)
>
```

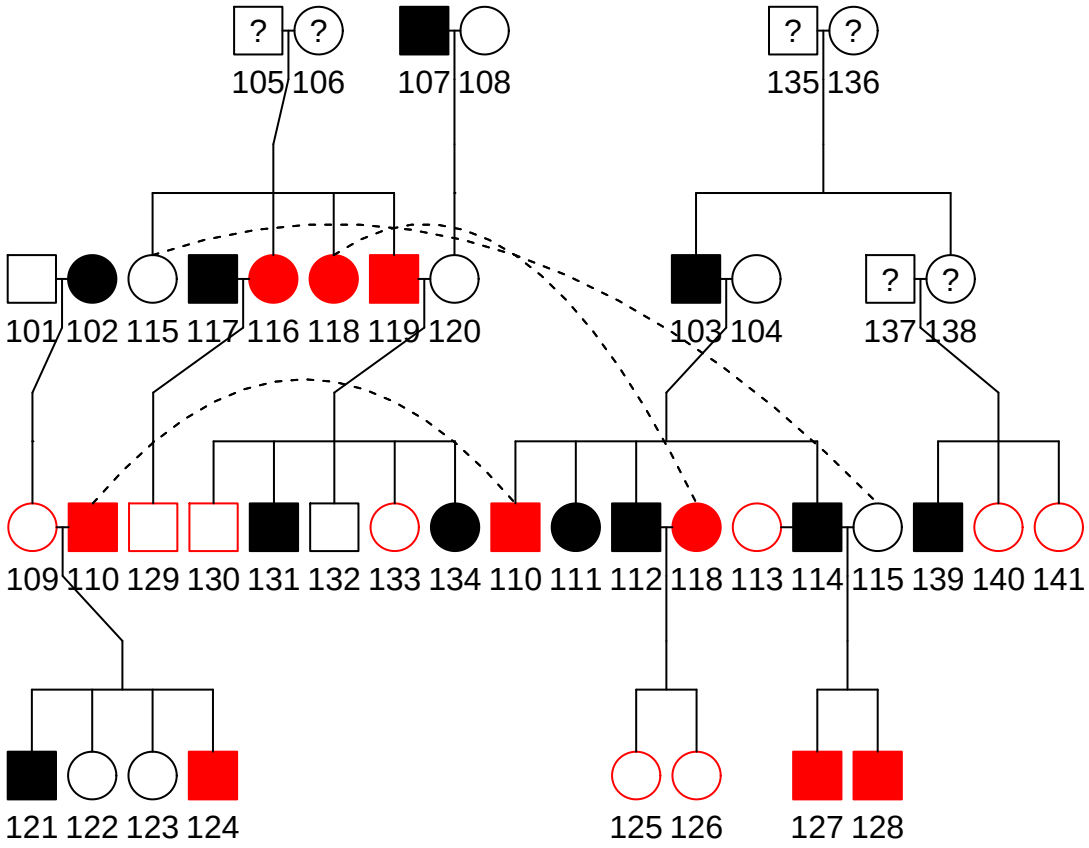


Figure 3: Pedigree 1, Original

Align by Input Order

The plot method does a decent job aligning subjects given the order of the subjects when the pedigree object is made, and sometimes has to make two copies of a subject. If we change the order of the subjects when creating the pedigree, we can help the plot method reduce the need to duplicate subjects, as Figure 4 no longer has subject 110 duplicated.

```
> df1reord <- df1[c(35:41,1:34),]
> ped1reord <- pedigree(df1reord$id, df1reord$father, df1reord$mother,
+         df1reord$sex, affected=df1reord$affected, relation=relate1)
> plot(ped1reord, col=df1reord$avail+1)
>
```

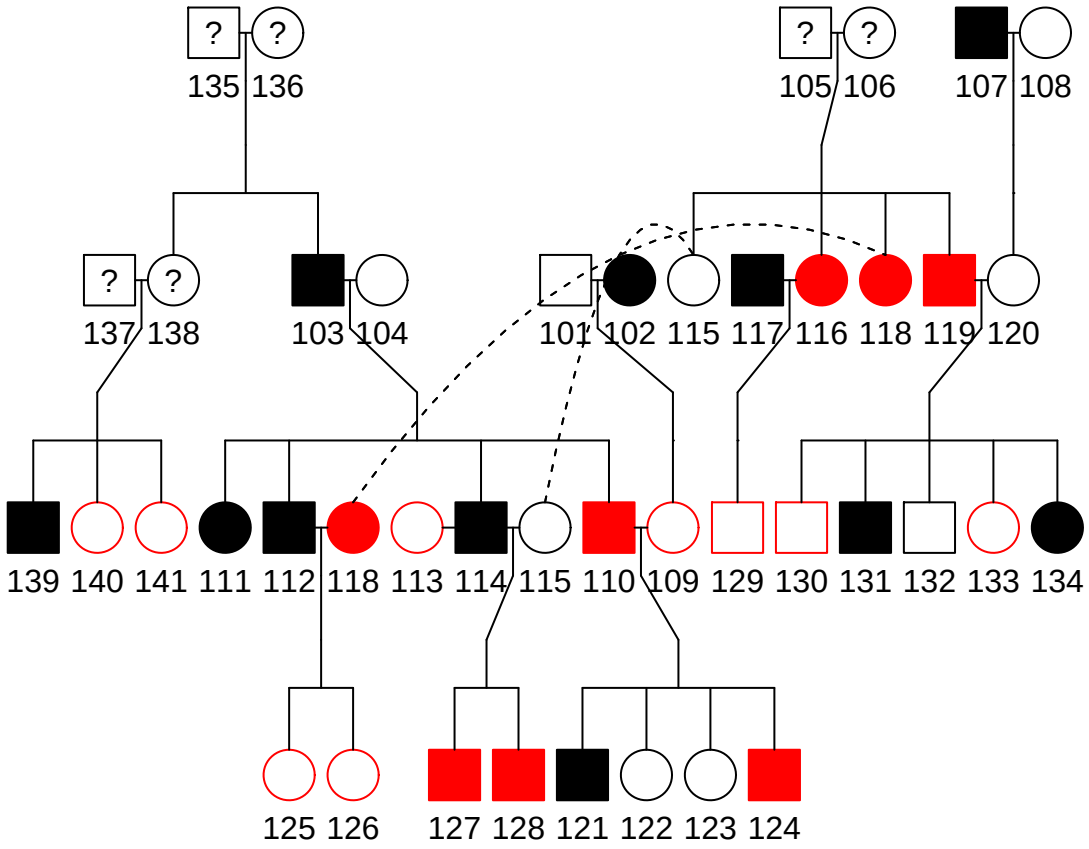


Figure 4: Pedigree 1, Re-Ordered

5 Pedigree Utility Functions

Data.Frame

A pedigree object's main features are vectors with an element for each subject. It is sometimes useful to extract these vectors from the pedigree object into a *data.frame* with basic information that can be used to construct a new pedigree object. This is possible with the *as.data.frame*() method, as shown below.

```
> dfped2 <- as.data.frame(ped2)
> dfped2
```

```
    id dadid momid    sex affected.blue affected.bald status
1  201     0     0   male             1             0      1
2  202     0     0 female            NA             0      1
3  203     0     0   male             1             0      0
4  204   201   202 female             0             0      0
5  205   201   202   male            NA             1      0
6  206   201   202 female             1             0      0
7  207   201   202 female             1             0      0
8  208   201   202 female             0             0      0
9  209     0     0   male             0             0      0
10 210   203   204   male             0             1      0
11 211   203   204   male             0             1      0
12 212   209   208 female             0             0      0
13 213   209   208   male             0             0      0
14 214   209   208   male             1             1      0
```

```
>
```

Subsetting and Trimming

Pedigrees with large size can be a bottleneck for programs that run calculations on them. The kinship2 package contains some routines to identify which subjects to remove. We show how a subject (e.g. subject 210) can be removed from *ped*2, and how the pedigree object is changed by verifying that the *relation* matrix no longer has the twin relationship between subjects 210 and 211, as indicated by *indx*1 and *indx*2. Also note that the *relation* matrix indices are updated for persons 212 and 213 who have index 11 and 12 after subject 210 is removed.

```
> ped2.rm210 <- ped2[-10]
> data.frame(ped2.rm210)
```

```
    id dadid momid    sex affected.blue affected.bald
1  201     0     0   male             1             0
2  202     0     0 female            NA             0
3  203     0     0   male             1             0
4  204   201   202 female             0             0
5  205   201   202   male            NA             1
6  206   201   202 female             1             0
7  207   201   202 female             1             0
8  208   201   202 female             0             0
9  209     0     0   male             0             0
10 211   203   204   male             0             1
11 212   209   208 female             0             0
12 213   209   208   male             0             0
13 214   209   208   male             1             1
```

```
> ped2.rm210$relation
```

```
  indx1 indx2    code
2    11    12 UZ twin

> ped2$relation

  indx1 indx2    code
1    10    11 MZ twin
2    12    13 UZ twin

>
>
```

The steps above only work for subsetting by the index of the pedigree object vectors, not by the *id* of the subjects themselves. We provide *pedigree.trim*, which trims subjects from a pedigree by their *id*. Below is an example of removing subject 110, as done above, then we further trim the pedigree by a vector of subject ids. We check the trimming by looking at the *id* vector and the *relation* matrix.

```
> ped2.trim210 <- pedigree.trim(210, ped2)
> ped2.trim210$id

 [1] 201 202 203 204 205 206 207 208 209 211 212 213 214

> ped2.trim210$relation

  indx1 indx2    code
2    12    13 UZ twin

> ped2.trim.more <- pedigree.trim(c(212,214), ped2.trim210)
> ped2.trim.more$id

 [1] 201 202 203 204 205 206 207 208 209 211 213

> ped2.trim.more$relation

[1] indx1 indx2 code
<0 rows> (or 0-length row.names)

>
```

6 Shrinking

A new function in kinship2 is *pedigree.shrink*, which shrinks a pedigree to a specified bit size while maintaining the maximal amount of information for genetic linkage and association studies. Using an indicator for availability and affected status, it removes subjects in this order:

1. unavailable with no available descendants
2. available and are not parents
3. available who have missing affected status
4. available who are unaffected
5. available who are affected

We show how to shrink pedigree 1 to bit size 30, which happens to be the bit size after removing only the unavailable subjects. We show how to extract the shrunken pedigree object from the *pedigree.shrink* result, and plot it.

```
> shrink1.B30 <- pedigree.shrink(ped=ped1,
+                   avail=df1$avail, maxBits=30)
> print(shrink1.B30)

================================================================================
                                  Shrink of Pedigree
================================================================================
Pedigree Size:
                 N.subj Bits
Original             41   46
Only Informative     28   29
Trimmed              28   29

 Unavailable subjects trimmed:
 101 102 107 108 111 113 121 122 123 131 132 134 139

> ped1.B30 <- shrink1.B30$pedObj
> plot(ped1.B30, col=shrink1.B30$avail + 1)
>
>
```

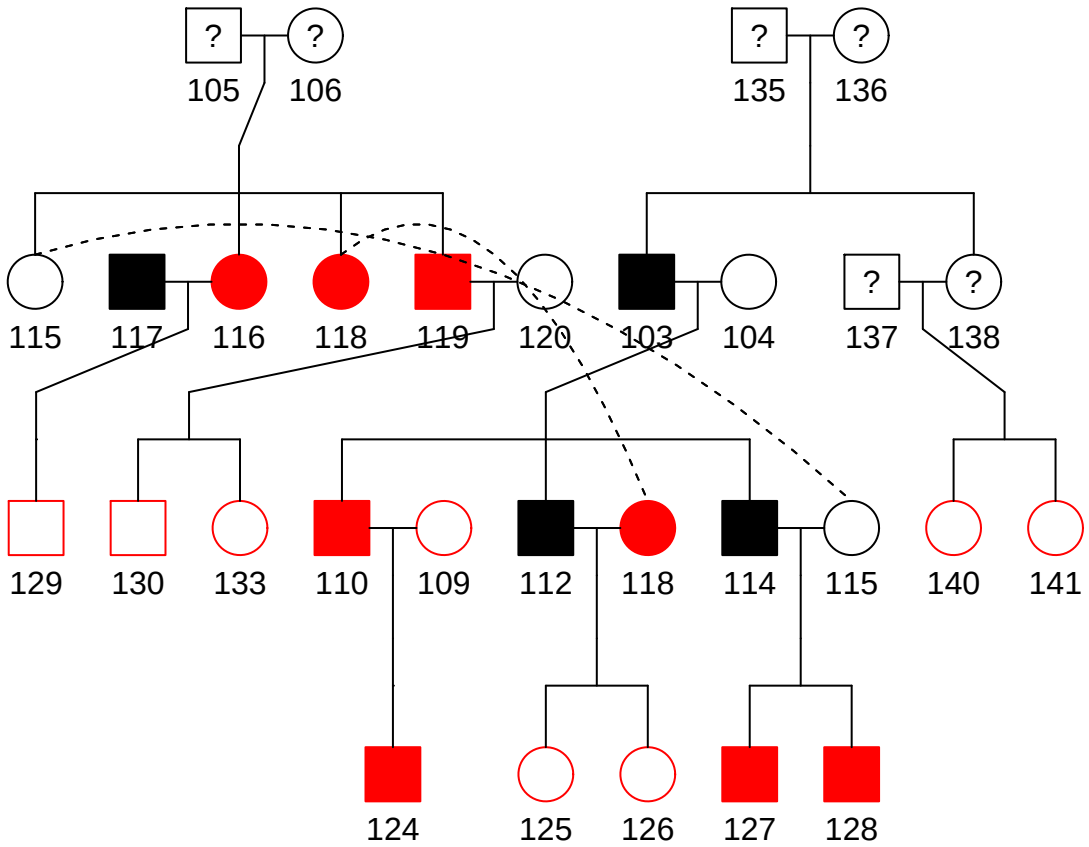


Figure 5: Pedigree 1, 30 bits

Now shrink pedigree 1 to bit size 25, which requires removing subjects who are informative. If there is a tie between multiple subjects about who to remove, the method randomly chooses one of them. With this seed setting, the method removes subjects 126 then 125.

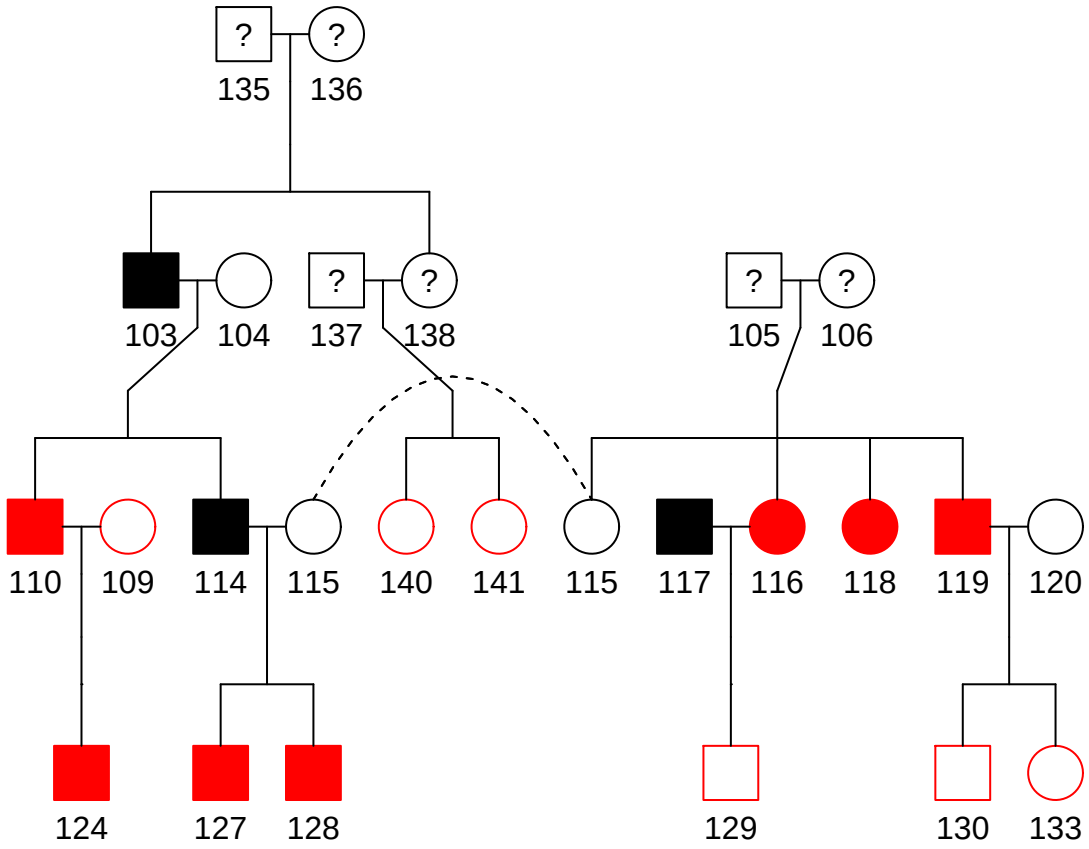
```
> set.seed(10)
> shrink1.B25 <- pedigree.shrink(ped=ped1, avail=df1$avail,
+                                maxBits=25)
> print(shrink1.B25)

================================================================================
                                  Shrink of Pedigree
================================================================================
Pedigree Size:
                 N.subj Bits
Original             41   46
Only Informative     28   29
Trimmed              25   23

 Unavailable subjects trimmed:
 101 102 107 108 111 113 121 122 123 131 132 134 139

 Informative subjects trimmed:
 126 125

> ped1.B25 <- shrink1.B25$pedObj
> plot(ped1.B25, col=shrink1.B25$avail + 1)
>
```



7 Select Unrelateds

In this section we briefly show how to use *pedigree.unrelated* to find a set of the maximum number of unrelated available subjects from a pedigree. The input required is a pedigree object and a vector indicating availability. In some pedigrees there are numerous sets of subjects that satisfy the maximum number of unrelateds, so the method randomly chooses from the set. We show two sets of subject ids that are selected by the routine and discuss below.

```
> df2<- sample.ped[sample.ped$ped==2,]
> ped2 <- pedigree(df2$id, df2$father, df2$mother,
+        df2$sex, affected=df2$affected)
> set.seed(10)
> set1 <- pedigree.unrelated(ped2, avail=df2$avail)
> set1

[1] "203" "206"

> set2 <- pedigree.unrelated(ped2, avail=df2$avail)
> set2

[1] "203" "214"

>
```

We can easily verify the sets selected by *pedigree.unrelated* by referring to Figure 1 and see that subjects 203 and 209 are unrelated to everyone else in the pedigree except their children. Furthermore, we see in *df*2 that of these two, only subject 203 is available. Therefore, any set of unrelateds who are available must include subject 203 and one of the these subjects: 201, 204, 206, 207, 212, and 214, as indicated by the kinship matrix for pedigree 2 subset to those with availability status of 1.

```
> df2

   ped  id father mother sex affected avail
42   2 201      0      0   1        1     1
43   2 202      0      0   2       NA     0
44   2 203      0      0   1        1     1
45   2 204    201    202   2        0     1
46   2 205    201    202   1       NA     0
47   2 206    201    202   2        1     1
48   2 207    201    202   2        1     1
49   2 208    201    202   2        0     0
50   2 209      0      0   1        0     0
51   2 210    203    204   1        0     0
52   2 211    203    204   1        0     1
53   2 212    209    208   2        0     1
54   2 213    209    208   1        0     0
55   2 214    209    208   1        1     1

> kin2[df2$avail==1,df2$avail==1]

      201  203   204   206   207    211    212    214
201 0.500 0.00 0.250 0.250 0.250 0.1250 0.1250 0.1250
203 0.000 0.50 0.000 0.000 0.000 0.2500 0.0000 0.0000
204 0.250 0.00 0.500 0.250 0.250 0.2500 0.1250 0.1250
206 0.250 0.00 0.250 0.500 0.250 0.1250 0.1250 0.1250
207 0.250 0.00 0.250 0.250 0.500 0.1250 0.1250 0.1250
211 0.125 0.25 0.250 0.125 0.125 0.5000 0.0625 0.0625
212 0.125 0.00 0.125 0.125 0.125 0.0625 0.5000 0.2500
214 0.125 0.00 0.125 0.125 0.125 0.0625 0.2500 0.5000

>
```

8 R Session Information

```
> toLatex(sessionInfo())
```

- R version 2.15.0 (2012-03-30), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Matrix 1.0-6, kinship2 1.3.7, lattice 0.20-6, quadprog 1.5-4
- Loaded via a namespace (and not attached): grid 2.15.0, tools 2.15.0

References

[1] Bates D and Maechler M. (2011). **Matrix: Sparse and Dense Matrix Classes and Methods.** R package version 0.999375-50. *http://CRAN.R-project.org/package=Matrix*

[2] Bennet RL, Steinhaus French K, Resta RG, Lochner Doyle D. (2008). **Standardized Human Pedigree Nomenclature: Update and Assessment of the Recommendations of the National Society of Genetic Counselors.** *J. Gene. Counsel.,* 17, 1424-433.

[3] Sinnwell JP, Therneau TM, Atkinson EJ, Schaid DJ, Matsumoto ME, McDonnell SK (2011). **kinship2: An enhancement to the kinship R package with additional pedigree utilities**. Submitted.