`waffect` Package (v.1.2) Tutorial

Vittorio Perduca and Gregory Nuel

This document provides a short overview of how to use `waffect`. It shows how to simulate phenotypes given a disease model in the binary and multiclass cases and then how to use `waffect` for assessing the power of GWAs. Details about the algorithms implemented in the package can be found in [2].

We assume that the reader has correctly installed the package by running the command `install.packages("waffect")` from the `R` Console.

Disclaimer. If you use `waffect` in published research, please cite the companion article [2]. The authors would be glad to hear how `waffect` is employed. You are kindly encouraged to notify Gregory Nuel <`gregory.nuel@parisdescartes.fr`> and Vittorio Perduca <`vittorio.perduca@parisdescartes.fr`> about any work you publish that makes use of `waffect`.

Notations. n is the total number of individuals in the study. The phenotype and genotype of the j-th individual are Y_j and X_j respectively.

1 Simulating case/controls phenotypes

Notations. In the binary case, phenotypes are denoted by $1, 0$: $Y_j = 1$ if individual j is a case, $Y_j = 0$ if individual j is a control. By *disease model* we mean the vector of probabilities $\pi = (\pi_j)_{j=1,\dots,n}$ with $\pi_j = \mathbb{P}(Y_j = 1|X_j)$, that is the conditional probability that the j-th individual is a case given her/his genotype. The number of cases and controls are n_1 and n_0 respectively (with $n = n_0 + n_1$). These numbers are fixed from the beginning: for instance they are the actual number of cases and controls observed in the real dataset. The package makes it possible to simulate a new vector of phenotypes accordingly to the disease model and such that the number of cases and controls are exactly n_1 and n_0 respectively.

Suppose $n = 5$ and consider the vector of probabilities[1]

$$\pi = (0.5, 0.2, 0.9, 0.7, 0.1).$$

We simulate a phenotypic dataset for these individuals accordingly to the disease model H1 given by π and such that the number of cases is $n_1 = 2$ ($n_2 = 3$) as follows using the function `waffect`:

[1] In Section 3 we will show how to compute such probabilities from the genotypes and given a disease model.

```
> library(waffect); set.seed(42)
> pi <- c(0.5,0.2,0.9,0.3,0.1)
> waffect(prob=pi, count=c(2,3), label=c(1,0))

[1] 1 0 1 0 0
```

As we didn't specify the method, phenotypes are assigned with the default *backward* algorithm. The output is a list of phenotypes coded by `1`s and `0`s. By rerunning the function `waffect` we obtain another phenotypic dataset with $n_1 = 2$:

```
> waffect(prob=pi, count=c(2,3), label=c(1,0))

[1] 1 0 1 0 0
```

Instead of specifying both n_1 and n_2 we can simply enter the the number n_1 of cases:

```
> waffect(prob=pi, count=2, label=c(1,0))

[1] 1 0 1 0 0
```

If we forget to specify the labels, by default the codes for cases and controls will be 1 and 0:

```
> waffect(prob=pi, count=c(2,3))

[1] 1 0 1 0 0
```

However we are free to use our preferred coding, the only thing to remember is that the first element in the argument of `label` must be the code for cases:

```
> waffect(prob=pi, count=c(2,3), label=c("case", "control"))

[1] "case"    "control" "case"    "control" "control"
```

The following command generates phenotypes with cases and controls coded by 2 and 1 to be used with `PLINK` (see Section 3)

```
> waffect(prob=pi, count=c(2,3), label=c(2,1))

[1] 2 1 2 1 1
```

Under the null hypothesis H0, each individual has the same probability to be a case (no matter her/his genotype), that is all the entries in π are equal. If we want to simulate a phenotypic dataset under H0, it is sufficient to specify the numbers n_1, n_2 of cases and controls:

```
> waffect(count=c(7,9), label=c(1,0))

 [1] 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1
```

By default, the function will consider `prob=rep(0.1, sum(count))`. Equivalently, if we are given the observed phenotypes

```
> obs <- c(1,1,0,0,0,1,0,1,1,0)
```

we can simulate a new phenotypic dataset under H0 by simply permuting them with the standard `R` function

```
> sample(obs)

 [1] 0 1 0 1 0 1 0 1 1 0
```

The permutation breaks any link between phenotypes and genotypes.

2 Simulating phenotypes in the multiclass case

Notations. If there are K classes, phenotypes are denoted by an integer in $\{1, \dots, K\}$. By *disease model* we mean the $K \times n$ matrix of probabilities $\pi = (\pi_{kj})$ with $\mathbb{P}(Y_j = k|X_j) = \pi_{kj}$. By n_k we denote the total number of individuals in the k-th class. The package makes it possible to simulate a new vector of phenotypes accordingly to the disease model and such that for each k the total number of individuals with phenotype in the k-th class is exactly n_k.

Suppose we have $n = 5$ individuals each belonging to one out of $K = 3$ classes, and that the disease model is given by the matrix

$$\pi = \begin{pmatrix} 0.5 & 0.3 & 0.1 & 0.2 & 0.1 \\ 0.4 & 0.5 & 0.1 & 0.6 & 0.7 \\ 0.1 & 0.2 & 0.8 & 0.2 & 0.2 \end{pmatrix}.$$

We assign a class to each individual so that there are exactly $n_1 = 2$ individuals in the first class, $n_2 = 2$ individuals in the second class and $n_3 = 1$ individuals in the third class:

```
> pi1 = c(0.5,0.4,0.1)
> pi2 = c(0.3,0.5,0.2)
> pi3 = c(0.1,0.1,0.8)
> pi4 = c(0.2,0.6,0.2)
> pi5 = c(0.1,0.7,0.2)
> pi = cbind(pi1,pi2,pi3,pi4,pi5)
> waffect(prob = pi, count = c(2,2,1),label=1:3)

[1] 2 1 3 2 1
```

The output is the list of classes of all the individuals.

3 Assessing the power of GWAs

Given a GWA study method, it is crucial to assess its statistical power to detect susceptibility variants. Power can be estimated empirically by simulating disease (case and control) phenotypes. We illustrate how to asses the statistical power of GWA studies using `waffect` for phenotype simulations. In particular we will proceed as follows:

- We consider the toy dataset with 100 individuals (40 cases and 60 controls) and 1000 SNPs included in the package.
- We assume an arbitrarily defined single marker disease model H1 and two methods of associations: for each method, the aim is to assess its statistical power to detect the susceptibility SNP.
- By using `waffect` we simulate 200 phenotypes under H0 and 200 phenotypes under H1 so that in each simulation there are exactly 40 cases and 60 controls.
- For each simulation, we perform a single marker analysis with `PLINK` [3, 4], a standard software for doing GWAs. For each simulation, this results in a signal of association given by a vector of 1000 p-values (one for each SNP).
- For each simulation, in order to avoid multiple-testing issues, we resume the vector of p-values with two alternative single valued real statistics S_1, S_2. As a result, for each statistics S_1, S_2 we have 200 values under $H0$ and 200 values under $H1$ (these results are included in the package for the readers's who don't have `PLINK`) .
- For each method S_1, S_2, we assess the tradeoff between the true positive rate (i.e. the power or sensitivity) and the false positive rate (i.e. 1 - the specificity) by computing the corresponding ROC curve with the `R` package `pROC`.

The toy dataset included in the package is encoded in PED and MAP formats to be used with `PLINK` [3, 4]. Readers willing to use other softwares or packages for producing the association signals have to convert the data files into the suitable format. In the following, we include the complete commands for performing association analysis with `PLINK`; however we also included the results of `PLINK` analysis into four `.txt` files which come with the package, so that the readers who don't have `PLINK` installed can still reproduce our power study. We assume that the `R` package `pROC` is installed, otherwise simply run `install.packages("pROC")` from the `R` Console.

As a preliminary step, we store the data included in the package (which comes with the extension `.txt` as required by CRAN) into files with the extensions `.ped` and `.map` required by `PLINK`:

```
> library(waffect)
> data(map)
> write.table(map,file = "data.map", row.names=FALSE, col.names = FALSE)
> data(ped)
> write.table(ped,file = "data.ped", row.names=FALSE, col.names = FALSE)
```

The genotypes contained in the PED file were obtained from public data released by the *1000 Genomes Projects* [1], however in creating the PED files we assigned dummy values to the covariates not needed for the present analysis. We adopt the `PLINK` standard: **cases are coded by 2 and controls by 1**. `data.map` has one row for each SNP and four columns:

`chr`: the chromose, here all the SNPs are in the chromosome X
`SNP_id`: the SNP identifier, here an integer from 1 to 1000
`dist`: the genetic distance, here set to 0 for all SNPs
`bp_pos`: the base-pair position, here an integer from 1 to 1000.

`data.ped` has one row for each individual and 2006 columns:

`fam_id`: family ID, here an integer from 1 to 100
`ind_id`: individual ID, an integer from 1 to 100
`pat_id`: paternal ID, here a dummy variable
`mat_id`: maternal ID, dummy variable
`sex`: all females
`pheno`: observed phenotypes; codes: case = 2, control = 1.

Genotypes (column 7 onwards) are biallelic, one allele for each column. For instance, columns 7 and 8 contain the two alleles for SNP1 and columns 9 and 10 contain the alleles for SNP2.

We arbitrarily define an additive disease model H0 with SNP number 500 as disease SNP. Additive means that the relative risk grows linearly with the number of rare alleles of the disease SNP[2]. We assume that the additive effect is $\beta = 0.5$ and the baseline penetrance of the disease is $f_0 = 0.1$ (since the susceptibility variants are assumed to be rare, f_0 is approximately equal to the prevalence of the disease). Therefore the probability that individual j is a case is

$$\pi_j = f_0 \cdot RR_j = f_0 \cdot (1.0 + \beta \cdot X_j^{500})$$

where X_j^{500} is the number of rare alleles in the genotype of the SNP number 500 for individual j. RR_j is the Relative Risk for individual j. The rare allele for the SNP number 500 is `T` as can be easily checked:

```
> ped <- read.table("data.ped")
> x <- ped[,c(6+500*2-1,6+500*2)]
> length(which( x == "T")) < length(which( x == "A"))

[1] TRUE
```

We compute the vector of probabilities $(\pi_j)_{j=1,\ldots,100}$ as follows:

```
> ad <- 0.5
> RR <- rep(NA, 100)
> RR[x[,1] == "A" & x[,1] == "A"] <- 1.0 + ad * 0
> RR[x[,1] != x[,2]] <- 1.0 + ad * 1
> RR[x[,1] == "T" & x[,2] == "T"] <- 1.0 + ad * 2
> f0 <- 0.1
> pi = f0*RR
```

Now that we have the probabilities π_j, we can simulate phenotypes under H1 using `waffect` . In particular, we generate 200 phenotypic datasets and put the results in a 100×200 table:

[2]The choice of the disease model is completely unconstrained (number of SNPs involved, epistasis, Gene-Environment interactions, hybrid genetic models...).

```
> Nsim <- 200
> pheno_H1 <- matrix(NA, nrow = 100, ncol = Nsim)
> for(i in 1:Nsim)
+         pheno_H1[,i] <- waffect(prob = pi, count=40, label=c(2, 1))
```

We store the results in a file in the format required by PLINK for alternate phenotypes: the first two columns must be the familiy ID (in our case just an integer from 1 to 100) and the individual ID:

```
> pheno_H1 <- cbind(1:100,1:100,pheno_H1)
> write.table(pheno_H1, file = "pheno_H1.txt", row.names = FALSE, col.names = FALSE)
```

Similarly we use `waffect` to simulate 200 phenotypic datasets under H0:

```
> pheno_H0 <- matrix(NA, nrow = 100, ncol = Nsim)
> for(i in 1:Nsim)
+         pheno_H0[,i] <- waffect(count = c(40,60), label = c(2,1))
> pheno_H0 <- cbind(1:100,1:100,pheno_H0)
> write.table(pheno_H0, file = "pheno_H0.txt", row.names = FALSE, col.names = FALSE)
```

We can now compute the signal of association for each simulation.

`PLINK` **available.** By default we assumed that PLINK is not installed on the readers' machines and therefore we commented out the following `R` commands. Please delete all the `#` in order to run the code below.

We will store the results obtained with PLINK in two folders:

```
> #system("mkdir H1_signal H0_signal")
```

We run PLINK from `R` as follows :

```
> #system("plink --file data --model --pheno pheno_H1.txt --all-pheno --out H1_signal/H1")
> #system("plink --file data --model --pheno pheno_H0.txt --all-pheno --out H0_signal/H0")
```

Because we have specified the `--model` option, PLINK performs four statistical tests for each SNP. For example the results for SNP number 1 in a given simulation looks like

```
  CHR SNP A1 A2    TEST      AFF    UNAFF  CHISQ DF      P
1   0   1  T  A    GENO 16/6/18 20/11/29 0.5099  2 0.7750
2   0   1  T  A   TREND   38/42    51/69 0.2934  1 0.5880
3   0   1  T  A ALLELIC   38/42    51/69 0.4859  1 0.4858
4   0   1  T  A     DOM   22/18    31/29 0.1071  1 0.7435
5   0   1  T  A     REC   16/24    20/40 0.4630  1 0.4962
6   0   2  T  A    GENO 17/6/17 20/12/28 0.9710  2 0.6154
```

We are only interested in the p-values $p_1, \ldots, p_{1000}$ of the Cochran-Armitage trend test, a test which is often used when the disease model is expected to be additive. The first statistic S_1 we consider simply takes the smallest p-value among the the p-values of all the SNPs. Equivalently

$$S_1 := \max_{j=1,\ldots,1000} -\log_{10} p_j$$

We compute S_1 values (one for each simulation under H1 and H0) by running the following code (this will take a few seconds):

```
> #p1_H1 <- rep(NA,Nsim)
> #for(i in 1:Nsim){
> #        FileIn <- paste("H1_signal/H1.P", i , ".model", sep="")
> #        aux <- read.table(FileIn, header = TRUE)
> #        aux <- aux[which(aux$TEST == "TREND"),]
> #        p1_H1[i] <- max(-log10(na.omit(aux$P)))
> #}
>
> #p1_H0 <- rep(NA,Nsim)
> #for(i in 1:Nsim){
> #        FileIn <- paste("H0_signal/H0.P", i , ".model", sep="")
> #        aux <- read.table(FileIn, header = TRUE)
> #        aux <- aux[which(aux$TEST == 'TREND'),]
> #        p1_H0[i] <- max(-log10(na.omit(aux$P)))
> #}
```

`PLINK` **not available.** We performed `PLINK` analysis for both the two simulations under H_0 and H_1 (400 simulated datasets in total) and stored the results into the files `p1_H0.txt` and `p1_H1.txt` which come with the package. The readers who don't have `PLINK` can just load the two vectors of p-values obtained with `PLINK`:

```
> data(p1_H1)
> p1_H1 <- p1_H1$signal
> data(p1_H0)
> p1_H0 <- p1_H0$signal
```

In order to measure the performance of S_1, it is convenient to study its Receiver Operator Characteristic (ROC) curve which is nothing but a graphical representation of the sensitivity for all possible values of the specificity. The ROC curve itself can be further summarized by the Area Under the Curve (AUC) which can be qualitatively interpreted as follows: AUC ≤ 0.6 means "fail"; $0.6 <$ AUC ≤ 0.70 means "poor"; $0.7 <$ AUC ≤ 0.80 means "fair"; $0.8 <$ AUC ≤ 0.9 means "good"; $0.9 <$ AUC ≤ 1.0 means "excellent".

Now that the we have for each simulation a value of S_1, it is straightforward to compute the AUC and its confidence interval using the package `pROC` [5]:

```
> library(pROC)
> roc(controls = p1_H0, cases = p1_H1, ci=TRUE, plot = TRUE)

Call:
roc.default(controls = p1_H0, cases = p1_H1, ci = TRUE, plot = TRUE)

Data: 200 controls < 200 cases.
Area under the curve: 0.5321
95% CI: 0.4754-0.5888 (DeLong)
```

The ROC curve is depicted in Figure 1. The AUC is 0.53 with $[0.48, 0.59]$ as 95% confidence interval, therefore S_1 is not successful in detecting a positive signal.

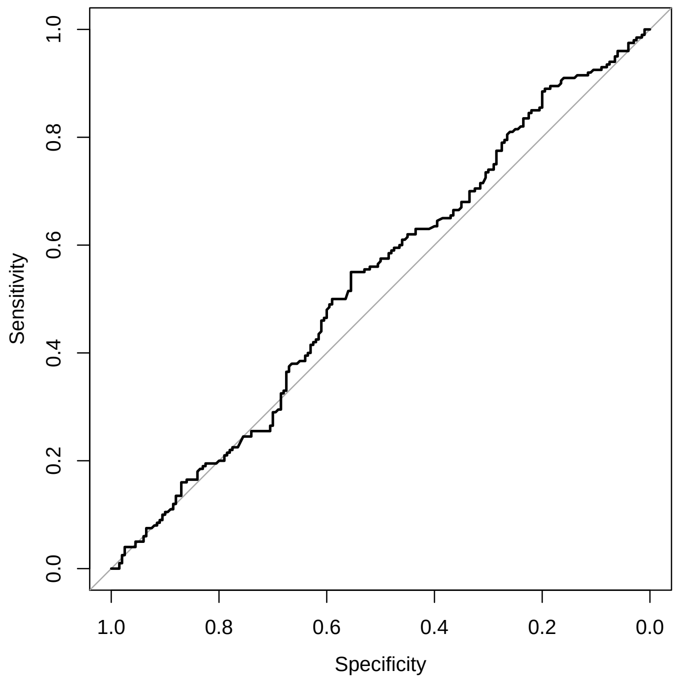

Figure 1: ROC curve for the statistics S_1.

If we have a a biological *a priori* we can try to narrow the investigation to the candidate gene level. In this perspective, the second statistic S_2 takes the smallest p-value in an interval centered around the disease SNP whose length is 5 SNPs[3]:

$$S_2 := \max_{498 \leq j \leq 502} \log_{10} p_j.$$

`PLINK` **available.** Please delete all the `#` in order to run the code below.

```
> #region <- 498:502
>
> #p2_H0 <- rep(NA,Nsim)
> #for(i in 1:Nsim){
> #       FileIn <- paste("H0_signal/H0.P", i, ".model", sep = "")
> #       aux <- read.table(FileIn, header = TRUE)
> #       aux <- aux[which(aux$TEST == "TREND"),]
> #       aux <- aux[region,]
> #       p2_H0[i] <- max(-log10(na.omit(aux$P)))
> #}
>
> #p2_H1 <- rep(NA,Nsim)
```

[3]Assuming that there is a SNP every 2kb, the length of an interval with 5 consecutive SNPs is compatible with the average size of genes.

```
> #for(i in 1:Nsim){
> #         FileIn <- paste("H1_signal/H1.P", i, ".model", sep="")
> #         aux <- read.table(FileIn, header=TRUE)
> #         aux <- aux[which(aux$TEST == "TREND"),]
> #         aux <- aux[region,]
> #         p2_H1[i] <- max(-log10(na.omit(aux$P)))
> #}
```

`PLINK` **not available.** We performed `PLINK` analysis for both the simulations under H_0 and H_1 and then stored the results into the files `p2_H0.txt` and `p2_H1.txt` which come with the package.

```
> data(p2_H1)
> p2_H1 <- p2_H1$signal
> data(p2_H0)
> p2_H0 <- p2_H0$signal
```

We are now ready to compute the ROC curve for S_2:

```
> roc(controls = p2_H0, cases = p2_H1, plot = TRUE, ci=TRUE)

Call:
roc.default(controls = p2_H0, cases = p2_H1, ci = TRUE, plot = TRUE)

Data: 200 controls < 200 cases.
Area under the curve: 0.7215
95% CI: 0.6712-0.7718 (DeLong)
```

The AUC for S_2 is 0.72 with $[0.67, 0.77]$ as 95% confidence interval, the ROC curve is depicted in Figure 2. We conclude that, as expected, S_2 is more performant than S_1, however the overall performance is still poor.

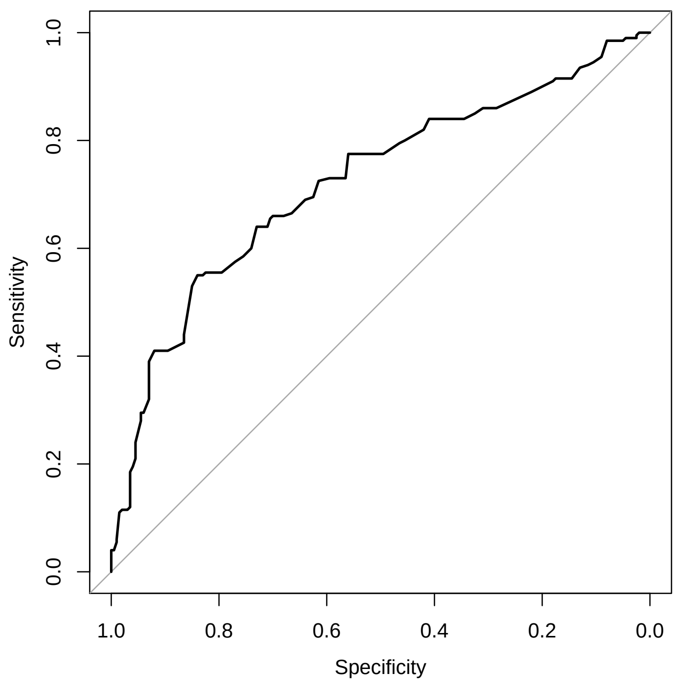

Figure 2: ROC curve for the statistics S_2.

References

[1] R. Durbin, D. Altshuler, G. Abecasis, D. Bentley, A. Chakravarti, A. Clark, F. Collins, M. Francisco, P. Donnelly, M. Egholm, et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[2] V. Perduca, R. Mourad, C. Sinoquet, and G. Nuel. Alternative methods for H1 simulations in genome wide association studies. *Human Heredity*, 73(2):95–104, 2012.

[3] S. Purcell. PLINK 1.07. http://pngu.mgh.harvard.edu/purcell/plink/.

[4] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. Ferreira, D. Bender, J. Maller, P. Sklar, P. de Bakker, M. Daly, and P. Sham. PLINK: a toolset for whole-genome association and population-based linkage analysis. *American Journal of Human Genetics*, page 81, 2007.

[5] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12:77, 2011.