# Haplotypes

**Thomas Lumley**
**Ken Rice**

UW Biostatistics

*Seattle, June 2009*

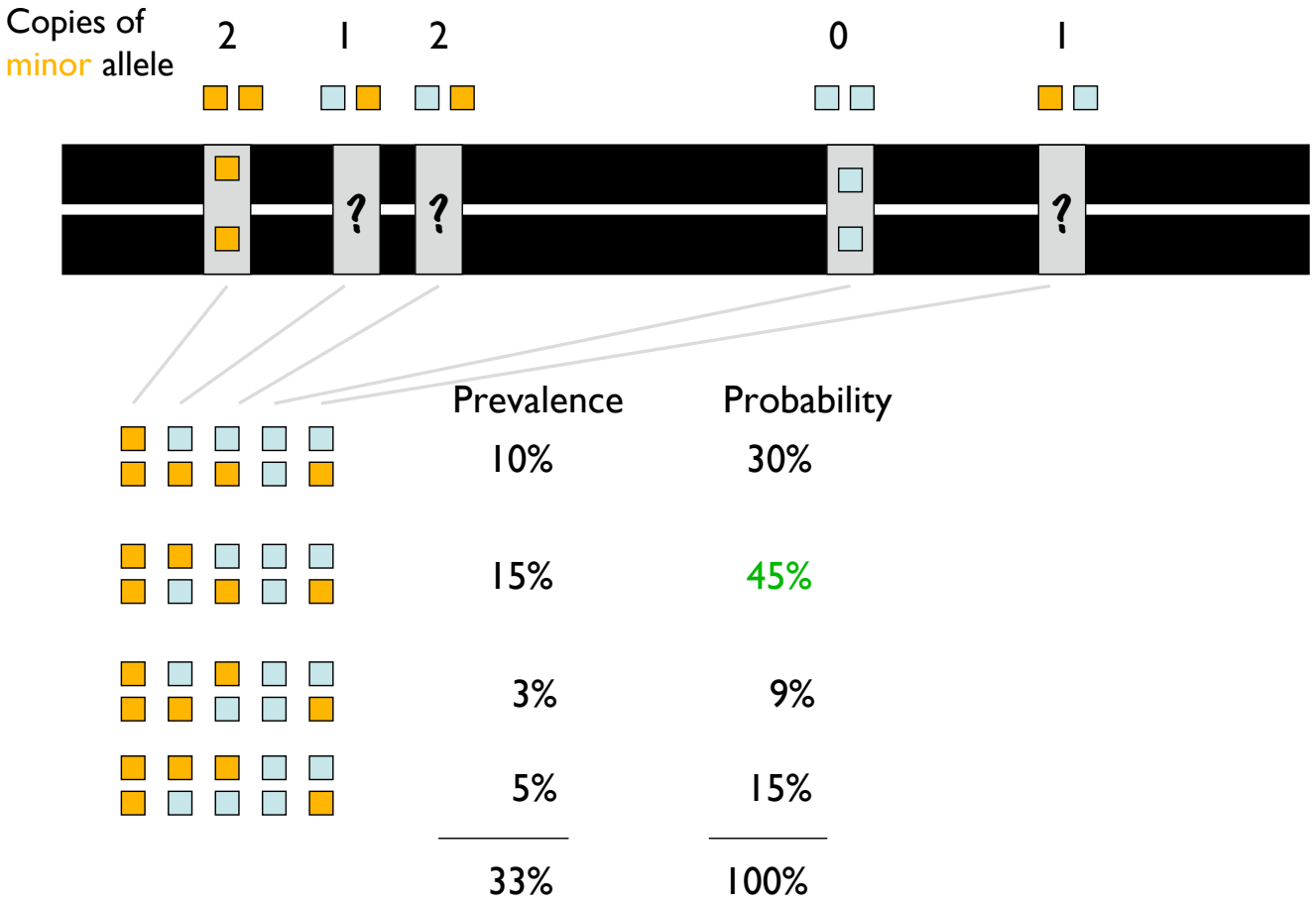
# SNPs and diplotypes

Current technologies allow inexpensive measurements of the number of copies of each allele of a SNP, but not direct measurements of which copy of the chromosome carries each allele.

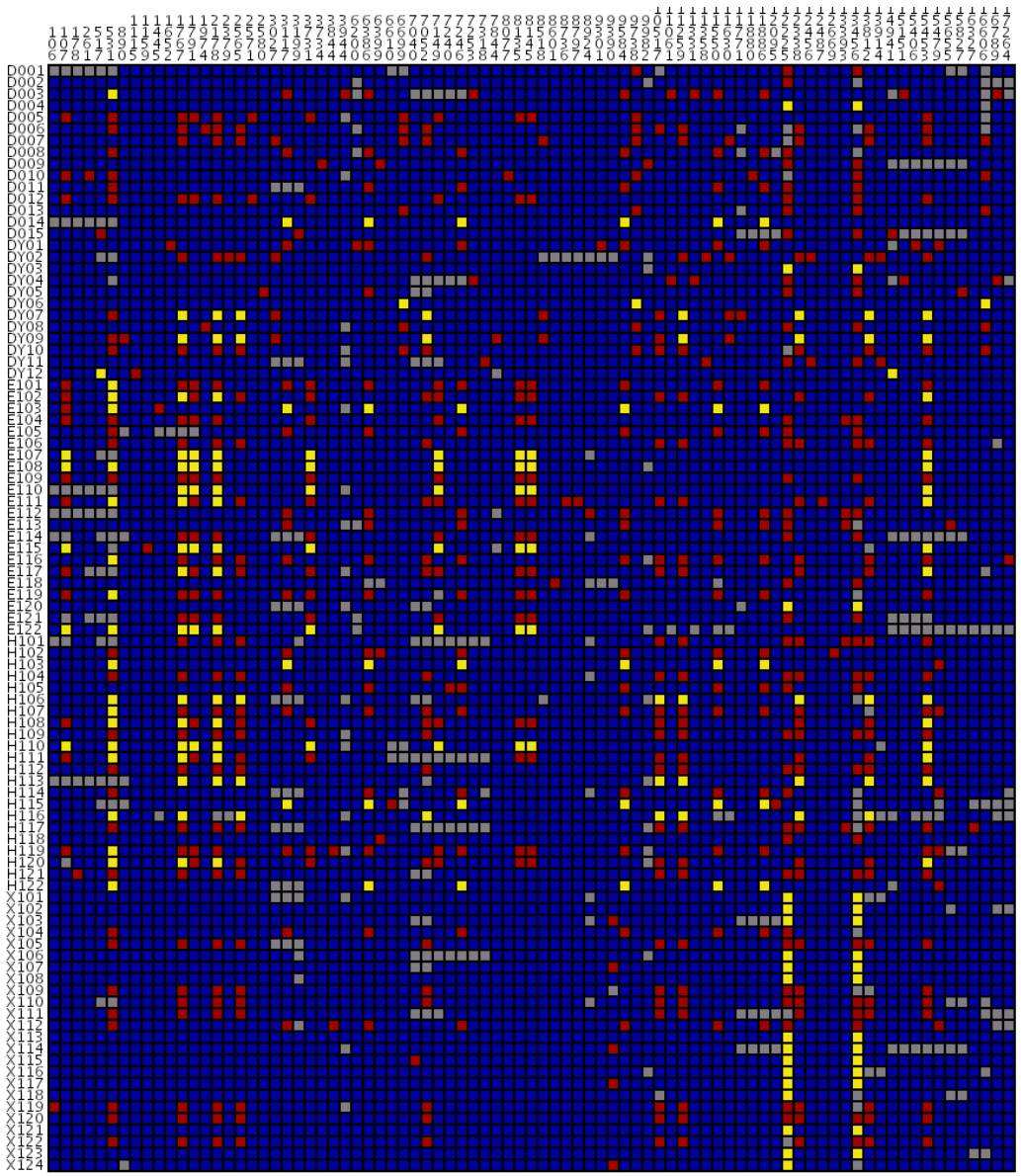
For an individual who is heterozygous at $k$ loci there are $2^{k-1}$ possible arrangements of the SNPs across the two copies of the gene.

Different arrangements of the SNPs along a chromosome (haplotypes) are not equally common. Usually only a few of the $2^{k-1}$ possibilities have non-negligible probability.

# SNPs and diplotypes



# SNPs and diplotypes



VG2 of: adh1a.prettybase.txt

# Haplotypes as predictors

Three reasons for interest in haplotypes

- Natural summary of multiple SNPs
- Effect of SNPs may depend on whether they are on the same copy
- Haplotype may be a better marker for untyped polymorphisms

Jury still out on when haplotypes are useful.

# Inferring haplotypes

Basic idea: compute all possible haplotype pairs for each individual and assign a probability weight to each one. Fit your favorite model using these weights.

Probabilities for haplotypes depend on outcome and regression coefficients ($\beta$): a poor outcome makes a high-risk haplotype more likely. The link is weaker for common haplotypes, small $\beta$.

Approaches:

- Joint estimation of $\beta$, haplotypes
- Haplotype probabilities estimated at $\beta = 0$
- Most probable haplotype imputed at $\beta = 0$.

For realistic effects of common haplotypes even crude methods work well.

# CRAN task views

CRAN

Mirrors
What's new?
Task Views
Search

About R

R Homepage

Software

R Sources
R Binaries
Packages
Other

## CRAN Task Views

| | |
|---|---|
| Bayesian | Bayesian Inference |
| Cluster | Cluster Analysis & Finite Mixture Models |
| Econometrics | Computational Econometrics |
| Environmetrics | Analysis of ecological and environmental data |
| Finance | Empirical Finance |
| Genetics | Statistical Genetics |
| Graphics | Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization |
| gR | gRaphical models in R |
| MachineLearning | Machine Learning & Statistical Learning |
| Multivariate | Multivariate Statistics |
| SocialSciences | Statistics for the Social Sciences |
| Spatial | Analysis of Spatial Data |

To automatically install these views, the ctv package needs to be installed, e.g., via
`install.packages("ctv")`
and then the views can be installed via `install.views` (after loading ctv), e.g.,
`install.views("Econometrics")`

Task views provide an annotated summmary of R packages on a topic and a mechanism for downloading the relevant packages with a single command.

# Genetics task view

Some relevant packages:

**LDheatmap:** Linkage-disequilibrium heatmap

**gap:** miscellaneous utilities for family and population data

**tdthap, powerpkg:** analysis and sample size for TDT studies

**hapassoc, haplo.stats:** generalized linear models for haplotype effects

**haplo.ccs:** haplotype effects in case-control studies (soon to have Cox model and case-cohort analysis as well).

# Data input

SNP data are naturally coded 0/1/2 for the number of copies of the minor allele. `hapassoc` is designed for SNP data and uses this format

For more general marker data we need two data columns for each marker. `haplo.stats` and `haplo.ccs` use this format.

# Data input

```
genexpand<-function(snpcounts, coding=NULL){
    p <- ncol(snpcounts)
    if (is.null(coding)) coding<-cbind(rep(1,p),rep(2,p))
    m<-matrix(ncol=2*p, nrow=nrow(snpcounts))
    for(i in 1:p){
       m[,2*i-1] <- coding[i,1+(snpcounts[,i]>0)]
       m[,2*i]  <- coding[i,1+(snpcounts[,i]>1)]
    }
    nms <- colnames(snpcounts)
    rownames(m)<-rownames(snpcounts)
    if (!is.null(nms))
      colnames(m)<-as.vector(t(outer(nms,c(1,2),paste,sep="_")))
    m
}
```

# Data input

```
> gg
     A B C D
[1,] 1 0 2 0
[2,] 2 1 2 0
[3,] 0 2 1 1
> genexpand(gg)
     A_1 A_2 B_1 B_2 C_1 C_2 D_1 D_2
[1,]   2   1   1   1   2   2   1   1
[2,]   2   2   2   1   2   2   1   1
[3,]   1   1   2   2   2   1   2   1

> genexpand(gg, coding=cbind(c("A","A","T","G"),
                             c("T","C","G","C")))
     A_1 A_2 B_1 B_2 C_1 C_2 D_1 D_2
[1,] "T" "A" "A" "A" "G" "G" "G" "G"
[2,] "T" "T" "C" "A" "G" "G" "G" "G"
[3,] "A" "A" "C" "C" "G" "T" "C" "G"
```

# haplo.ccs

Use `haplo()` function to wrap the genotype information. Otherwise similar to `glm()`. `haplo()` also specifies mode of inheritance, threshold for pooling rare haplotypes.

Example data: 330 cases, 3:1 controls, data simulated from haplotype frequencies for renin, involved in blood pressure control.

```
> library(haplo.ccs)
> data(renin)
> summary(haplo.ccs(case ~ age + factor(race) + gender*haplo(geno)))

Formula: case ~ age + factor(race) + gender * haplo(geno)
Estimates:
              Relative Risk Robust SE  t Value P(T>|t|)
223144 (Ref)         0.0012    0.5220 -12.8416   0.0000
212124               0.5605    0.2842  -2.0372   0.0418
222144               0.4553    0.6136  -1.2823   0.1999
222221               1.5762    0.2450   1.8570   0.0635
223124               1.3360    0.2292   1.2639   0.2065
323121               0.6466    0.3006  -1.4505   0.1472
```

# haplo.ccs

```
age                 1.1042    0.0082  12.0723   0.0000
factor(race)2       1.2470    0.1698   1.3003   0.1937
factor(race)3       0.5832    0.2188  -2.4640   0.0139
factor(race)4       1.0109    0.2658   0.0408   0.9675
gender              0.9782    0.3475  -0.0633   0.9495
212124:gender       0.3017    0.4125  -2.9046   0.0037
222144:gender       4.0507    0.7154   1.9554   0.0508
222221:gender       1.5114    0.3173   1.3018   0.1932
223124:gender       1.6839    0.2926   1.7812   0.0751
323121:gender       2.0163    0.3577   1.9607   0.0501

Haplotypes:
             Frequency
223144 (Ref)    0.3295
212124          0.1549
222144          0.0258
222221          0.1364
223124          0.2095
323121          0.1439

Number of Fisher Scoring Iterations: 5
```

# GLMs

`hapassoc` and `haplo.stats` both fit any generalized linear model to cohort or cross-sectional data. They both jointly estimate the haplotype probabilities and the regression parameters, giving maximum likelihood estimates.

Both require some preprocessing of the data.

`hapassoc` with $n$ observations and $k$ SNPs needs to allocate a $n2^{k-1} \times 2^{k-1}$ matrix, which is not possible in R for, eg, $n = 1000$, $k = 11$.

We will use `haplo.glm` on an example modified from real data on blood pressure and an anonymized gene involved in blood pressure control (some SNPs dropped, some bases relabelled).

# GLMs

The first five of 1000 observations look like

```
> head(bpdata)
     sex sbp dbp snp1 snp2 snp3 snp4 snp5 snp6 snp7 snp8 snp9 snp10 snp11 bmi
1 FEMALE 171  89   CC   TT   TT   TT   CC   GG   AA   TT   TT    CC    TT  25
2   MALE 160  99   TT   TT   CC <NA>   CC   AG   AT   CC   CT    CC    CT  35
3 FEMALE 142  83   CT   TT   TC   CT   CC   AG   TT   CC   TT    CT    TT  34
4   MALE 126  71   CT   TT   CC <NA>   CC   AA   TT   CC   TT    CT    CT  32
5 FEMALE 126  82   CT   TT   CC   CC   CC   AA   TT   CC   TT    CT    CT  34
```

First we need to extract the genotype columns and convert them to pairs of single-letter columns

```
snpsplit<-function(v) cbind(substr(v,1,1),substr(v,2,2))
bpsnps<-do.call(cbind, lapply(bpdata[,4:14], snpsplit))
> head(bpsnps)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,] "C"  "C"  "T"  "T"  "T"  "T"  "T"  "T"  "C"  "C"   "G"   "G"   "A"   "A"
[2,] "T"  "T"  "T"  "T"  "C"  "C"  NA   NA   "C"  "C"   "A"   "G"   "A"   "T"
[3,] "C"  "T"  "T"  "T"  "T"  "C"  "C"  "T"  "C"  "C"   "A"   "G"   "T"   "T"
[4,] "C"  "T"  "T"  "T"  "C"  "C"  NA   NA   "C"  "C"   "A"   "A"   "T"   "T"
[5,] "C"  "T"  "T"  "T"  "C"  "C"  "C"  "C"  "C"  "C"   "A"   "A"   "T"   "T"
```

# GLMs

The function `setupGeno()` annotates this matrix for use by `haplo.glm`

```
bpgeno<-setupGeno(bpsnps)
```

and then fit the model

```
haplo.glm(formula = sbp ~ bpgeno + dbp + sex,
     data = bpdata,
     allele.lev = attr(bpgeno, "unique.alleles"),
     control = haplo.glm.control(haplo.freq.min = 0.025,
     haplo.min.info = 0.01,
     em.c = haplo.em.control(min.posterior = 0.001)))
```

# GLMs

The `haplo.freq.min` argument says to combine all haplotypes with lower frequencies that 2.5%, the `min.posterior` option says to ignore haplotypes that have less that 0.1% probability for all individuals, and the `allele.lev` overcomes some differences between R and S-PLUS.

Without the restrictions on rare haplotypes the model fit fails to converge after about 40 minutes. With the restrictions it takes about two minutes.

# GLMs

```
  Call:
haplo.glm(formula = sbp ~ bpgeno + dbp +
    sex, data = bpdata, allele.lev = attr(bpgeno,
    "unique.alleles"), control = haplo.glm.control(haplo.freq.min = 0.025,
    haplo.min.info = 0.01, em.c = haplo.em.control(min.posterior = 0.001)))

Coefficients:
              coef     se  t.stat     pval
(Intercept) 81.606 0.1113 732.942 0.00e+00
bpgeno.15   -1.240 0.3289  -3.770 1.73e-04
bpgeno.17    0.883 1.4521   0.608 5.43e-01
bpgeno.30    2.965 0.7936   3.736 1.98e-04
bpgeno.31    0.654 0.4479   1.461 1.44e-01
bpgeno.41    3.538 0.2422  14.609 0.00e+00
bpgeno.51    1.765 1.8523   0.953 3.41e-01
bpgeno.63    5.201 0.1365  38.096 0.00e+00
bpgeno.70    1.272 1.1149   1.141 2.54e-01
bpgeno.rare  0.585 1.2800   0.457 6.48e-01
dbp          0.734 0.0152  48.424 0.00e+00
sexMALE     -5.015 1.2599  -3.980 7.38e-05

Haplotypes:
            loc.1 loc.2 loc.3 loc.4 loc.5 loc.6 loc.7 loc.8 loc.9 loc.10 loc.11
```

# GLMs

```
bpgeno.15       C     T     C     C     C     A     T     C     T     C     C
bpgeno.17       C     T     C     C     C     A     T     C     T     T     C
bpgeno.30       C     T     T     T     C     A     T     C     T     C     T
bpgeno.31       C     T     T     T     C     A     T     C     T     T     T
bpgeno.41       T     A     C     T     C     A     T     C     T     C     T
bpgeno.51       T     A     C     T     T     G     A     T     C     C     T
bpgeno.63       T     T     C     C     C     A     A     C     T     T     T
bpgeno.70       T     T     C     C     C     A     T     C     T     T     T
bpgeno.rare     *     *     *     *     *     *     *     *     *     *     *
haplo.base      T     T     C     C     C     A     T     C     T     C     T
              hap.freq
bpgeno.15       0.0287
bpgeno.17       0.0894
bpgeno.30       0.0400
bpgeno.31       0.0292
bpgeno.41       0.0284
bpgeno.51       0.0504
bpgeno.63       0.0268
bpgeno.70       0.1867
bpgeno.rare     0.2965
haplo.base      0.2239
```

# GLMs

One component of the returned values gives all the haplotype pairs considered by the model and their probabilities estimated at $\beta = 0$ and at the fitted $\widehat{\beta}$

```
> nrow(bpmodel$haplo.post.info)
[1] 2945
> sum(bpmodel$haplo.post.info$post.init>0.75)
[1] 758
> sum(bpmodel$haplo.post.info$post.init>0.9)
[1] 548
```

So for most individuals there is one haplotype pair much more likely than any other.

We can also plot the probabilities at $\beta = 0$ and at the final $\widehat{\beta}$: there are some changes, but most are small.

# GLMs