Variation in recombination rate and its genetic determinism in Sheep populations.

Supplemental methods

Morgane Petit, Carole Moreno, Bertrand Servin

Contents

1 Introduction

This document is an `org-mode` file that contains all necessary information enabling to reproduce the results presented in the paper *Fine scale structure and genetic determinism of recombination rate in sheep from combining multiple genome-wide datasets* by Petit et al. (2017). It can be used in `emacs` to actually run the analyses, and can be exported as a pdf/rtf … file for documentation.

1.1 Data Sources

This study exploits genetic data from two sources: a pedigree dataset from the Lacaune Sheep population genotyped with a medium density SNP chip and a population dataset of 70 Lacaune males genotyped with a high density SNP chip .

The relevant files can be downloaded as follows:

```
mkdir -p data/population/HDpanel
mkdir -p data/family/Lacaunes
## Get Panel data
baseurl=https://www.zenodo.org/record/237116/files
wget -P data/population/HDpanel $baseurl/frenchsheep_HD.bim
wget -P data/population/HDpanel $baseurl/frenchsheep_HD.bed
wget -P data/population/HDpanel $baseurl/frenchsheep_HD.fam
## Get Pedigree data
baseurl=https://www.zenodo.org/record/804264/files
wget -P data/family/Lacaunes $baseurl/lacaune_genotypes.fam
wget -P data/family/Lacaunes $baseurl/lacaune_genotypes.bed
wget -P data/family/Lacaunes $baseurl/lacaune_genotypes.bim
```

1.2 Prerequisites

Required packages, software and libraries to run the analysis

- `R` environment
  - `qvalue`
  - `Hmisc`
  - `zoo`
  - `ashr`
  - `Hmisc`
  - `lme4`
  - `mclust`
  - `reshape2`
- `python` version 2.7
  - `numpy / scipy / matplotlib`
- Additional Software:
  - To recreate intermediate results: PHASE, LINKPHASE, blupf90, bimbam
  - gemma : `http://www.xzlab.org/software.html`

1.3 Structure

The main directory contains (i) intermediate files and scripts to reproduce the analysis and (ii) results from the paper //Fine scale structure and genetic determinism of recombination rate in sheep from combining multiple genome-wide datasets// by Petit et al. (2017). The directory structure is as follows:

`data/` Supplemental data for the analyses

`data/family` Contains intermediate results from `LINKPHASE` and `AIREML` analysis

`data/population` Contains intermediate results from `PHASE` `data/genome` Information from OAR3.1 genome assembly

`data/gwas` Contains intermediate results to run GWAS on GRR (imputed genotypes and input files for `gemma`).

`data/Soay` Data from Johnston et al. (2016) on Soay Sheep population.

`results/` Results from the analyses

`code/` scripts used to produce `results`. Some of these scripts are extracted (technically *tangled*) from this document, while others need to be run from the command line, as specified below.

`figures/` figures produced by the analyses

1.4 Base analyses

The results presented in this document exploit outputs from initial analyses based on publicly available data (see 1.1). Because these analyses require a substantial amount of computing time, we provide in the `data` directory the relevant output files. However the following scripts can be used to reproduce these files:

`code/linkphase.sh` converts plink files to input files for LINKPHASE, runs LINKPHASE on each chromosome, identifies and corrects for double-crossovers (see `code/doubleco.R`). Creates output files in `data/family/linkphase.in` directory.

`code/PHASE.sh` extract Lacaune individuals from French sheep HD panel data, removing individuals that are present in the cohort. Converts plink file to PHASE input files (using plink itself and `code/fph2ph.py`). An example is included on how to run PHASE on a genome region to get posterior distribution samples of historical recombination rates.

`code/selectFIDs.R` R script to select FIDs as detailed in the manuscript. Creates the `data/family/FIDs.txt` file.

2 Results

2.1 High-Resolution Recombination Maps

2.1.1 Meiotic recombination maps

1. Constructing Recombination maps

   The construction of recombination maps from family data is done using the `python` script `code/family_map.py`. This script uses the output from LINKPHASE (in the `data/family/LINKPHASE/` directory) that contains crossover boundaries and run the estimation procedure described in Petit et al. (2017).

   ```
   python code/family_map.py
   ```

   This produces recombination maps in the `results/family/` directory, for windows of one megabases and all intervals of the SNP array. The result of prior calibration using the Kadri et al. approach for initial c_j estimates is provided in Figure SM1. It also produces a file in the `results/combined/` directory that contains, for each one megabase window, a sample of 20 values from the posterior distribution of c_j. This is used later for the comparison with population-based recombination maps.

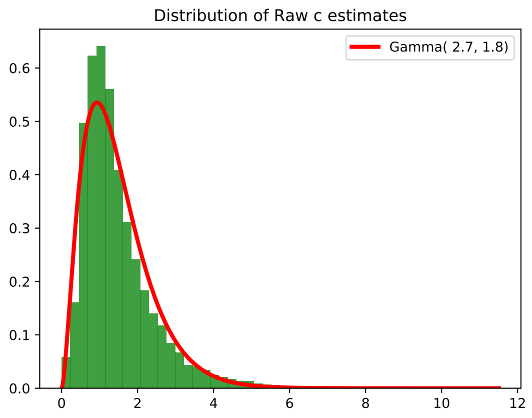


Figure SM1: Calibration of the prior distribution of $\mathrm{c_j}$ (cM/Mb)

We then produce a representation of recombination maps on each autosome, the resulting plots are given in file `figures/recombination_map.pdf`. As an illustration, recombination maps for chromosome 24 are shown on Figure SM2.

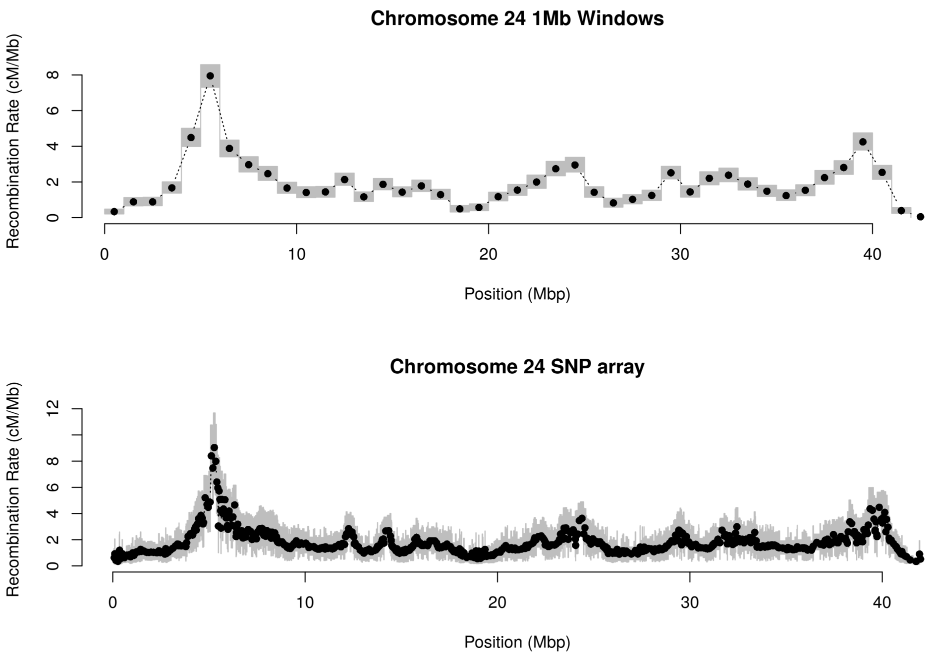


Figure SM2: Meiotic recombination map for chromosome 24

We can also pool chromosomes by rescaling distances among each chromosome to lie between 0 and 1. Looking specifically at the distribution of recombination rate estimates as a function of the physical distance (in Megabases) from the nearest end, reveals a potential bias in regions with 4 Megabases of a chromosome end. This is possibly due to undetected crossovers in these regions. Figure SM3 shows rescaled maps along with the recombination rate estimates as a function of the distance to a chromosome end.

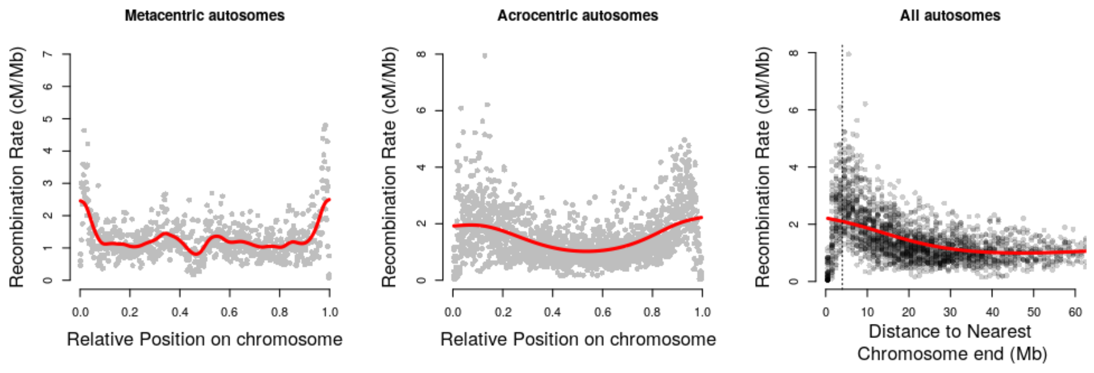


Figure SM3: Meiotic recombination rate variation along chromosomes

2. Statistical Analysis of Recombination maps

   (a) Effect of the number of meioses on recombination rate estimates

   As the number of offspring (*i.e.* observed meioses) varies a lot among individuals, we want to check wether this influences its mean recombination rate estimate, *i.e.* verify that there is no obvious bias due to family size in estimating recombination rates. To this end we fit a linear model on the effect of the number of meiosis on the average number of crossovers per meiosis:

```
library(xtable)
grr=read.table('results/family/parent_recombination.txt',head=T)
mod.nmeio=lm(Ri ~ nbMeio,data=grr)
res=summary(mod.nmeio)
coef(res)[2,c(1,4)]
```

0.015329808544272
0.0528827701092922

Estimate	Std. Error	t value	Pr(>|t|)
35.3011	0.2150	164.18	0.0000
0.0153	0.0079	1.94	0.0529

Table SM1: Estimating and testing the effect of the number of observed meioses on individual recombination rates.

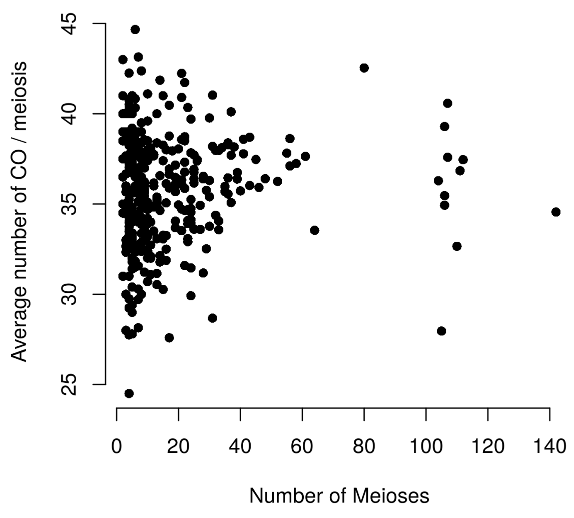


Figure SM4: The mean number of crossover of an individual is not related to its number of offspring.

This analysis reveals that the effect, if any (p=0.0529), is very small (1.5 more crossover / 100 meioses) (Table SM1, Figure SM4).

(b) Chromosome specific recombination rates

We estimate chromosome specific recombination rates, and fitted two relationships between the physical size of a chromosome and its recombination rate estimate: `mod.size.log` fits the recombination rate as a function of log(size) and `mod.size.inv` as a function of 1/size.

First, we perform the analysis on the 1 megabase recombination map, the resulting chromosome recombination rates are given in Table SM2 and Figure SM5.

```
library(Hmisc)
library(xtable)
### CHROMOSOME REC RATE 1 Mb WINDOWS
mod=lm(m_cj~chr-1,data=rec.1m,weights=1/s_cj^2)
res=summary(mod)
rec=res$coefficients[,1]
rec.std.err=res$coefficients[,2]
err=confint(mod)
mod.size.log=lm(rec~log(chr.size),weights=1/rec.std.err^2)
mod.size.inv=lm(rec~I(1/chr.size),weights=1/rec.std.err^2)
mod.size.log.b=coefficients(mod.size.log)
mod.size.inv.b=coefficients(mod.size.inv)

size2rec.log=function(s) {
    mod.size.log.b[1]+mod.size.log.b[2]*log(s)
}
size2rec.inv=function(s) {
    mod.size.inv.b[1]+mod.size.inv.b[2]/s
}
```

Estimate	Std. Error	t value	Pr(>|t|)
1.0100	0.0343	29.43	0.0000
0.9634	0.0354	27.22	0.0000
1.1147	0.0404	27.62	0.0000
1.1265	0.0561	20.09	0.0000
1.1241	0.0597	18.82	0.0000
1.0788	0.0561	19.23	0.0000
1.1375	0.0618	18.40	0.0000
1.0956	0.0639	17.13	0.0000
1.0218	0.0613	16.67	0.0000
0.7468	0.0544	13.73	0.0000
1.4740	0.0936	15.75	0.0000
1.1406	0.0713	16.00	0.0000
1.2048	0.0716	16.83	0.0000
1.3549	0.0891	15.21	0.0000
1.1193	0.0701	15.97	0.0000
1.0810	0.0723	14.96	0.0000
1.1847	0.0766	15.48	0.0000
1.1883	0.0796	14.93	0.0000
1.2628	0.0874	14.44	0.0000
1.0237	0.0869	11.79	0.0000
1.3054	0.1007	12.97	0.0000
1.2561	0.0962	13.06	0.0000
1.1127	0.0815	13.66	0.0000
1.5381	0.1206	12.76	0.0000
1.3660	0.1068	12.79	0.0000
1.5255	0.1147	13.30	0.0000

Table SM2: Chromosome effect model estimates

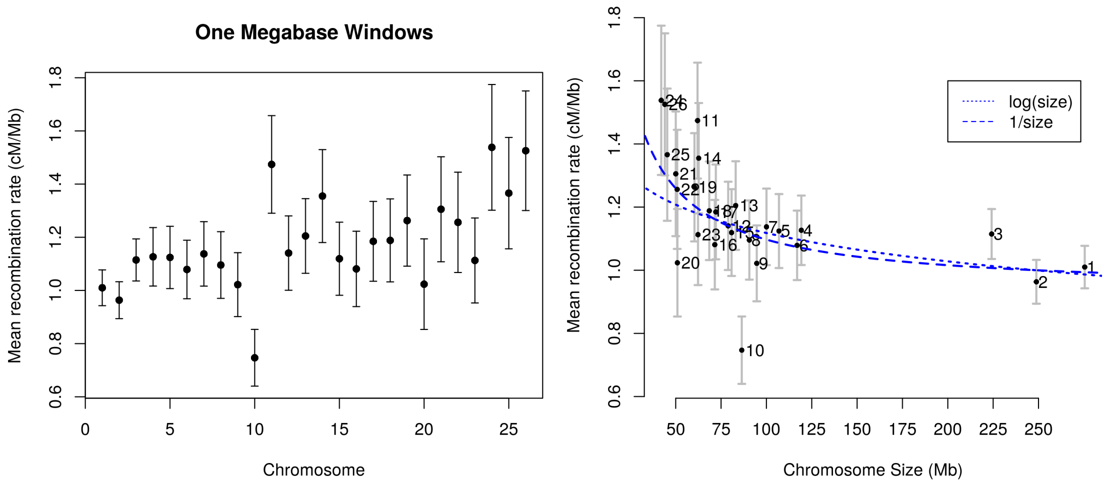


Figure SM5: Chromosome specific recombination rates (1Mb windows)

We can perform a similar analysis on SNP array intervals (Table SM3, Figure SM6).

```
### CHROMOSOME REC RATE SNP ARRAY
mod=lm(m_cj~chr-1,data=rec.snp,weights=1/s_cj^2)
res=summary(mod)
rec=res$coefficients[,1]
rec.std.err=res$coefficients[,2]
err=confint(mod)
mod.size.log=lm(rec~log(chr.size),weights=1/rec.std.err^2)
mod.size.inv=lm(rec~I(1/chr.size),weights=1/rec.std.err^2)

mod.size.log.b=coefficients(mod.size.log)
mod.size.inv.b=coefficients(mod.size.inv)
size2rec.log=function(s) { mod.size.log.b[1]+mod.size.log.b[2]*log(s) }
size2rec.inv=function(s) { mod.size.inv.b[1]+mod.size.inv.b[2]/s }
```

Estimate	Std. Error	t value	Pr(>|t|)
1.1759	0.0060	197.19	0.0000
1.1603	0.0062	188.31	0.0000
1.2489	0.0068	183.62	0.0000
1.2499	0.0094	132.72	0.0000
1.2647	0.0102	124.25	0.0000
1.2295	0.0095	129.78	0.0000
1.2662	0.0105	120.82	0.0000
1.2442	0.0109	114.47	0.0000
1.1932	0.0104	114.43	0.0000
1.0579	0.0103	102.58	0.0000
1.4772	0.0155	95.34	0.0000
1.2721	0.0121	105.13	0.0000
1.2929	0.0120	107.33	0.0000
1.4114	0.0152	93.00	0.0000
1.2537	0.0119	105.36	0.0000
1.2359	0.0125	98.78	0.0000
1.2964	0.0131	98.70	0.0000
1.3104	0.0134	97.79	0.0000
1.3357	0.0147	90.88	0.0000
1.2623	0.0154	81.96	0.0000
1.3398	0.0170	78.92	0.0000
1.3686	0.0161	85.11	0.0000
1.2340	0.0143	86.45	0.0000
1.5203	0.0203	74.75	0.0000
1.4177	0.0176	80.36	0.0000
1.4931	0.0190	78.72	0.0000

Table SM3: Chromosome effect model estimates (SNP array)

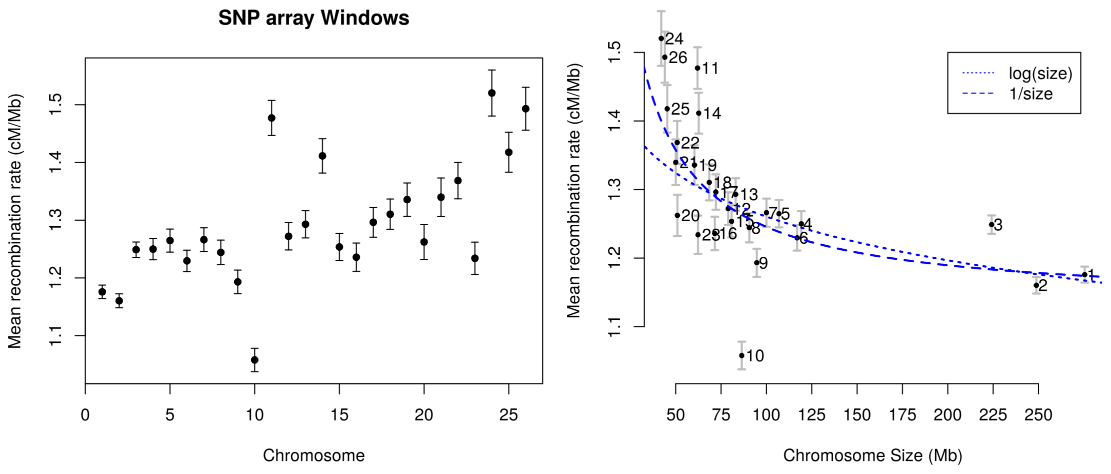


Figure SM6: Chromosome recombination rates estimates (SNP array)

(c) GC Content

GC content was extracted for each 1 Mb windows and each SNP array interval from the reference genome sequence OAR v3.1.

Results are provided in the `results/genome/GC_Content_*` files.

```
## One Mb windows
gc1=read.table('data/genome/GC_Content_1Mb.txt',head=TRUE)
## SNP array intervals
gcs=read.table('data/genome/GC_Content_SNP_array.txt',head=TRUE)
## check the data are aligned
require(assertthat)
assert_that(mean(gcs$left==dat.snp$left)==1)
assert_that(mean(gc1$left==dat.1m$left)==1)

gc1$cj=dat.1m$m_cj
gcs$cj=dat.snp$m_cj


## remove extreme regions
for (ichr in 1:26) {
    subset=(gc1$chr==ichr)
    subset= subset & ((gc1$left<4e6) | (gc1$right > (chr.size[ichr]-4e6)))
    gc1=gc1[!subset,]
    subset=(gcs$chr==ichr)
    subset=subset & ((gcs$left<4e6) | (gcs$right > (chr.size[ichr]-4e6)))
    gcs=gcs[!subset,]
}
```

A first look at the data shows that the relationship between GC content and recombination rate is not really linear (Figure SM7).

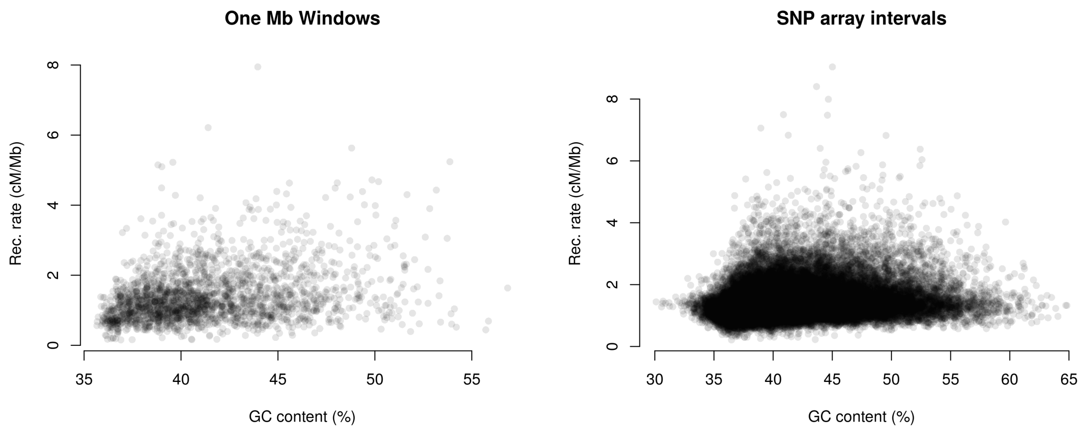


Figure SM7: Recombination rate and GC content – Raw scale

We can quantile transform GC content to get a much nicer linear relationship between GC and recombination rate (Figure SM8).

```
dist.gc1=ecdf(gc1$gc)
dist.gcs=ecdf(gcs$gc)
## quantiles of the gc distribution
gc1$qgc=dist.gc1(gc1$gc)
gcs$qgc=dist.gcs(gcs$gc)
```

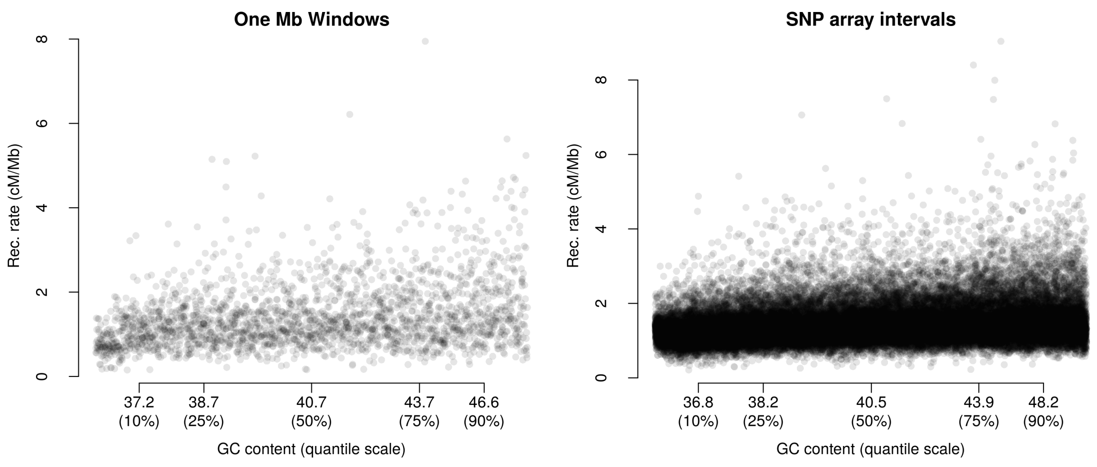


Figure SM8: Recombination rate and GC content – Quantile scale

We can fit linear models to test for the effect of GC content on recombination rate, adjusted for a chromosome effect.

```
## Model on 1 Mb windows, raw GC covariate:
gc1$chr=as.factor(gc1$chr)
lm.1m.chr=lm(cj~chr,data=gc1)
lm.1m.chr.gc=lm(cj~chr+gc,data=gc1)
aov.1m.gc=anova(lm.1m.chr,lm.1m.chr.gc)
pval.1m.gc=aov.1m.gc['Pr(>F)'][2,1]
```

```
## Model on 1Mb windows, quantile-transformed gc
lm.1m.chr.qgc=lm(cj~chr+qgc,data=gc1)
aov.1m.qgc=anova(lm.1m.chr,lm.1m.chr.qgc)
pval.1m.qgc=aov.1m.qgc['Pr(>F)'][2,1]

## Model on SNP windows, raw GC covariate:
gcs$chr=as.factor(gcs$chr)
lm.snp.chr=lm(cj~chr,data=gcs)
lm.snp.chr.gc=lm(cj~chr+gc,data=gcs)
aov.snp.gc=anova(lm.snp.chr,lm.snp.chr.gc)
pval.snp.gc=aov.snp.gc['Pr(>F)'][2,1]

## Model on SNP windows, quantile-transformed gc
lm.snp.chr.qgc=lm(cj~chr+qgc,data=gcs)
aov.snp.qgc=anova(lm.snp.chr,lm.snp.chr.qgc)
pval.snp.qgc=aov.snp.qgc['Pr(>F)'][2,1]
```

```
0
```

Intervals	GC content	$-\log_{10}(p)$
1 Mb	raw	23.95
1 Mb	transformed	26.07
SNP	raw	218.19
SNP	transformed	Inf

Table SM4: Significance of GC content effect on recombination rate

For all models considered, the effect of GC content on recombination rate is highly significant. We store the data in a new file, annotating the SNP and 1Mb pedigree maps.

```
write.table(gc1,file='results/family/1Mb_map_annotated.txt',
            col.names=T,row.names=F,quote=FALSE)
write.table(gcs,file='results/family/SNP_array_map_annotated.txt',
            col.names=T,row.names=F,quote=FALSE)
```

2.1.2 Population-based recombination maps

The PHASE output (in directory `results/population/PHASE`) contains the posterior distribution of LD-based recombination rates. PHASE was run on chromosome windows of 2.2 Mb, with an overlap of 100Kb between successive windows to avoid border effects. Each window has its own output file within its own sub-directory of the form (`CHRNUM/BEGIN-END`). From these posterior distribution, we first extract point estimates of LD-based recombination rates using the python script `code/make_pop_map.py`.

This script creates the `results/population/HD_SNP_array_map.txt` file.

```
python code/make_pop_map.py --dir data/population/PHASE/ --pad 1e5
```

From the PHASE output, we extract samples of the posterior distribution of recombination rates on one megabase windows along the genome. They will be used for the combination of meiotic and LD-based recombination estimates.

```
python code/get_ld_rec_samples.py
```

1. Identification of crossing over hotspots Based on the distribution of interval specific recombination intensities, we identified cross-over hotspots as follows.

   We fit a mixture of two Gaussian distribution to the genome-wide distribution of interval recombination intensities. The fit to the observed distribution is good, and the major component is interpreted as the distribution of non-hotposts intervals. We can thus for a given interval test the hypothesis that its intensity comes from the major component of the mixture, and get a corresponding p-value (Figure fig:call$_{\text{hostpots}}$).

   ```
   ## read in LD map
   rhomap = read.table('results/population/HD_SNP_array_map51.txt', head=T)
   ## model the distribution of recombination intensities lambda (log scale)
   ## as a mixture of normal (note the prior distribution is p(log(lambda)~N(.)))
   library(mclust)
   mc=Mclust(log10(rhomap$lambda),G=2)
   ## get parameters of the two components
   pars = mc$parameters
   xx=seq(-3,3,0.01)
   d1.pars=c(pars$mean[1],sqrt(pars$variance$scale[1]))
   d2.pars=c(pars$mean[2],sqrt(pars$variance$scale[2]))
   d1=dnorm(xx, mean = d1.pars[1], sd = d1.pars[2])
   d2=dnorm(xx, mean = d2.pars[1], sd = d2.pars[2])
   ## compute p-values corresponding to H0: interval in background distribution
   ## vs. H1: hotspot
   if (pars$pro[1]>pars$pro[2]) {
       null.pars=d1.pars
   } else {
       null.pars=d2.pars
   }
   pval=pnorm(log10(rhomap$lambda), mean=null.pars[1],
       sd=null.pars[2], lower.tail=F)
   ## Graphical representation of the fit
   par(mfrow=c(1,2))
   hist(log10(rhomap$lambda), n=100, freq=F,
        xlab=expression(log10(lambda[i])), main='')
   lines(xx, pars$pro[1]*d1, col=2, lwd=2)
   lines(xx, pars$pro[2]*d2, col=4, lwd=2)
   hist(pval,main='P-Value distribution',xlab='',freq=F,n=100)
   abline(h=1,lwd=2,lty=2,col='gray')
   ```

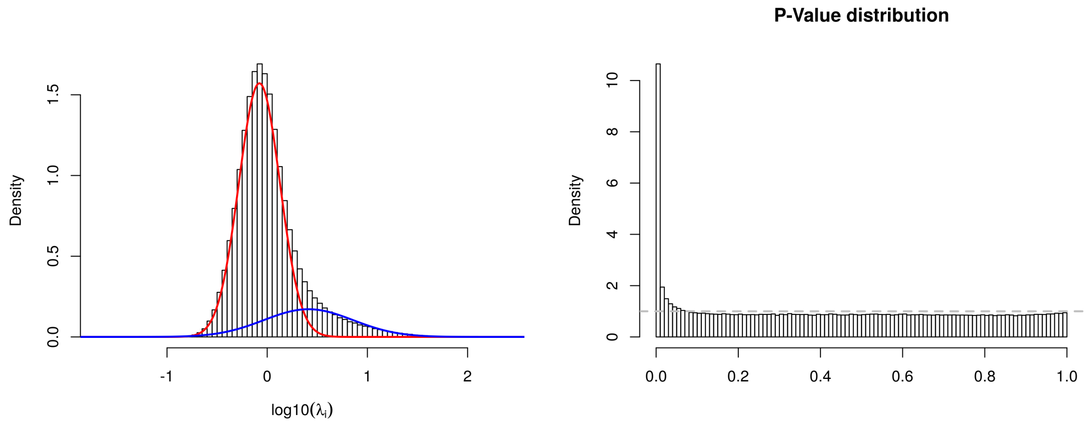


Figure SM9: Distribution of interval specific recombination intensities and clustering in hotspots / non hotspots

Based on the p-value distribution, we used the Storey and Tibshirani (2003) approach to estimate (i) the proportion of crossover hotspots on the sheep genome and (ii) call significant hotspot intervals at an FDR of 5%.

```
library(qvalue)
qval = qvalue(pval)
length(pval)
## Estimate of the number of hotspots in the sheep genome
nhs=length(pval)*(1-qval$pi0)
print(paste('Estimated # hotspots intervals:',nhs))
## call hotspots
rhomap$hotspot = qval$qvalues < 0.05
rhomap$qvalue = qval$qvalues
hsmap=subset(rhomap, select = c('chr', 'left', 'right', 'qvalue', 'hotspot' ))
write.table(hsmap,row.names=F,quote=F,file='results/population/hotspots51.txt')
```

To measure heterogeneity in the distribution of recombination, we compute a gini coefficient (Kaur and Rockman, 2014) based on the evolution of cumulated physical distance of intervals ordered by their genetic distance (Figure SM10).

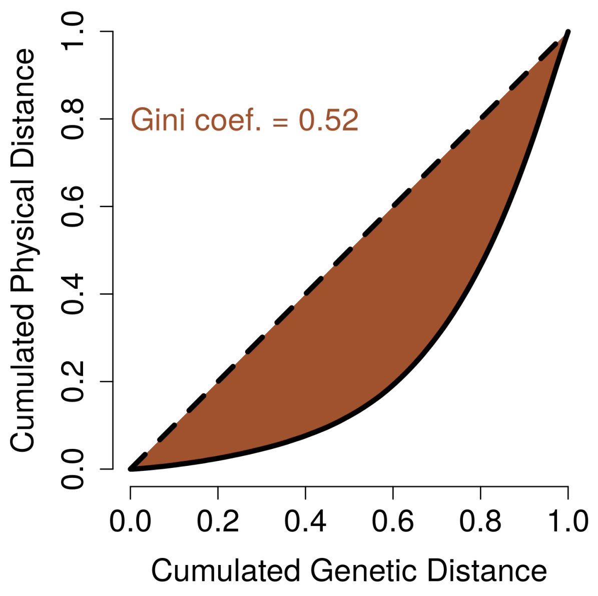


Figure SM10: Heterogeneity in recombination rate measured from the evolution of physical distance covered by the genetic map.

2.1.3 Effect of hotspot density on meiotic recombination rate

These commands must be ran after performing the LD-based analysis and detecting crossover hotspots.

First, we gather the number of hotspots within each interval on the one-Megabase and SNP array meiotic recombination maps.

```
hsmap=read.table('results/population/hotspots51.txt',h=T)



## Get number of hotspots in intervals
getnhs=function(v,hsdat=hsmap) {
    v=as.integer(v)
    hs.loc=hsdat[(hsdat[,1]==v[1])&(hsdat[,2]>=v[2])&(hsdat[,3]<=v[3]),]
    return(sum(hs.loc[,5]))
}

gc1$nhs=apply(gc1[,c(1,2,3)],1,getnhs)
gcs$nhs=apply(gcs[,c(1,2,3)],1,getnhs)

## calculate hotspot density (in HS / 10Kb) for SNP array map
gcs$len=gcs$right-gcs$left
gcs$hsdens=1e4*gcs$nhs/gcs$len
```

Correlations between meiotic recombination rates and historical hotspots on the SNP array map

```
csnp.nhs=cor.test(gcs$cj,gcs$nhs)
csnp.hsdens=cor.test(gcs$cj,gcs$hsdens)
```

```
c1.nhs=cor.test(gc1$cj,gc1$nhs)
pval.vec=c(c1.nhs$p.value,csnp.nhs$p.value,csnp.hsdens$p.value)
res.cor=data.frame("Intervals"=c('1 Mb','SNP','SNP'),
    "HS model"=c('number/density','number','density'),
    "corr."=c(c1.nhs$estimate,csnp.nhs$estimate,csnp.hsdens$estimate),
    "p-value"=-log10(pval.vec))

colnames(res.cor)[2]=c('HS effect')
colnames(res.cor)[3]=c('Correlation')
colnames(res.cor)[4]=c('$-\\log_{10}(p)$')

print(xtable(res.cor,
  caption='Correlation between historical hotspots and meiotic recombination rate.%
  Number:  number of hotspots.%
  Density:  density of hotspots (in HS/10Kb).'),
  type='latex',include.rownames=FALSE,sanitize.text.function=function(x){x})
```

Intervals	HS effect	Correlation	$-\log_{10}(p)$
1 Mb	number/density	0.46	115.30
SNP	number	0.19	Inf
SNP	density	0.15	220.53

Table SM5: Correlation between historical hotspots and meiotic recombination rate.Number: number of hotspots.Density: density of hotspots (in HS/10Kb).

Then we add as a new covariate the number of hotspots to our linear models on meiotic recombination rates.

```
lm.1m.chr.qgc.nhs=lm(cj ~ chr + qgc + nhs, data=gc1)
aov.1m.qgc.nhs=anova(lm.1m.chr.qgc,lm.1m.chr.qgc.nhs)
pval.1m.qgc.nhs=aov.1m.qgc.nhs['Pr(>F)'][2,1]

lm.snp.chr.qgc.nhs=lm(cj ~ chr + qgc + nhs, data=gcs)
aov.snp.qgc.nhs=anova(lm.snp.chr.qgc,lm.snp.chr.qgc.nhs)
pval.snp.qgc.nhs=aov.snp.qgc.nhs['Pr(>F)'][2,1]


lm.snp.chr.qgc.hsdens=lm(cj ~ chr + qgc + hsdens, data=gcs)
aov.snp.qgc.hsdens=anova(lm.snp.chr.qgc,lm.snp.chr.qgc.hsdens)
pval.snp.qgc.hsdens=aov.snp.qgc.hsdens['Pr(>F)'][2,1]

res.hs=data.frame("Intervals"=c('1 Mb','SNP','SNP'),
        "HS model"=c('number/density','number','density'),
    "logp"=-log10(c(pval.1m.qgc.nhs,pval.snp.qgc.nhs,pval.snp.qgc.hsdens)))


colnames(res.hs)[2]=c('HS effect')
colnames(res.hs)[3]=c('$-\\log_{10}(p)$')
```

```
print(xtable(res.hs,
caption='Significance of hotspot effect on meiotic recombination rate.%
Number: effect of the number of hotspots.%
Density: effect of the density of hotspots (in HS/10Kb).'),
type='latex',include.rownames=FALSE,sanitize.text.function=function(x){x})
```

Intervals	HS effect	$-\log_{10}(p)$
1 Mb	number/density	104.23
SNP	number	316.41
SNP	density	191.83

Table SM6: Significance of hotspot effect on meiotic recombination rate.Number: effect of the number of hotspots.Density: effect of the density of hotspots (in HS/10Kb).

Calculate correlations corrected for chromosome and gc content effects.

```
csnp.nhs=cor.test(residuals(lm.snp.chr.qgc),gcs$nhs)
csnp.hsdens=cor.test(residuals(lm.snp.chr.qgc),gcs$hsdens)
c1.nhs=cor.test(residuals(lm.1m.chr.qgc),gc1$nhs)
pval.vec=c(c1.nhs$p.value,csnp.nhs$p.value,csnp.hsdens$p.value)
res.cor.res=data.frame("Intervals"=c('1 Mb','SNP','SNP'),
    "HS model"=c('number/density','number','density'),
    "corr."=c(c1.nhs$estimate,csnp.nhs$estimate,csnp.hsdens$estimate),
    "p-value"=-log10(pval.vec))

colnames(res.cor.res)[2]=c('HS effect')
colnames(res.cor.res)[3]=c('Correlation')
colnames(res.cor.res)[4]=c('$-\\log_{10}(p)$')

print(xtable(res.cor.res,
  caption='Correlation between historical hotspots and meiotic recombination rate%
  corrected for chromosome and GC content effects.%
  Number:  number of hotspots.%
  Density:  density of hotspots (in HS/10Kb).'),
  type='latex',include.rownames=FALSE,sanitize.text.function=function(x){x})
```

Intervals	HS effect	Correlation	$-\log_{10}(p)$
1 Mb	number/density	0.43	100.69
SNP	number	0.18	314.09
SNP	density	0.14	190.55

Table SM7: Correlation between historical hotspots and meiotic recombination ratecorrected for chromosome and GC content effects.Number: number of hotspots.Density: density of hotspots (in HS/10Kb).

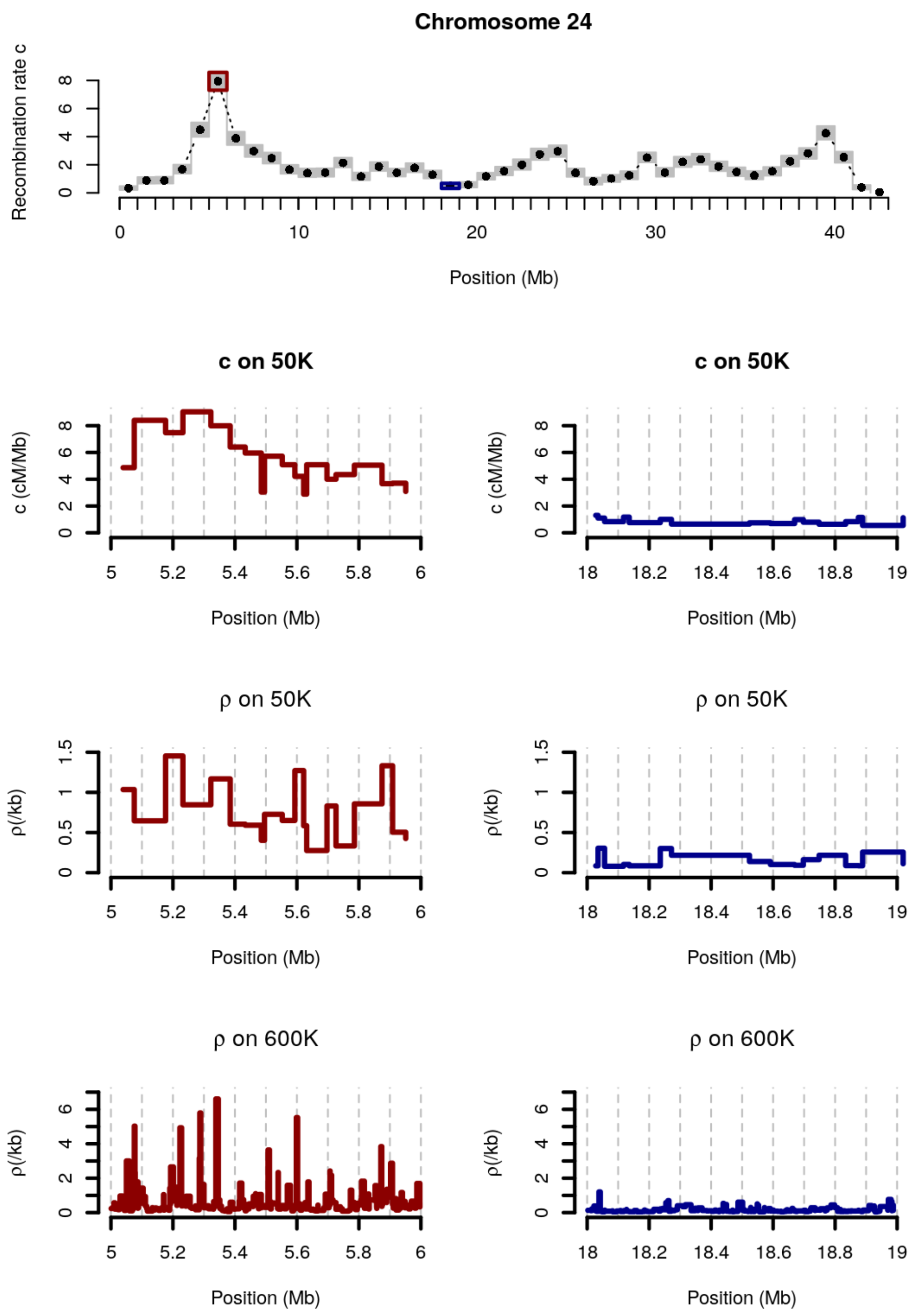
2.1.4 Combining family- and LD-based inferences

1. Illustration of the approach

   We can illustrate the comparison betwee the two approach on chromosome 24. First we run a script to gather data from LD-based and pedigree-based maps. This creates two files `results/combined/compare_family_1Mb.txt` and `results/combined/compare_family_60K.txt`.

   ```
   python code/compare_maps.py
   ```

   This allows to plot the different maps easily, as shown here for two windows on chromosome 24



2. Linear Mixed model combining pedigree and LD analyses

   A high-density recombination map should provide the recombination rate in small intervals on the genome. Here the intervals are defined by the position of the HD SNP array markers. To obtain a HD recombination map, the approach taken is to correct LD-based recombination rates for the influence of demography and scale them in centiMorgans/Megabase unit. The influence of demography is measured by a multiplicative factor that corresponds to the effective number of chromosomes (Ne). This parameter Ne can change along the genome, in particular due to past selection. To account for this effect, we estimate a different value of the parameter for windows of 1 megabase along the genome, using linear mixed models.

The model considered includes a `method` fixed effect, which estimates the (log10 of the) main multiplicative factor 4*Ne, *i.e* it provides an estimate of the (log10) average effective population size of the Lacaune population.

Next, it includes a chromosome random effect to account for different recombination rates between chromosomes.

Finally, it includes a `seg` random effect, which models the recombination rate variation between windows and finally a *method within segment* `seg/method` effect that models difference in recombination rates estimates by the two methods (family and pop) for each window.

```
library(lme4)
library(reshape2)
## read in data and combine samples
fam=read.table('results/combined/comb_1Mb_map_family.txt',head=T)
pop=read.table('results/combined/comb_1Mb_map_pop51.txt',head=T)
tot=rbind(fam,pop)
tot$seg=paste(tot$chr,tot$left,tot$right,sep='-')
tot$seg=as.factor(tot$seg)
chrsize=do.call(rbind,list(by(tot$right,tot$chr,max)))
## remove extremities
for(i in 1:26) {
    end=chrsize[i]
    torm=(tot$chr==i)&(tot$right<=4e6)
    tot=tot[!torm,]
    torm=(tot$chr==i)&(tot$right>=(end-4e6))
    tot=tot[!torm,]
}
## The response considered is the log(10) of recombination rate
tot$logc=log10(tot$value)

## Fit a linear mixed model to estimate local Ne
mymod=lmer(logc~method+(1|chr)+(1|seg/method),data=tot)
```

```
Attaching package: ‘ascii’

The following object is masked from ‘package:Matrix’:

    expand
| Linear mixed model fit by REML ['lmerMod']                          |
|---------------------------------------------------------------------|
| Formula: logc ~ method + (1  \vert  chr) + (1  \vert  seg/method)   |
| Data: tot                                                           |
|                                                                     |
| REML criterion at convergence: -250447.6                            |
|                                                                     |
| Scaled residuals:                                                   |
| Min       1Q   Median       3Q      Max                             |
| -10.8780  -0.5469   0.0124   0.5739   8.9327                            |
|                                                                        |
| Random effects:                                                        |
| Groups     Name        Variance Std.Dev.                               |
| method:seg (Intercept) 0.012832 0.11328                                |
| seg        (Intercept) 0.031638 0.17787                                |
| chr        (Intercept) 0.001552 0.03940                                |
| Residual               0.002685 0.05181                                |
| Number of obs: 89160, groups:  method:seg, 4458; seg, 2229; chr, 26 |
|                                                                        |
| Fixed effects:                                                         |
| Estimate Std. Error t value                                            |
| (Intercept) 0.100447   0.009185    10.9                                |
| methodpop   4.449792   0.003411  1304.6                                |
|                                                                        |
| Correlation of Fixed Effects:                                          |
| (Intr)                                                                 |
| methodpop -0.186                                                       |
```

The `(Intercept)` term estimates the base recombination rate: on a log 10 scale ~ 0.1, corresponding to ~ 1.26 cM/Mb. The `methodpop` term estimates log10(4Ne), where Ne is the genome averaged effective population size, here ~ 4.45, corresponding to Ne ~ 7000 individuals.

We can now rescale LD-based recombination rate for the genome average Ne and compare them to meiotic recombination rates. We first extract the predicted recombination rate in each segments for both approaches, then remove the log10(4Ne) term from the LD-based estimates.

```
require(lme4)
## Genome wide scaling factor for LD-based estimates
log.Ne=fixef(mymod)[2]
## Get predicted values in each segment
pred=tot[tot$rep==0,]
fit.val=predict(mymod,pred)
pred$fit=fit.val
pred.wide=dcast(pred,chr + left + right + seg ~ method,value.var='fit')
linmod=lm(I(pop-log.Ne)~family,data=pred.wide)


fam.c=pred.wide$family
## rescale LD-based estimates
pop.c=pred.wide$pop-log.Ne

## correlation between LD and family based estimates
cor.c=cor(pred.wide$family,pred.wide$pop)

par(mar=c(5,6,1,1))
```

```
plot(fam.c,pop.c,
     pch=19,col=rgb(0.63,0.32,0.17,0.2),axes=F,
     xlab='Meiotic recombination rate (cM/Mb)',
     ylab='',
     xlim=c(-1,1),
     ylim=c(-1.5,1),
     cex.lab=1.5
     )
lab.x=c(0.1,0.25,0.5,1,2,4,10)
tks.x=log10(lab.x)
lab.y=c(0.05,0.25,0.1,0.5,1,2,4,10)
tks.y=log10(lab.y)
axis(1,at=tks.x,lab=lab.x,cex.axis=1.3)
axis(2,at=tks.y,lab=lab.y,las=2,cex.axis=1.3)
title(ylab='Scaled historical recombination rate (cM/Mb)',line=4,cex.lab=1.5)
abline(coef=coef(linmod),lwd=2,lty=2)
text(log10(4),log10(0.1),labels=paste('Correlation = ',round(cor.c,digits=2)),cex=1
```

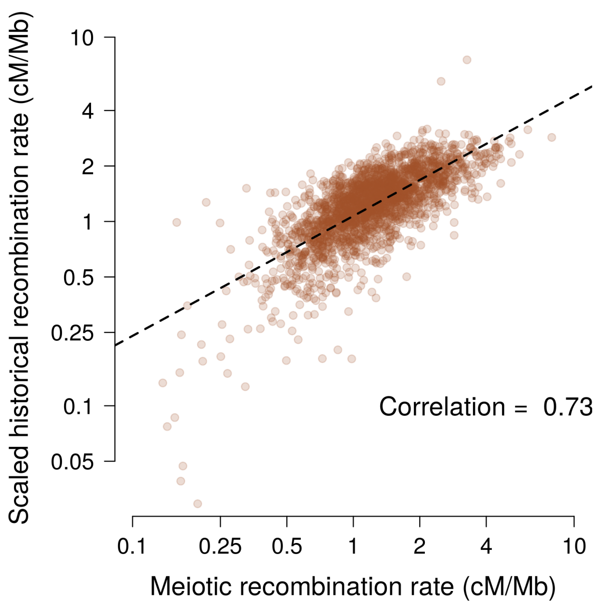


Figure SM11: Meiotic and scaled historical recombination rates in windows of one megabase. The dashed line is the regression for population recombination rate on the family recombination rate. Values are shown on a logarithmic scale.

3. Identification of windows with outlying differences between historical and meiotic recombination rates

   We can then examine individual windows to identify outliers: windows where LD-based estimates are significantly too high or too low, given the meiotic recombination rate. To avoid effects due to low coverage of the HD SNP chip, we correct for the a SNP density effect.

```
## Get HD SNP map
```

```
snps=read.table('data/population/HDpanel/Lacaune.bim')
colnames(snps)=c('chr','name','gen','pos','A1','A2')

## function to compute number of SNP within a window
nsnp.seg=function(coord,pmap) {
    sub.snp=(pmap$chr==as.integer(coord[1]))&(pmap$pos>=coord[2])&(pmap$pos<=coord
    return(sum(sub.snp))
}

pred.wide$nsnps=apply(pred.wide[,c(1,2,3)],1,nsnp.seg,pmap=snps)

pred.wide$len=pred.wide$right-pred.wide$left
pred.wide$snpdens=pred.wide$nsnps/pred.wide$len

## fit a linear model of historical rate with meiotic rate,
## adjusting for snp density (on a log scale)

linmod=lm(I(pop-log.Ne)~I(log10(snpdens))+family,data=pred.wide)

## We want to look at regions where pop and family are significantly different
require(MASS)
str.resid.pop=studres(linmod)
pred.wide$pval.hi=as.vector(pnorm(str.resid.pop,lower.tail=F))
pred.wide$pval.lo=as.vector(pnorm(str.resid.pop))

hh=hist(str.resid.pop,n=100,
    main='Historical Rates Residuals',
    xlab='Studentized Residuals',
    freq=FALSE)
xx=seq(-10,10,0.01)
lines(xx,dnorm(xx),lwd=2)
```

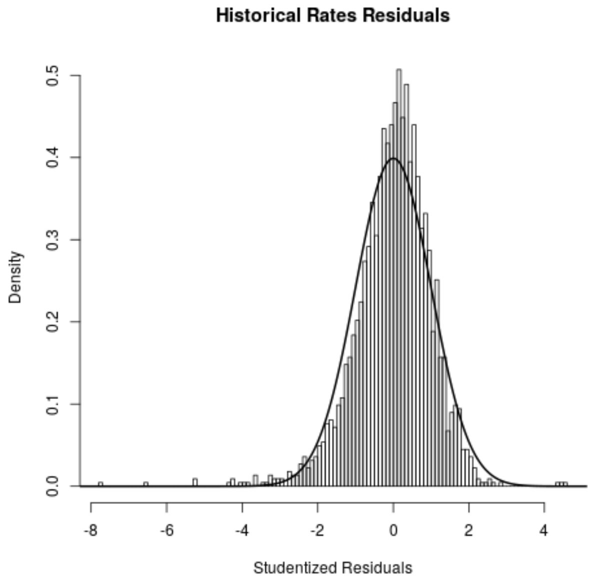


Figure SM12: Residuals from the linear modeling of recombination rates combining meiotic and historical estimates

We can see clear outliers from the expected standard Gaussian distribution, which we now extract

```
require(qvalue)
require(xtable)

pred.wide$resid=as.double(str.resid.pop)
pred.wide$regresid=as.double(residuals(linmod))

pred.wide$quantile=pnorm(qqnorm(pred.wide$family,plot.it=F)$x)
## compute values from
pred.wide$pval.hi=as.vector(pnorm(pred.wide$resid,lower.tail=F))
pred.wide$pval.lo=as.vector(pnorm(pred.wide$resid))

pval=2*pnorm(abs(pred.wide$resid),lower.tail=F)
pred.wide$pval2side=as.double(pval)
qval=qvalue(pval,pi0.method='bootstrap')

fdr.th=max(qval$pvalues[qval$qvalues<0.02])

zones=subset(pred.wide[pred.wide$pval2side<fdr.th,],
    select=c(chr,left,right,quantile,regresid,pval2side))

zones$chr=as.factor(zones$chr)
zones$left=zones$left*1e-6
zones$right=zones$right*1e-6
zones$quantile=round(zones$quantile,digits=3)
zones$regresid=round(10^zones$regresid,digits=2)
```

```
zones$pval2side=format.pval(zones$pval2side,digits=2)
colnames(zones)=c('Chromosome','Left (Mbp)',
            'Right (Mbp)','Meiotic rec. rank','Ratio','p-value')

print(xtable(zones,
  caption='Genome regions where meiotic and historical %
recombination rates differ significantly.'),include.rownames=FALSE,
  type='latex',sanitize.text.function=function(x){x})

write.table(zones,quote=F,row.names=F,file='results/combined/outliers_regions.txt'
```

Loading required package: qvalue Loading required package: xtable

Chromosome	Left (Mbp)	Right (Mbp)	Meiotic rec. rank	Ratio	p-value
3	103.00	104.00	0.06	0.28	1.6e-05
3	109.00	110.00	0.04	0.28	1.8e-05
6	36.00	37.00	0.14	0.21	1.2e-07
6	37.00	38.00	0.23	0.22	1.9e-07
10	29.00	30.00	0.77	0.31	8.8e-05
10	36.00	37.00	0.01	0.29	2.1e-05
10	42.00	43.00	0.00	0.15	4.3e-11
10	43.00	44.00	0.00	0.11	1.2e-14
12	4.00	5.00	0.92	3.73	7.4e-06
13	63.00	64.00	0.33	0.31	5.6e-05
20	28.00	29.00	0.01	3.54	1.7e-05
23	10.00	11.00	0.97	3.82	5.1e-06

Table SM8: Genome regions where meiotic and historical recombination rates differ significantly.

4. Construction of a high-density recombination map

   Based on meiotic recombination rate, we estimate, within each 1Mb window the local effective population size. Then, we scale the historical recombination rates by this size to get historical recombination rates in centiMorgans/Megabase units and produce a high density recombination map, in form of a `bim` file with genetic distances as third column.

```
  ## local.Ne estimates log10(4Ne)
  local.Ne=pred.wide$pop-pred.wide$family

  ## get a global estimate for sub-telomeric regions
  genome.Ne=median(local.Ne)

  ## gather local.Ne estimates
  ## For sub-telomeric regions, this is set to genome.Ne
  mypop=pop[pop$rep==0,c(1,2,3)]
  mypop$seg=paste(mypop$chr,mypop$left,mypop$right,sep='-')
  mypop$scale=genome.Ne
  mypop$scale[match(pred.wide$seg,mypop$seg)]=local.Ne
  colnames(mypop)[5]='scale (log10(4Ne))'
  write.table(mypop[,c(1,2,3,5)],row.names=F,quote=F,
              file='results/combined/pop_scale.txt')

  ## Scale our map
  popscale=read.table('results/combined/pop_scale.txt',skip=1)

  pop.hd=read.table('results/population/HD_SNP_array_map.txt',h=T)

  scale.delta=function(seg,scale=popscale) {
      idx=NULL
      seg=unlist(seg)
      idx=which((popscale[,1]==seg[1])&(popscale[,2]<=seg[2])&(popscale[,3]>=seg[3]
      if (length(idx)<1) {
          idx=max(which((popscale[,1]==seg[1])&(popscale[,2]<=seg[2])))
          return(popscale[idx,4])
      } else {
          return(popscale[idx,4])
      }
  }

  tt=unlist(apply(pop.hd,1,scale.delta))

  pop.hd$scale=tt
  pop.hd$d=100*pop.hd$delta*10^-pop.hd$scale ## in cM
  pop.hd$c=1e6*pop.hd$d/(pop.hd$right-pop.hd$left) ## in cM/Mb
  pop.hd$rho=1e3*pop.hd$delta/(pop.hd$right-pop.hd$left) ## per Kb

write.table(pop.hd,quote=F,row.names=F,
            file='results/combined/HD_SNP_array_map_scaled.txt')

## Now create a bim file with genetic distances,
## using linear approximation for markers
## snps object contains original bim file data

  for (chrom in 1:26) {
      mysnps=snps$chr==chrom
      mymap=pop.hd$chr==chrom
      myd=c(0,cumsum(pop.hd$d[mymap]))
      myp=c(pop.hd$left[mymap][1],pop.hd$right[mymap])
      ff=approxfun(myp,myd,rule=2)
      snps$gen[mysnps]=round(ff(snps$pos[mysnps]),digits=3)
  }

write.table(snps,quote=F,row.names=F,
            file='results/combined/Illumina_Ovine_HD.bim')
```

2.1.5 Comparison of Recombination maps in Soay and Lacaune

1. Comparison to new Soay maps

   We downloaded Soay data from Johnston et al. (2016) from the dryad repository. Specifically we used the files named:

   - `20150129merged1_66nodups.QC2.{bed,bim,fam}`
   - `2_FamilyPedigree_FullClean_g.txt`

   Specifically we kept all individuals from the file `20150129merged1_66nodups.QC2.{bed,bim,fam}` that are listed in the file `2_FamilyPedigree_FullClean_g.txt` (column ANIMAL).

   We constructed integer identifiers for each individual (`code/soay_renumfam.py`). The file `data/Soay/keep_indivs.txt` contains animals from `2_FamilyPedigree_FullClean_g.txt` with their ids changed from their original IDs to linkphase integers (see R script `code/soay_getindiv2keep`

   We then ran LINKPHASE on the Soay data:

   ```
   ./code/soay_prepare_linkphase.sh
   ./code/soay_run_linkphase.sh
   ```

   Finally, we can select focal individuals (FIDs) from the Soay data, using the same script used for the Lacaune given in `code/selectFIDs.R`. A first run of the estimation procedure revealed that one individual had a very high number of crossovers (~100) per meioses (RE4844) so we discarded it by commenting it out in the `data/Soay/FIDs.txt` file. Soay recombination maps can be estimated using

   ```
   python code/soay_family_map.py
   ```

   ```
   lac=read.table('results/family/SNP_array_map.txt',h=T)
   soay=read.table('results/Soay/SNP_array_map.txt',h=T)
   ## intervals where average rec. rate is less than 1cM/Mb
   sub=0.5*(soay$m_cj+lac$m_cj)<1.5

   library(RColorBrewer)

   coul=brewer.pal(4,'Set1')
   par(mfrow=c(1,2),mar=c(5,5,1,1))
   ## Comparison of rec. rates
   ss.rate=lowess(log10(soay$m_cj),log10(lac$m_cj),f=0.05)
   plot(y=log10(soay$m_cj),x=log10(lac$m_cj),pch=16,col=coul[2],
        xlim=c(-1,1),
        ylim=c(-1,1),
        xlab='Lacaune rec. rate (cM/Mb)',
        ylab='Soay rec. rate (cM/Mb)',axes=F)
   points(y=log10(soay$m_cj[sub]),x=log10(lac$m_cj[sub]),pch=16,col='gray')
   lines(x=ss.rate$y,y=ss.rate$x,col=coul[1],lwd=4)
   abline(0,1,col=coul[3],lwd=3,lty=2)
   axis(1,at=log10(c(0.1,0.2,0.5,1,2,5,10)),labels=c(0.1,0.2,0.5,1,2,5,10))
   ```

```
axis(2,at=log10(c(0.1,0.2,0.5,1,2,5,10)),labels=c(0.1,0.2,0.5,1,2,5,10))
## Comparison of precisions
#plot(log10(lac$m_cj),(soay$s_cj/lac$s_cj)^2,pch=16,col=coul[2],
plot(log10(0.5*(lac$m_cj+soay$m_cj)),(soay$s_cj/lac$s_cj)^2,pch=16,col=coul[2],
     xlim=c(-1,1),ylim=c(0,9),
     xlab='Average rec. rate (cM/Mb)',
     ##     ylab=expression(sigma[Soay]^2/sigma[Lacaune]^2),
     ylab='Posterior Variance Ratio (Soay/Lacaune)',
     axes=FALSE)
axis(1,at=log10(c(0.1,0.2,0.5,1,2,5,10)),labels=c(0.1,0.2,0.5,1,2,5,10))
axis(2,at=c(0,1,2,4,6,8))
#points(log10(lac$m_cj[sub]),(soay$s_cj[sub]/lac$s_cj[sub])^2,pch=16,col='gray')
points(log10(0.5*(lac$m_cj+soay$m_cj)[sub]),(soay$s_cj[sub]/lac$s_cj[sub])^2,pch=16
ss.prec=lowess(log10(lac$m_cj),(soay$s_cj/lac$s_cj)^2,f=0.05)
lines(ss.prec,col=coul[1],lwd=4)
abline(h=1,lwd=3,lty=2,col=coul[3])
```

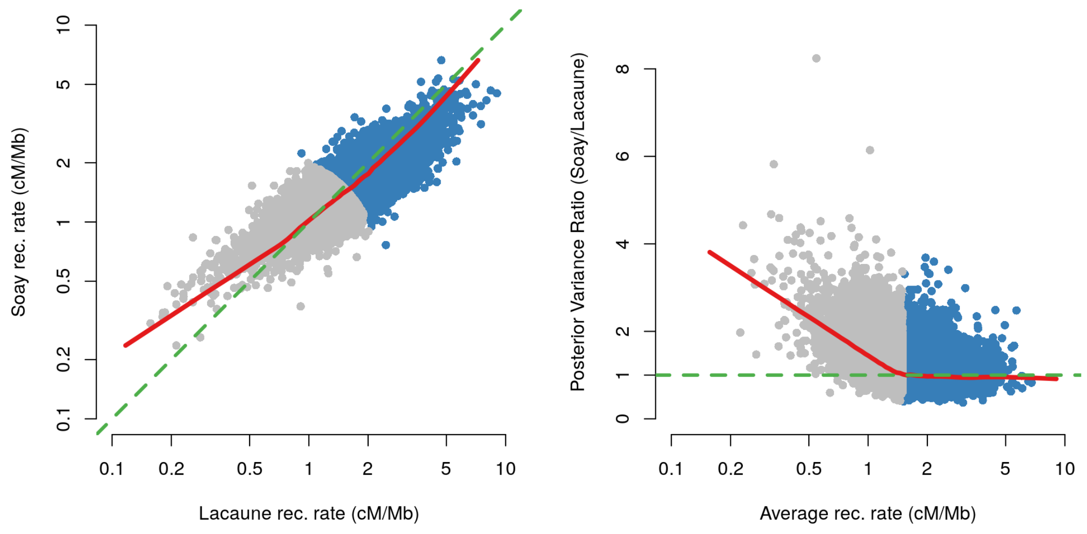


Figure SM13: Comparison of recombination rates in Lacaune and Soay males on the SNP array.

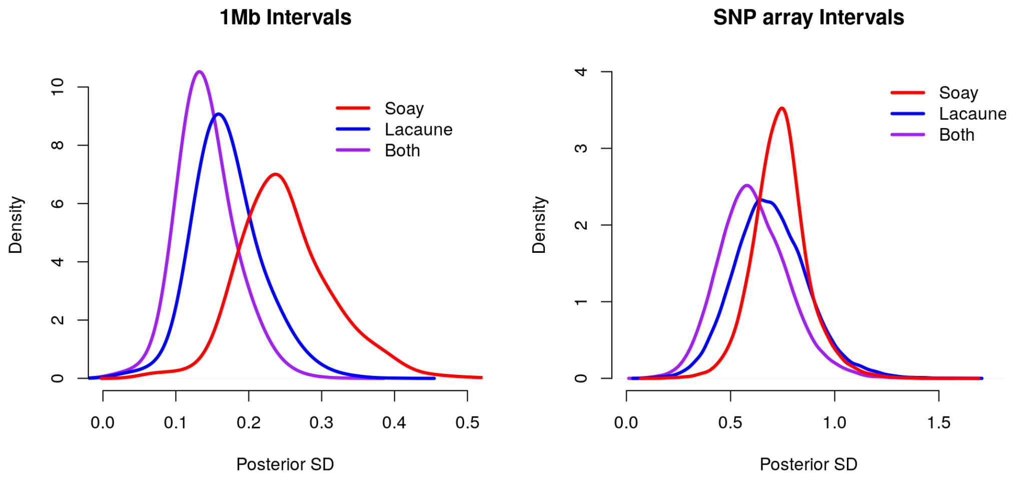
2. Combining datasets to create new recombination maps

   Given that the two populations exhibit similar recombination maps, we combined the datasets by considering crossovers detected in both populations to establish new recombination maps.

```
python code/soay_lacaune_family_map.py
```

   This creates two new recombination maps at the one megabase scale (`results/family/soay_lacaune_1M`) and at on the SNP array (`results/family/soay_lacaune_SNP_array_map.txt`).

   We can illustrate the gain in precision from these maps by looking at the distribution of posterior standard deviations of recombination rates



2.2 Genetic Determinism of Genome-wide recombination Rate in Lacaune Sheep

2.2.1 Covariate effects on GRR

For each sire selected in our FID list, we first extract for each of its offpring, its year of birth and insemination date of the ewe, gathering data from LINKPHASE output and in the `data/family/sheep_covariate` file. This is done by running the `code/list_meioses.py`.

```
python code/list_meioses.py
```

This creates the file `results/family/nco_meioses.txt`.

```
grr=read.table('results/family/nco_meioses.txt')
colnames(grr)=c('offspring','sire','year','insem','nco')
grr$insem.mo=with(grr,month.abb[insem])
grr$insem.mo=ordered(grr$insem.mo,levels=month.abb)
grr$sire=as.factor(grr$sire)
grr$year=as.factor(grr$year)
grr$offspring=as.factor(grr$offspring)
head(grr)
```

```
  offspring sire year insem nco insem.mo
1     10006 3923 2004     3  38      Mar
2      9267 3923 2004     6  35      Jun
3     10146 3923 2004     3  33      Mar
4     10149 3923 2004     4  30      Apr
5     10733 3923 2004     5  31      May
6      9677 3923 2004     6  28      Jun
```

Using this file, we can test for Year-of-birth and Insemination month effects.

```
library(lme4)

grr=grr[complete.cases(grr$year)&complete.cases(grr$insem),]
```

```
mod.null=lmer(nco ~ (1|sire),data=grr,REML=FALSE)

mod.year=lmer(nco ~ (1|sire)+year,data=grr,REML=FALSE)
## No Year of birth effect
anova(mod.null,mod.year)
## A small insemination month effect
mod.insem=lmer(nco ~ (1|sire)+insem.mo-1,data=grr,REML=FALSE)
ci=confint(mod.insem)[-c(1,2),]

anova(mod.null,mod.insem)
```

This significant effect seems mostly due to slightly increased recombination in May.

```
## fit using REML
mod.insem.reml=update(mod.insem,REML=TRUE)
ci=confint(mod.insem.reml)[-c(1,2),]
layout(matrix(c(1,1,1,1,2,2),ncol=2,byrow=T))
plot(2:8,fixef(mod.insem.reml),axes=F,pch=19,
     ylab='Mean number of Crossovers / meiosis',
     xlab='Insemination Month',
     type='b',lty=2,lwd=2,
     ylim=c(33,38.5),cex=2)
axis(1,at=2:8,labels=month.abb[2:8],las=2)
axis(2)
head(ci)
segments(x0=2:8,x1=2:8,y0=ci[,1],y1=ci[,2],lwd=2)
barplot(table(grr$insem.mo)[2:8],ylab='Number of Inseminations')
```

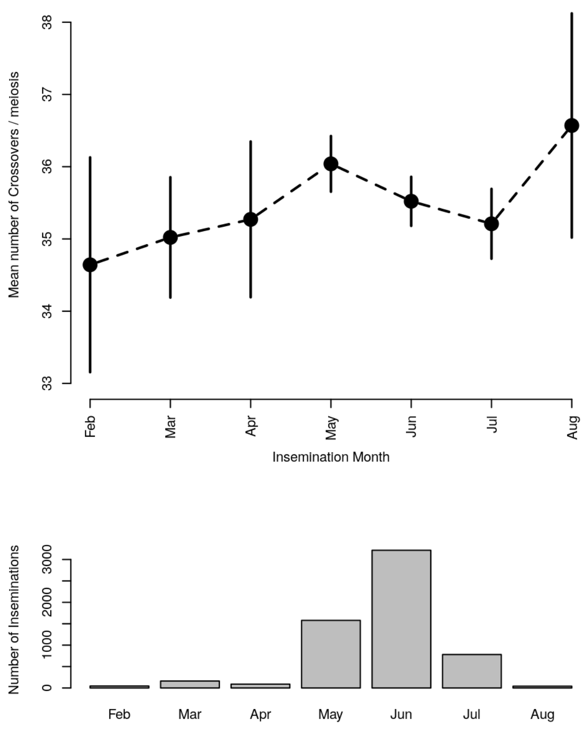


Figure SM14: Effect of insemination month on genome-wide recombination rate. Top: average and 95% confidence intervals. Bottom: number of observation per month.

2.2.2 Genetic parameters of GRR in the Lacaune population

To esimate genetic parameters of the genowide recombination rate (GRR) phenotype in the Lacaune population, we fit an additive genetic model, with covariance between individuals estimated from their pedigree going back 4 generations (file `data/family/lacaune_pedigree_4G.txt`). We use the `bluf90` suite, specifically `renumf90` to create input files for `airemlf90` that is used for esimating genetic parameters and additive genetic values. This analysis is run using the `code/run_aireml.py` script:

```
python code/run_aireml.py
```

Then we extract the predicted additive genetic values (BLUP) of each of the 345 sires. A comparison of BLUP to the mean number of crossover per meiosis illustrates nicely the difference between independant animal effects and genetic values, in particular shrinkage (notive how BLUPs are shrinked toward 0 compared to the least squares estimates).

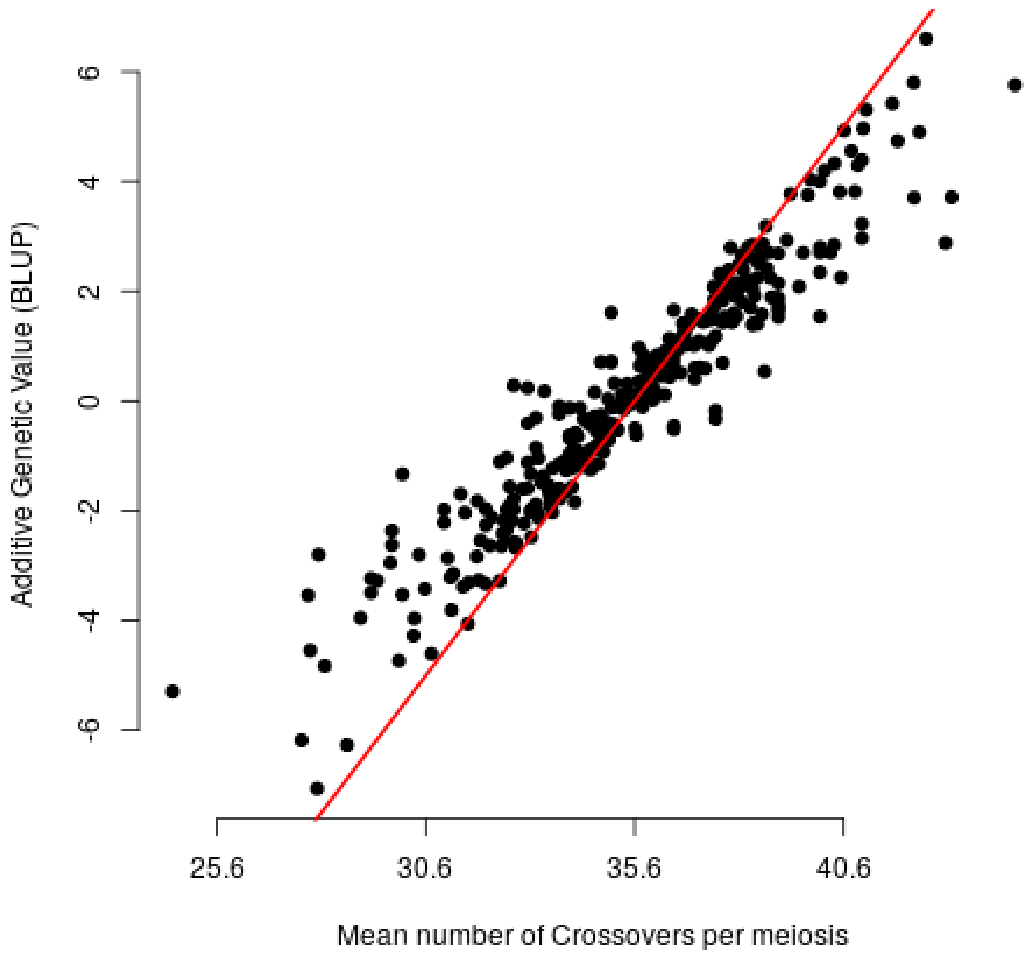
```
parec=read.table('results/family/parent_recombination.txt',h=T)
sol=read.table('results/family/solutions.aireml',skip=1)
## overall mean
mu=sol[1,4]
## Sire effects
blup=sol[sol$V2==4,]
```

```
## Correspondance between levels and sires
corresp.ped=read.table('results/family/renadd04.ped')

parec$id=corresp.ped$V1[match(parec$parent,corresp.ped$V10)]
parec$blup=blup$V4[parec$id]

m=mean(parec$Ri)
plot(parec$Ri-m,parec$blup,
     xlab='Mean number of Crossovers per meiosis',
     ylab='Additive Genetic Value (BLUP)',pch=19,axes=F)
axis(1,at=axTicks(1),labels=round(axTicks(1)+m,digits=1))
abline(0,1,lwd=2,col=2)
axis(2)

write.table(parec,file='results/family/parent_recombination_blup.txt',quote=F,row.names=
```



2.2.3 Genome-wide association study identifies two major loci affecting GRR in Lacaune sheep

1. Imputation

   We use bimbam for imputation of genotypes. The script to convert plink files to bimbam input is provided in `code/gwas_imputation.py`. This script will output a file with all bimbam commands to be run for imputation, it can be modified to actually run the imputation, but that can take a while on a single CPU.

2. GWAS analysis

   We run a genome wide association study on GRR using `gemma`. GWAS input data are provided in the `data/gwas/` directory. The files are:

   - `lacaune.mean.genotypes.txt`: mean genotypes of individuals at all markers, output of the `BIMBAM` imputation run
   - `grr_blup.txt`: individual deviations from the overall mean. Obtained from the AIREML analysis.
   - `snpinfo.txt`: SNP info file.

   First, we calculate the genomic relationship matrix between individuals:

```
gemma -g data/gwas/lacaune.mean.genotypes.txt \
      -p data/gwas/grr_blup.txt  \
      -gk 1 -outdir results/gwas/ -o grr
```

   Then, we run the single SNP GWAS analysis:

```
gemma -g data/gwas/lacaune.mean.genotypes.txt \
      -p data/gwas/grr_blup.txt  -a data/gwas/snpinfo_gemma.txt \
      -k results/gwas/grr.cXX.txt\
      -lmm 4 -o grr -outdir results/gwas/
```

   Now we can estimate a multiQTL model using BSLMM

```
gemma -g data/gwas/lacaune.mean.genotypes.txt \
      -p data/gwas/grr_blup.txt  -a data/gwas/snpinfo_gemma.txt \
      -k results/gwas/grr.cXX.txt\
      -bslmm 1 -o grr_multi -outdir results/gwas/

gemma -g data/gwas/lacaune.mean.genotypes.txt \
      -p data/gwas/grr_blup.txt  -a data/gwas/snpinfo_gemma.txt \
      -k results/gwas/grr.cXX.txt\
      -bslmm 1 -o grr_multi_10M -s 10000000 -outdir results/gwas/
```

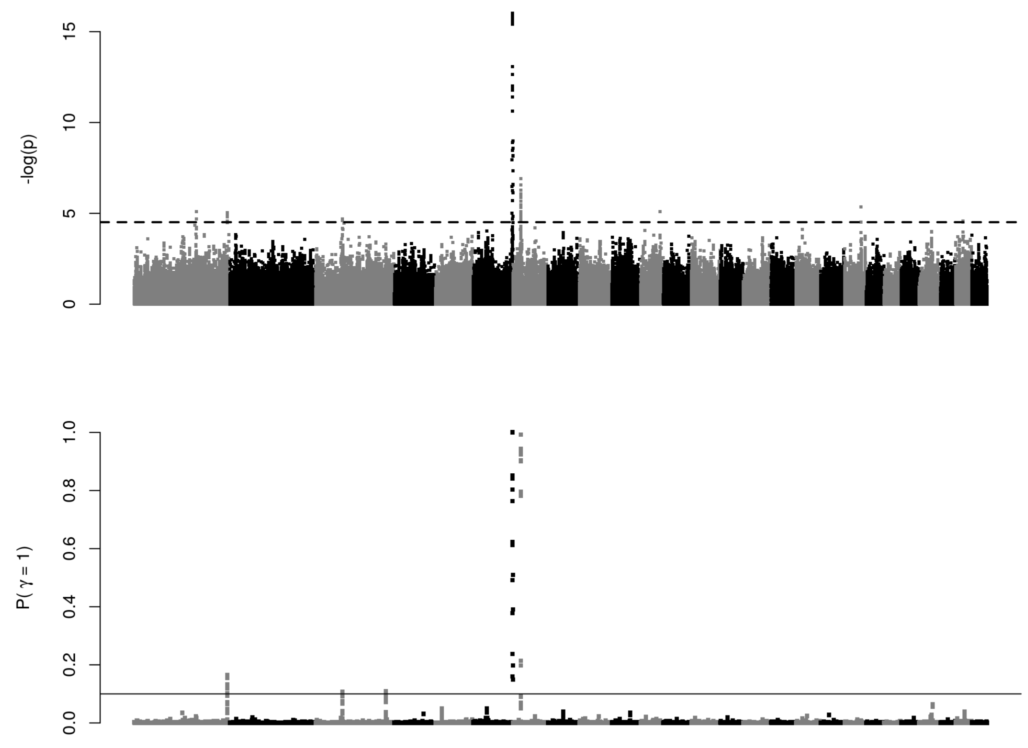
   Read in association results, estimate FDR and local QTL probabilities

```
library(ashr)
library(zoo)

### LMM results
assoc=read.table('results/gwas/grr.assoc.txt',h=T)
ash.grr=ash(assoc$beta,assoc$se)

### BSLMM results
gam=read.table('results/gwas/grr_multi_10M.param.txt',h=T)
ksnp=50
gam$locgamma=rollsum(gam$gamma,k=ksnp,fill=NA)
```

An examine likely QTLS:



2.2.4 Association with RNF212

The genotype of 266 individuals at four mutations in RNF212 are given in file =data/gwas/rnf212mut.{bed,bim.fam
As we only have genotypes at 266 individuals that we need to extract from the mean genotype file to calculate their genomic relationships.

```
fam=read.table('data/gwas/rnf212mut.fam')
gwas.indiv=scan('data/gwas/individuals.txt',what='character')
mgw=read.table('data/gwas/lacaune.mean.genotypes.txt')

rnf212.indiv=match(fam$V2,gwas.indiv)
rnf212.mgw=mgw[,c(1,2,3,3+rnf212.indiv)]
```

```
write.table(rnf212.mgw,quote=F,row.names=F,col.names=F,'data/gwas/rnf212.indivs.mean.ge

blup=scan('data/gwas/grr_blup.txt')
rnf212.blup=blup[rnf212.indiv]

write.table(rnf212.blup,quote=F,row.names=F,col.names=F,file='data/gwas/rnf212.blup')
```

We can then run gemma on the subset of individuals and test for association of RNF212 mutations

```
gemma -g data/gwas/rnf212.indivs.mean.genotypes.txt \
      -p data/gwas/rnf212.blup \
      -gk 1 -outdir data/gwas/ -o rnf212

gemma -bfile data/gwas/rnf212mut \
      -p data/gwas/rnf212.blup  \
      -k data/gwas/rnf212.cXX.txt\
      -lmm 4 -o rnf212 -outdir results/gwas/
```

We can also impute all individuals for the RNF212 mutations with bimbam:

```
## exctract the end of OAR6
plink --sheep --bfile data/gwas/FinalCohort --chr 6 --from-kb 115000 \
      --out data/gwas/cohort-chr6qtl --make-bed
plink --sheep --bfile data/gwas/cohort-chr6qtl --bmerge data/gwas/rnf212mut \
      --make-bed --out data/gwas/cohort-chr6qtl-mut

## get panel
plink --sheep --family --bfile data/population/HDpanel/frenchsheep_HD \
  --keep-cluster-names LAC LAM \
  --chr 6 --from-kb 115 \
  --out data/gwas/panel-chr6qtl-mut --recode-bimbam

## Recode bimbam
plink --sheep --bfile data/gwas/cohort-chr6qtl-mut --chr 6 \
      --recode-bimbam --out data/gwas/cohort-chr6qtl-mut

## Run bimbam
bimbam -g data/gwas/panel-chr6qtl-mut.recode.geno.txt -p 0 \
       -pos data/gwas/panel-chr6qtl-mut.recode.pos.txt \
       -g data/gwas/cohort-chr6qtl-mut.recode.geno.txt \
       -p data/gwas/cohort-chr6qtl-mut.recode.pheno.txt\
       -pos data/gwas/cohort-chr6qtl-mut.recode.pos.txt  \
       -e 10 -w 20 -s 1 -c 15 -o imput_rnf212 -wmg

mv output/imput_rnf212.mean.genotype.txt data/gwas/
awk '{print $1, $6 ,$5}' output/imput_rnf212.snpinfo.txt  | sed -e s/0$/6/g >data/gwas/
```

and then perform a local GWAS analysis with gemma:

```
gemma -g data/gwas/imput_rnf212.mean.genotype.txt \
      -p data/gwas/grr_blup.txt \
      -k results/gwas/grr.cXX.txt \
      -a data/gwas/imput_rnf212.snpinfo.txt \
      -lmm 4 -o rnf212 -outdir results/gwas/
```

```
gemma -g data/gwas/imput_rnf212.mean.genotype.txt \
      -p data/gwas/grr_blup.txt \
      -k results/gwas/grr.cXX.txt \
      -a data/gwas/imput_rnf212.snpinfo.txt -bslmm 1 -o rnf212_multi -outdir results/gwa
```

The top SNP is a mutation within the RNF212 gene:

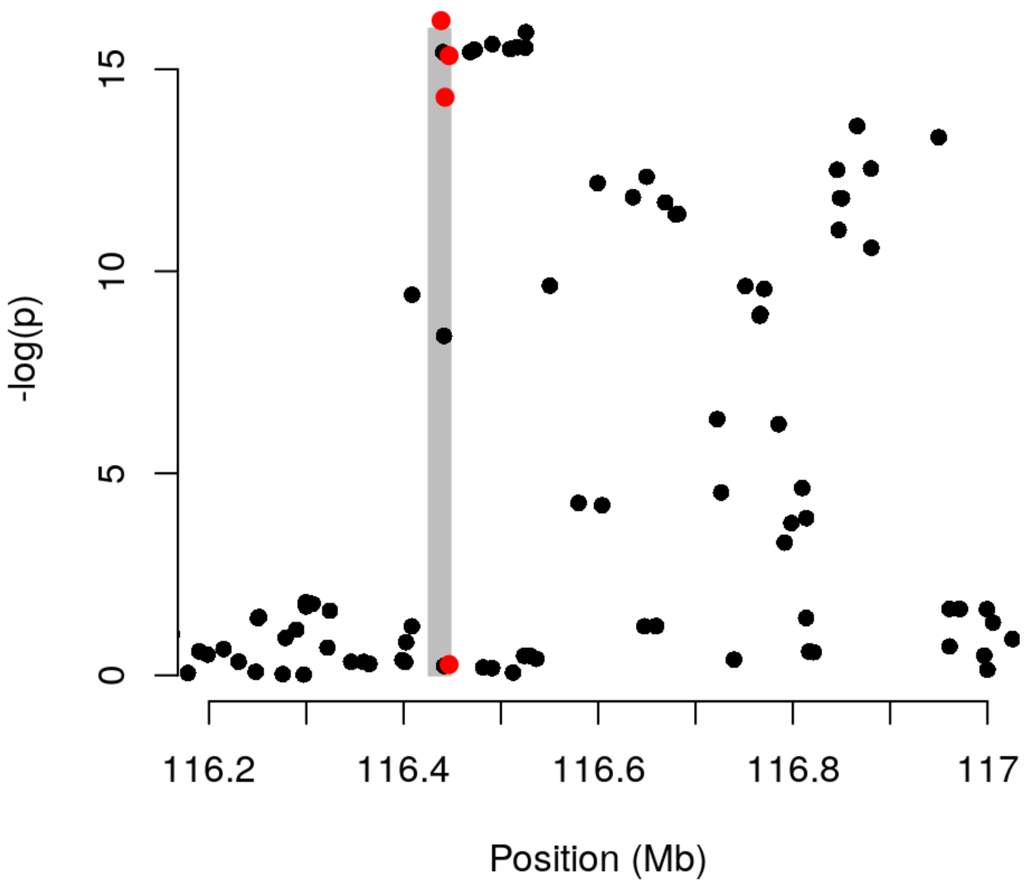


Figure SM15: Association in the distal end of chromosome 6

Get Soay association data in the region

```
curl http://www.genetics.org/content/genetics/suppl/2016/03/29/genetics.115.185553.DC1/
```

And we can compare the association results between the two populations

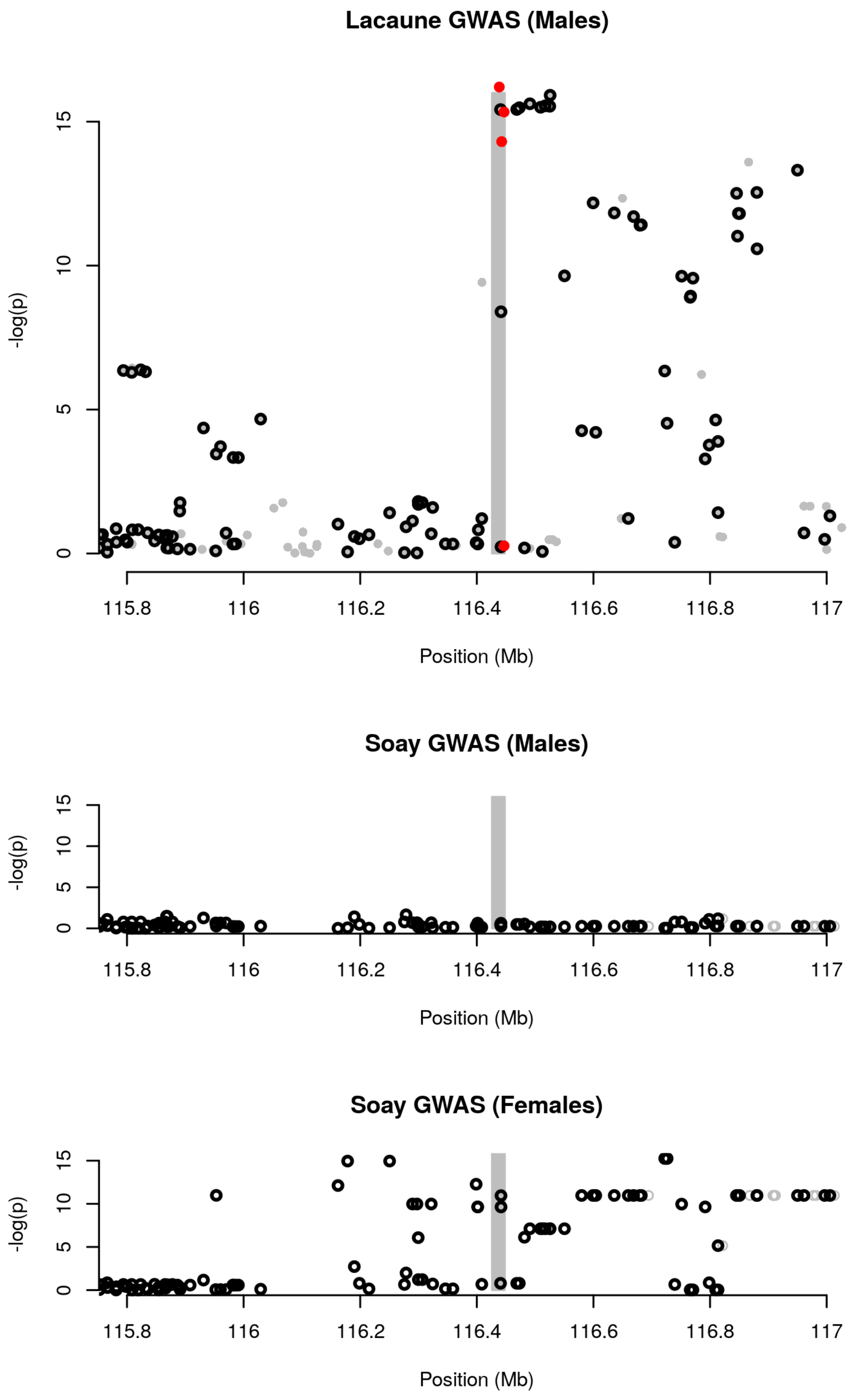


Figure SM16: Comparison of GWAS results in Lacaune and Soay at the distal end of OAR6