Examples for MarSuPial usage

Kevin Wei

Examples are provided to illustrate how to use `MarSuPial` to infer recombination rates, simulate crossovers and WGS, and infer allele frequency from marker selected pool count data.

Installation

Make sure the package is loaded

```
library("MarSuPial")
```

Generate the expected X chromosome recombination rate

We will first use the recombination rate from Fiston Lavier et al 2010 to generate genetic distance from an arbitray locus of 3Mb. We will first generate a function called rate_fun with the map function:

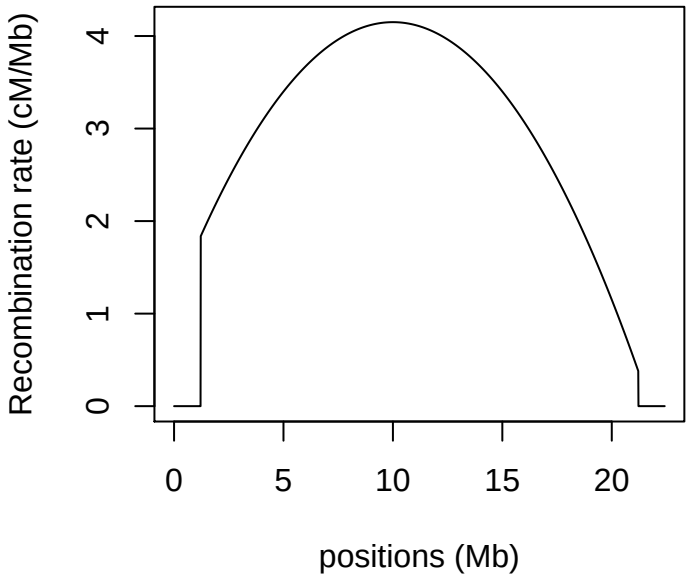
```
rate_function <- function(x){-0.03*x^2 + 0.6*x + 1.15}
```

Then, we will generate the chromosome wide recombination rate and genetic distance in 0.01Mb windows. The size of the chromosome and start and end of the rates are provided.

```
example <- r2d.locus(rate_function, 3, pos = 0.01, size = 22.422827, start = 1.22, end = 21.21)
```

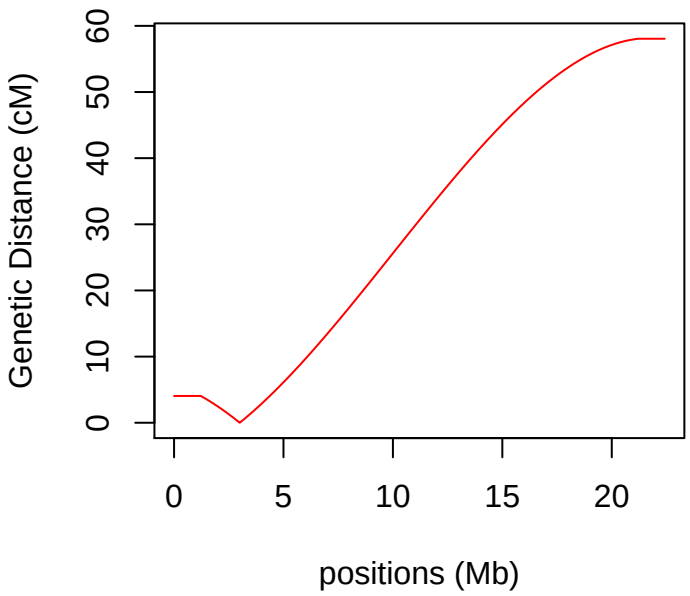
Let's see what the function looks like.

```
plot(example$pos, example$r, type = "l",
         ylab = "Recombination rate (cM/Mb)", xlab = "positions (Mb)")
```



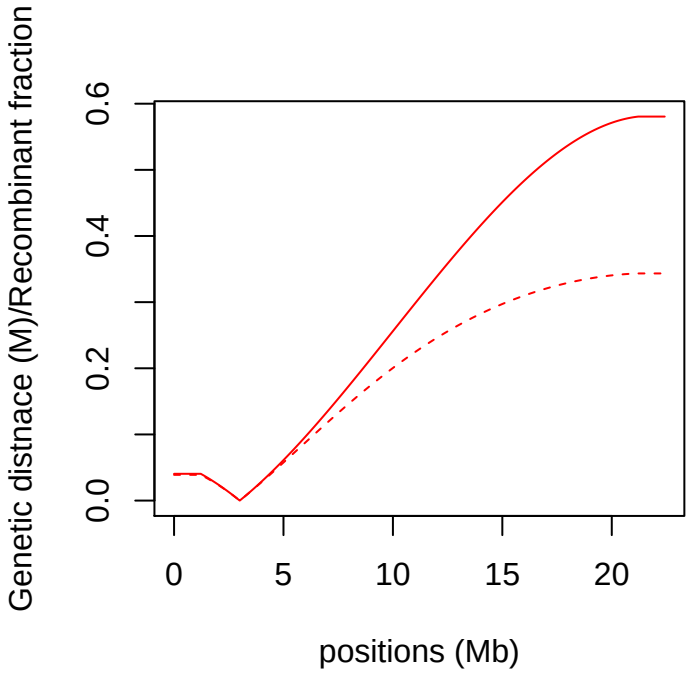
What about the genetic distance:

```
plot(example$pos, example$d, type = "l", col = "red",
         ylab = "Genetic Distance (cM)", xlab = "positions (Mb)")
```



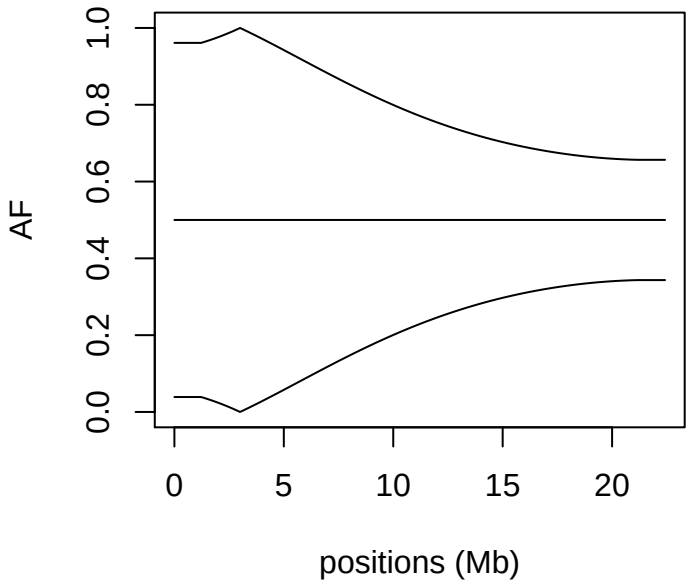
Let's convert the genetic distance to recombinant fraction using the haldane mapping function. Remember, the unit for the mapping function is morgan while the unit is currently cM.

```
exD <- d2D(example$d/100,  method = "haldane")
plot(example$pos, example$d/100, type = "l", col = "red",
         ylab = "Genetic distnace (M)/Recombinant fraction", xlab = "positions (Mb)")
points(example$pos, exD, type = "l", lty = 2, col = "red")
```



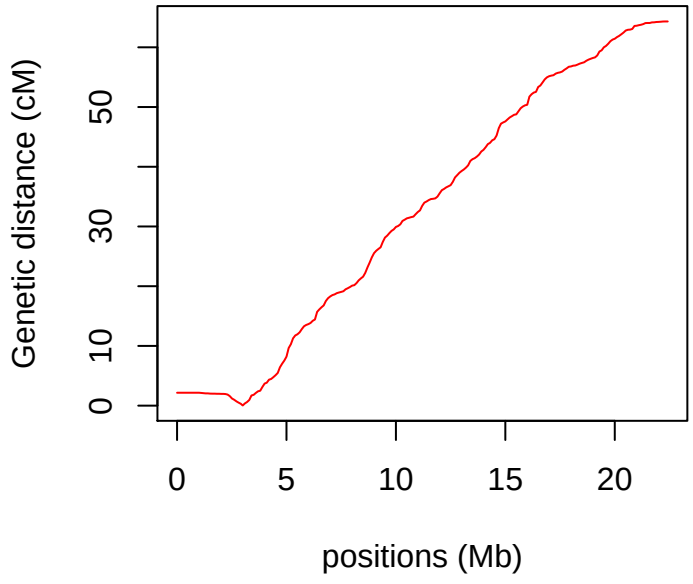
What will the allele frequency look like given different fitness differentials at the locus (3Mb) for male pol?

```
plot(example$pos, D2AF(exD,fitness = 1, ploidy = "haploid", mendel_rate = 0.5), type = "l",
        ylim = c(0,1), ylab = "AF", xlab = "positions (Mb)")
points(example$pos, D2AF(exD,fitness = 0, ploidy = "haploid", mendel_rate = 0.5), type = "l")
points(example$pos, D2AF(exD,fitness = -1, ploidy = "haploid", mendel_rate = 0.5), type = "l")
```



Let's try using estimated recombination rates from Comeron et al. 2012. Make sure the bed file is in your working directory.

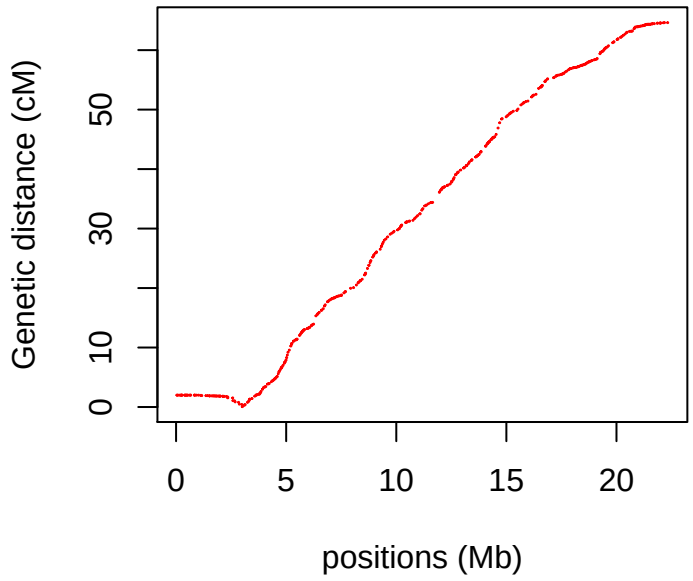
```
com <- read.table("comeron_X.bed", header = F, sep = "\t")
example2 <- r2d.locus(rate = com, l = 3, pos = 0.01, size = 22.422827)
plot(example2$pos, example2$d, type = "l", col = "red",
        ylab = "Genetic distance (cM)", xlab = "positions (Mb)")
```



We can also use any position instead of 0.01 windows!

```
positions <- sort(runif(n = 500, 0, 22.422827)) ## generates 500 positions randomly
example3 <- r2d.locus(rate = com, l = 3, size = 22.422827, pos = positions)
plot(example3$pos, example3$d, col = "red", pch = 20, cex = 0.1,
```

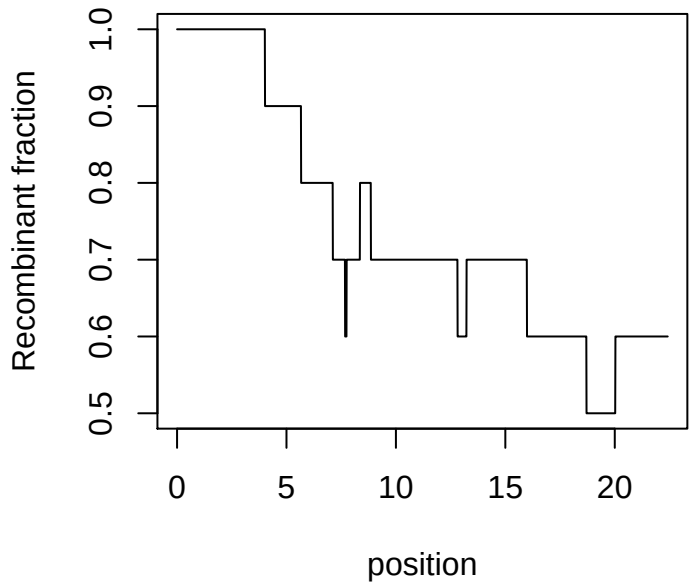
```
        ylab = "Genetic distance (cM)", xlab = "positions (Mb)")
```



Simulate crossovers and whole genome sequencing

So, now we have a r2d object (example, example2, and example3), let's try to simulate crossovers with them. We will first generate a pool of 100 individuals and observe the allele frequnecy in them.

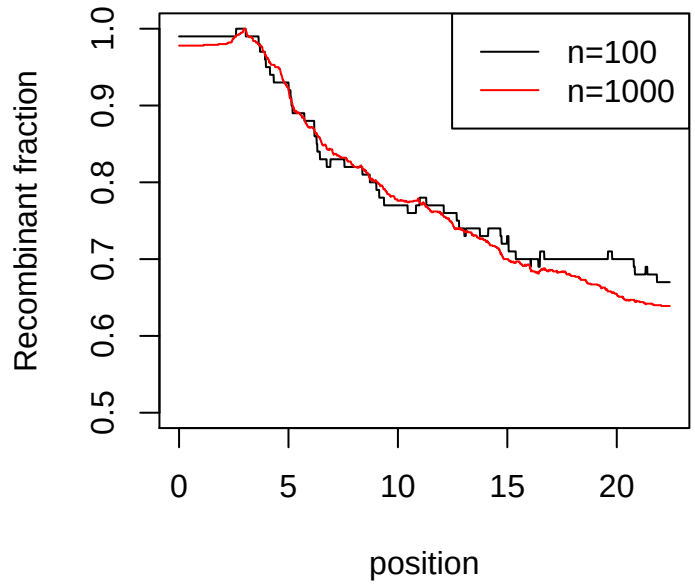
```
simmatrix <- COsim.ind(r2d = example2, n = 10, fitness = 1)
D <- sapply(1:ncol(simmatrix), function(x){sum(simmatrix[,x])})/10
plot(example2$pos, D, type = "l", ylim = c(0.5,1),
        xlab = "position", ylab = "Recombinant fraction")
```



Yikes, that doesn't look good. Let's increase the pool size!

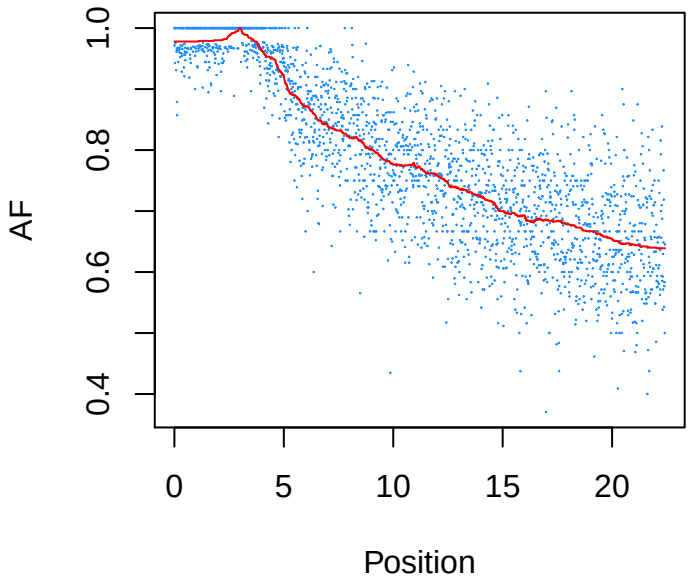
```
simmatrix <- COsim.ind(r2d = example2, n = 100, fitness = 1)
D <- sapply(1:ncol(simmatrix), function(x){sum(simmatrix[,x])})/100
```

```
plot(example2$pos, D, type = "l", ylim = c(0.5,1),
         xlab = "position", ylab = "Recombinant fraction")
simmatrix <- COsim.ind(r2d = example2, n = 1000, fitness = 1)
D <- sapply(1:ncol(simmatrix), function(x){sum(simmatrix[,x])})/1000
points(example2$pos, D, type = "l", ylim = c(0.5,1), col = "red")
legend("topright", legend = c("n=100", "n=1000"), col = c("black", "red"), lty = 1)
```



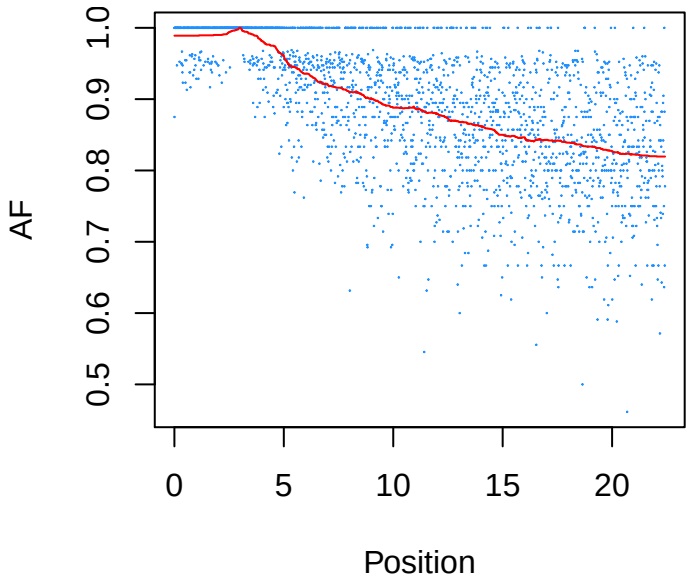
Now let's try simulating what the AF from 1000 individuals will look like if we sequneced it. Let's assume males are pooled and are sequenced to a coverage of 30x.

```
WGSsimulation <- AF2WGSsim(AF = D, dp = 30)
WGSAF <- WGSsimulation$counts/WGSsimulation$dp
plot(example2$pos, WGSAF, pch = 20, cex = 0.05, col = "dodgerblue",
         xlab = "Position", ylab = "AF")
points(example2$pos, D, type = "l", col = "red")
```



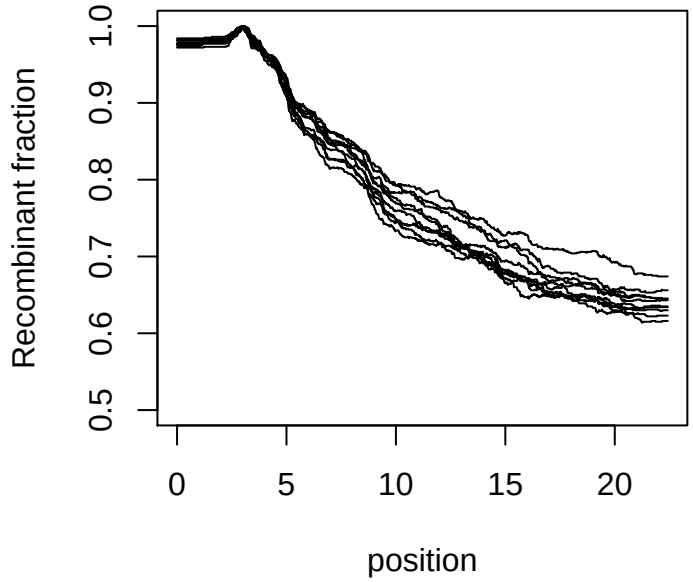
Now let's try females with coverage of 20x.

```
WGSsimulation <- AF2WGSsim(AF = (D + 1)/2, dp = 20)
WGSAF <- WGSsimulation$counts/WGSsimulation$dp
plot(example2$pos, WGSAF, pch = 20, cex = 0.05, col = "dodgerblue",
         xlab = "Position", ylab = "AF")
points(example2$pos, (D + 1)/2, type = "l", col = "red")
```



Generating individual's genotypes with COsim.ind is computationally expensive and makes it hard to simulate large numbers. Here we are generating 10 pools of 1000 individuals:

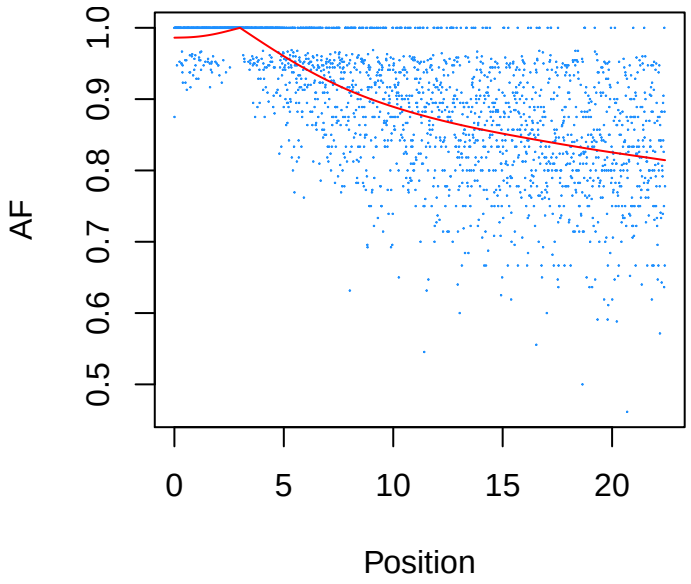
```
simmatrix <- COsim(r2d = example2, n = 1000, fitness = 1, trials = 10)
plot(example2$pos, 1:ncol(simmatrix), type = "n", ylim = c(0.5,1),
         xlab = "position", ylab = "Recombinant fraction")
for (i in 1:10) {
    points(example2$pos, simmatrix[i,], type = "l")
}
```



Estimating allele frequency from count data

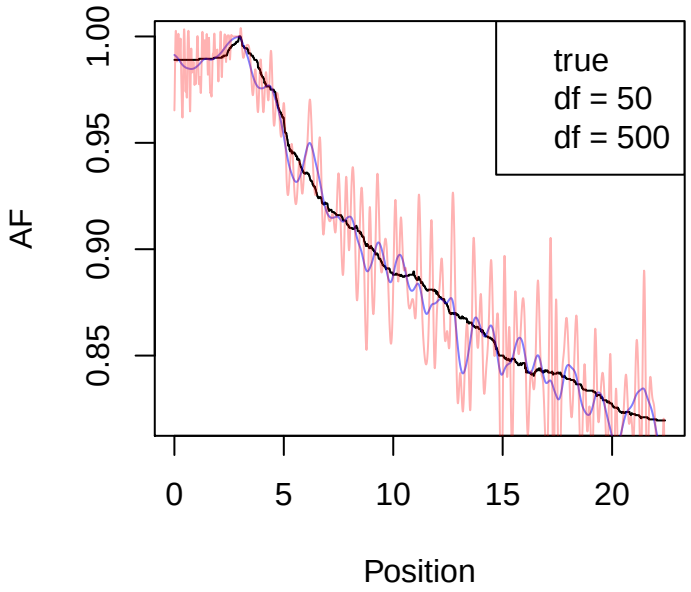
Let's see if we can recapitulate the recombinant fraction from the simulated counts.

```
curvefit <- AFfitsplines(sites = example2$pos, freq = WGSAF, depth = WGSsimulation$dp, locus = 3, locus
plot(example2$pos, WGSAF, pch = 20, cex = 0.05, col = "dodgerblue",
         xlab = "Position", ylab = "AF")
points(example2$pos, predictAF(AFfit = curvefit, example2$pos), type = "l", col = "red")
```



By default, the fitting method uses a cross validation technique to infer the smoothing resolution. But you can play with the degree of freedom to change the resolution. Higher degree of freedom means higher resolution but can also cause over fitting

```
curvefit <- AFfitsplines(sites = example2$pos, freq = WGSAF, depth = WGSsimulation$dp, locus = 3, locus
plot(example2$pos, (D + 1)/2, type = "l",
         xlab = "Position", ylab = "AF")
points(example2$pos, predictAF(AFfit = curvefit, example2$pos), type = "l", col = rgb(0,0,1,0.5))
curvefit <- AFfitsplines(sites = example2$pos, freq = WGSAF, depth = WGSsimulation$dp, locus = 3, locus
points(example2$pos, predictAF(AFfit = curvefit, example2$pos), type = "l", col = rgb(1,0,0,0.3))
legend("topright", legend = c("true", "df = 50", "df = 500"),
             col = c("black", rgb(0,0,1,0.5), rgb(1,0,0,0.3)))
```


AF
1.00
0.95
0.90
0.85
0
5
10
15
20
Position
true
df = 50
df = 500