R/mpMap Workshop

Part 2: Linkage Map Construction

Emma Huang

TAMU, 3 Sep. 2015

Plan

8:30-9:30

- Part 2: Linkage Map Construction (45 min)
    - Estimating recombination fractions
    - Grouping
    - Ordering
    - Refinement
- Exercises (10 min)
- Break/Questions (5 min)

Starting Point - Simulated Example

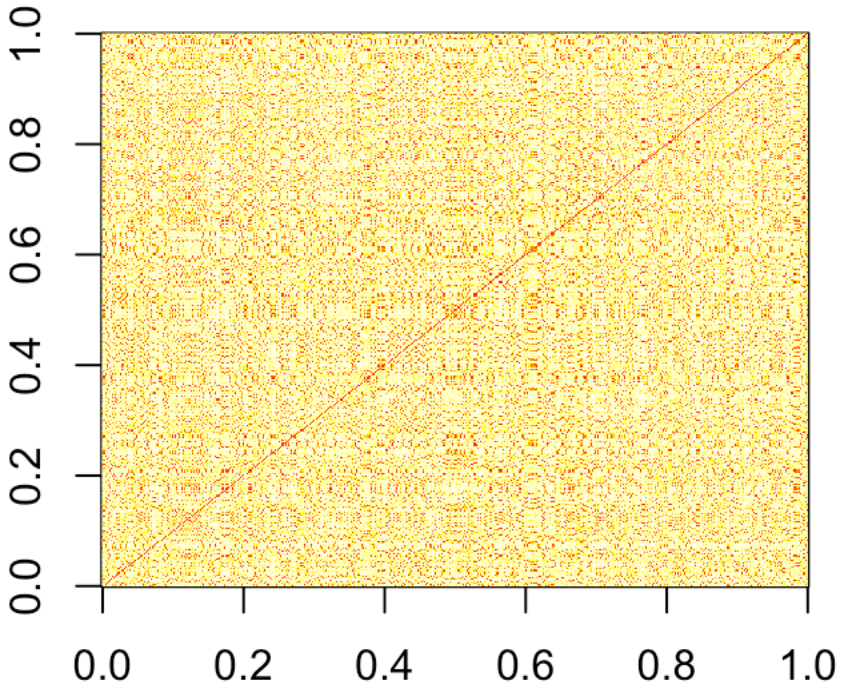
```
library(mpMap)
map <- sim.map(len=rep(100,5), n.mar=101, eq.spacing=T, incl
ped <- sim.mpped(4, 1, 1000)
dat <- sim.mpcross(map, ped)

## Randomize the order of the markers
ord <- sample(1:505)
randat <- subset(dat, markers=ord)
```

Step 1: Estimating RF

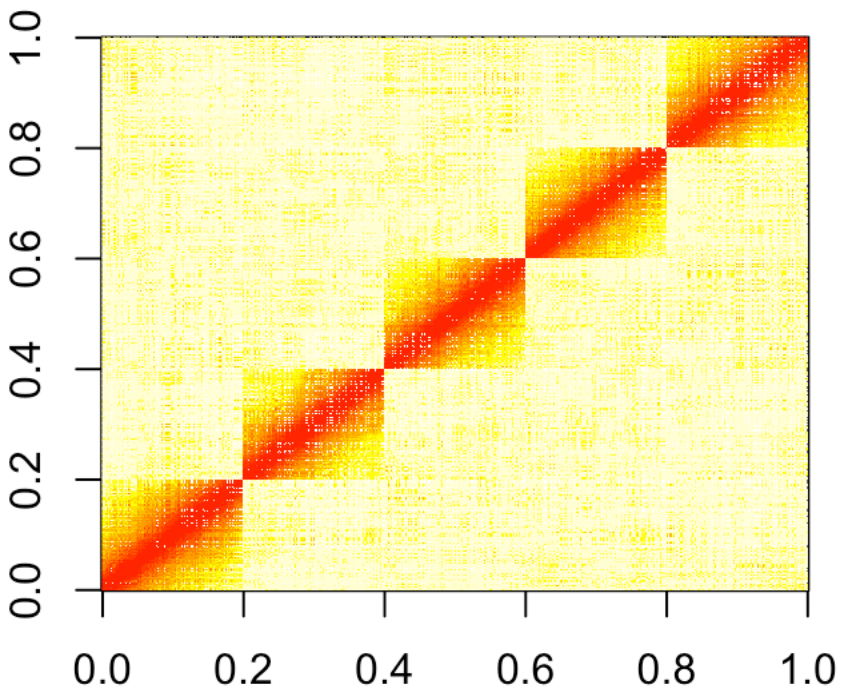
```
library(gdata)
datrf <- mpestrf(dat)
```

```
image(datrf$rf$theta)
```



End goal: RF for true order

```
image(dat1$rf$theta)
```



Estimating RF: Theory

Maximize the likelihood:

$$P(Y; r) = \Sigma_G P(Y|G)P(G; r)$$

- Y = observed genotypes
- G = underlying founder genotypes
- $P(Y|G)$ broken down by pairs of markers and individuals
    - takes values of 0 and 1
- $P(G; r)$ depends on pedigree
    - derived in Broman (2005)

Estimating RF: Practice

Maximize over a grid of recombination fraction values

On a larger scale:

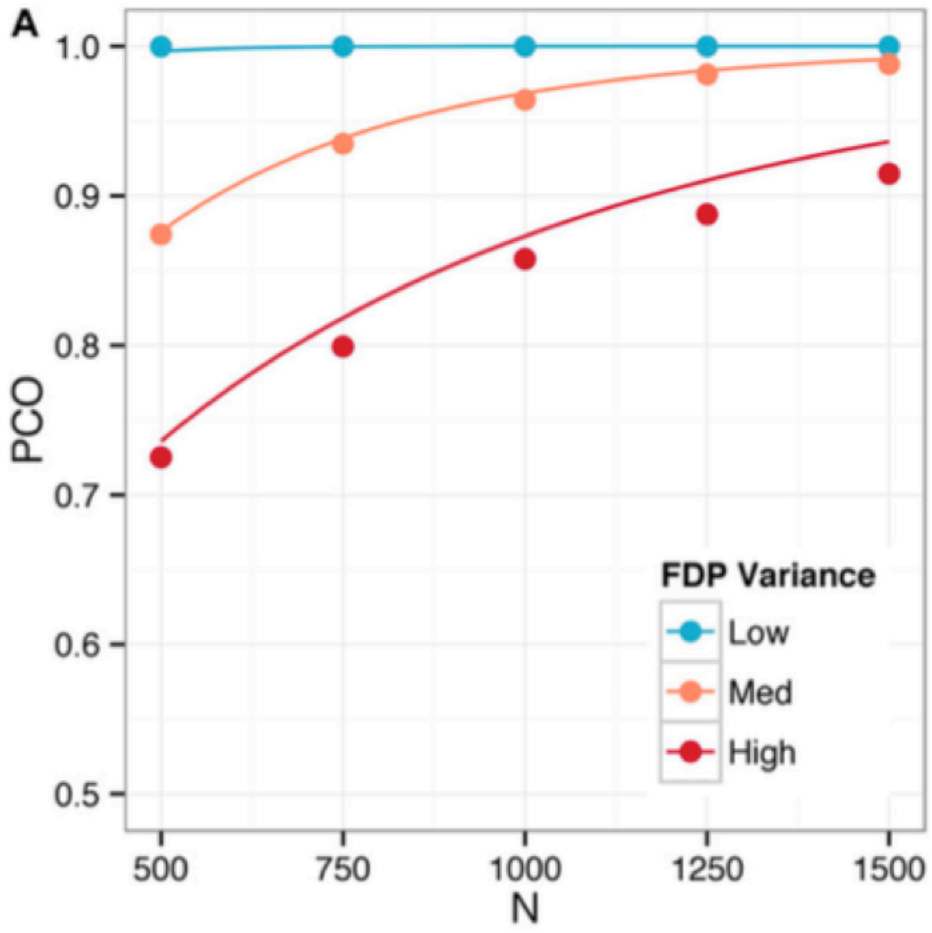
- GPU implementation
- MPI implementation
- Parallelized over all pairs of markers
- Time reduction from 2 hours to 25 sec
- Additional compilation options required

```
datrf <- mpestrf(dat, GPU=TRUE, mpi=TRUE)
```

An aside: Estimation Error

- Ahfock et al. (2014), Genetics 198:117-128
- Estimation of variability associated with RF estimates allows
    - computation of probability of correct order
    - hypothesis testing of marker ordering in triplets
    - characterization of uncertainty in map depending on marker density, founder distribution patterns, and sample size

Effect of estimation error on map



Step 1a: Binning Markers

- mpcollapse
  - groups markers with rf <= cutoff
  - within bins, forms haplotypes
  - imputes missing values where possible
  - recodes markers by matching to haplotypes
  - reduces to a single binned marker
- mpexpand
  - given full data, decompresses binned object
  - otherwise, produces expanded map

Example binning

```
mpbin <- mpcollapse(datrf)
```

```
## RF need to be re-estimated based on binned markers
```

```
dim(datrf$finals)
```

```
## [1] 1000  505
```

```
dim(mpbin$finals)
```

```
## [1] 1000  492
```

Binned markers

```
index <- which(duplicated(mpbin$bins$binMarkerName)
                | duplicated(mpbin$bins$binMarkerName, fromLa
head(mpbin$bins[index,])
```

```
##       MarkerName bin group binMarkerName
## D5M83      D5M83  12     1         C1B12
## D5M89      D5M89  12     1         C1B12
## D5M40      D5M40  26     2         C2B26
## D5M32      D5M32  26     2         C2B26
## D5M59      D5M59  46     2         C2B46
## D5M60      D5M60  46     2         C2B46
```

Binning process

```
cbind(datrf$founders[,i1], mpbin$founders[,i2])
```

```
##    D5M83 D5M89
## L1     1     0 1
## L2     0     0 2
## L3     1     1 3
## L4     0     1 4
```

```
cbind(datrf$finals[l1, i1], mpbin$finals[l1, i2])
```

```
##       D5M83 D5M89
## L1012     0     0  2
## L1018     0     1  4
## L1042     1     0  1
## L1048     1     1  3
## L1066    NA     0 NA
```

Step 2: Grouping Markers

- mpgroup
  - hierarchical clustering
  - metric based on rf and LOD
  - set number of groups to form

```
datgrp <- mpgroup(datrf, groups=5)
```

How well does grouping do?

```
table(datgrp$lg$groups)
```

```
##
##   1   2   3   4   5
## 101 101 101 101 101
```

```
chrtrue <- substr(names(datgrp$lg$groups), 2, 2)
table(chrtrue, datgrp$lg$groups)
```

```
##
## chrtrue   1   2   3   4   5
##       1   0   0   0   0 101
##       2 101   0   0   0   0
##       3   0 101   0   0   0
##       4   0   0 101   0   0
##       5   0   0   0 101   0
```

Step 3: Ordering Markers

- mporder
  - Seriation algorithm to order markers
  - Travelling Salesman heuristic solver
  - Minimize path length
  - Minimize Anti-Robinson events
- computemap
  - Estimates map positions based on pairwise rf
  - Positions based on neighborhood of markers

```
datord <- mporder(datgrp, type="2", criterion="AR_events")
```

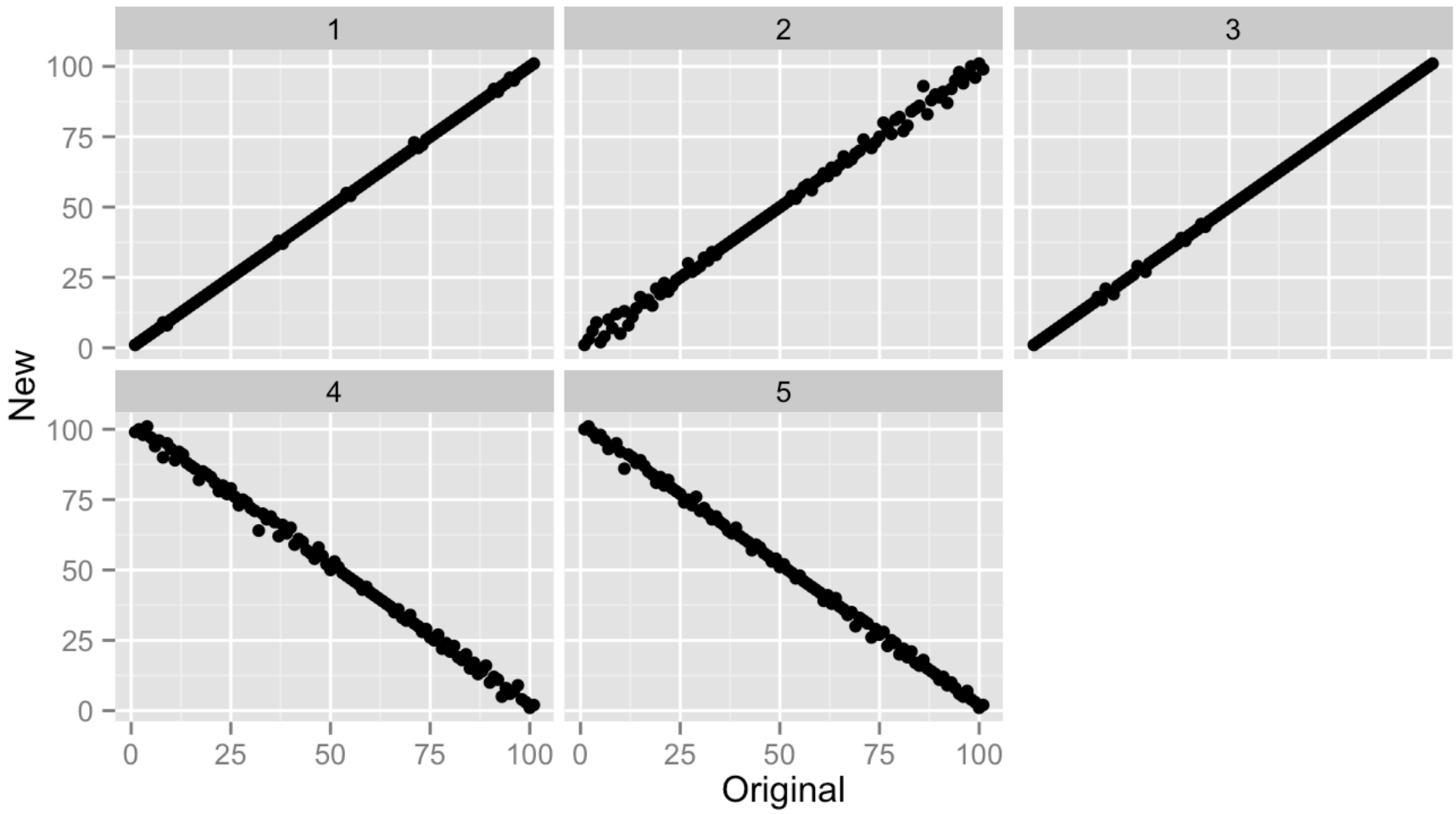
```
## Ordering chromosome 1...
## Ordering chromosome 2...
## Ordering chromosome 3...
## Ordering chromosome 4...
## Ordering chromosome 5...
```

```
datmap <- computemap(datord, maxOffset=1)
```

How well does automatic ordering do?

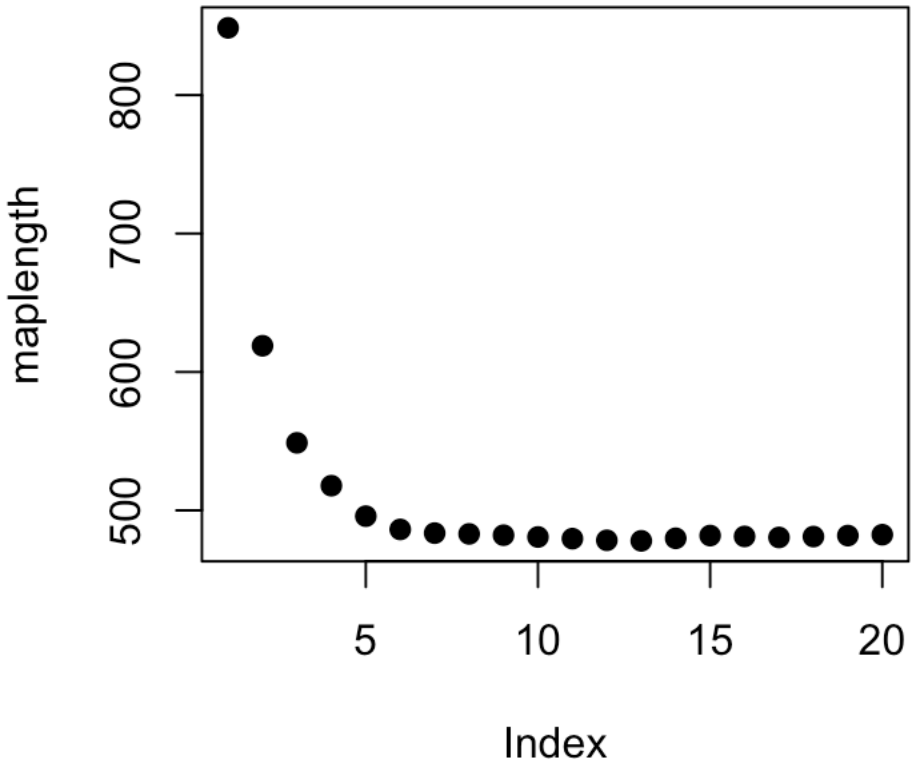
Comparison of orders for all chromosomes

```
ggplot(chrpl, aes(Original, New))+facet_wrap(~Chr)+geom_po
```



How to decide maxOffset?

Consider a variety - select one before major drop off



Step 4: Refinement - mpMapInteractive

- Visual comparison of different orders
- Interactive demo!

```
sub <- subset(datmap, chr=2)
newdat <- qtPlot(sub)
```

Step 4: Refinement - compare_orders

- Uses R/qtl engine to calculate XOs for different orders

```
sub <- subset(datmap, chr=1)

## Using map groupings for groups. Remove map object if you
```

```
nmrk <- ncol(sub$finals)
ord <- rbind(1:nmrk, c(7:1, 8:nmrk), c(2, 4, 1, 7, 5, 3, 6,
```

Compare_orders

```
ordxo <- compare_orders(sub, orders=ord, method="countXO")
```

```
## Using map groupings for groups. Remove map object if you
##  --Read the following data:
##   1000  individuals
##   101  markers
##   2  phenotypes
## Using map groupings for groups. Remove map object if you
## Using map groupings for groups. Remove map object if you
## Using map groupings for groups. Remove map object if you
##  --Read the following data:
##   1000  individuals
##   101  markers
##   2  phenotypes
## Using map groupings for groups. Remove map object if you
## Using map groupings for groups. Remove map object if you
##  --Read the following data:
##   1000  individuals
##   101  markers
##   2  phenotypes
```

Results

```
ordxo[, c(1:15, ncol(ordxo))]
```

```
##                                                    obligXO
## Initial 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15          3575
## 1       7 6 5 4 3 2 1 8 9 10 11 12 13 14 15          3583
## 2       2 4 1 7 5 3 6 8 9 10 11 12 13 14 15          3661
```

```
colnames(sub$finals)[1:8]
```

```
##    Chr11    Chr12    Chr13    Chr14    Chr15    Chr16    Chr17
##  "D2M1"   "D2M3"   "D2M6"   "D2M9"   "D2M2"   "D2M4"  "D2M10"
```

Final version

Refine order as much as possible based on diagnostics

```
datfinal <- qtPlot(datmap)
```

Recompute map with chosen maxOffset

```
datfinal <- computemap(datmap, maxOffset=5)
```

Step 5: Validation

- mapcomp
  - Compares two input maps
  - Reduces to common markers
  - Must have the same chromosome names
  - Can apply to maps or mpcross objects

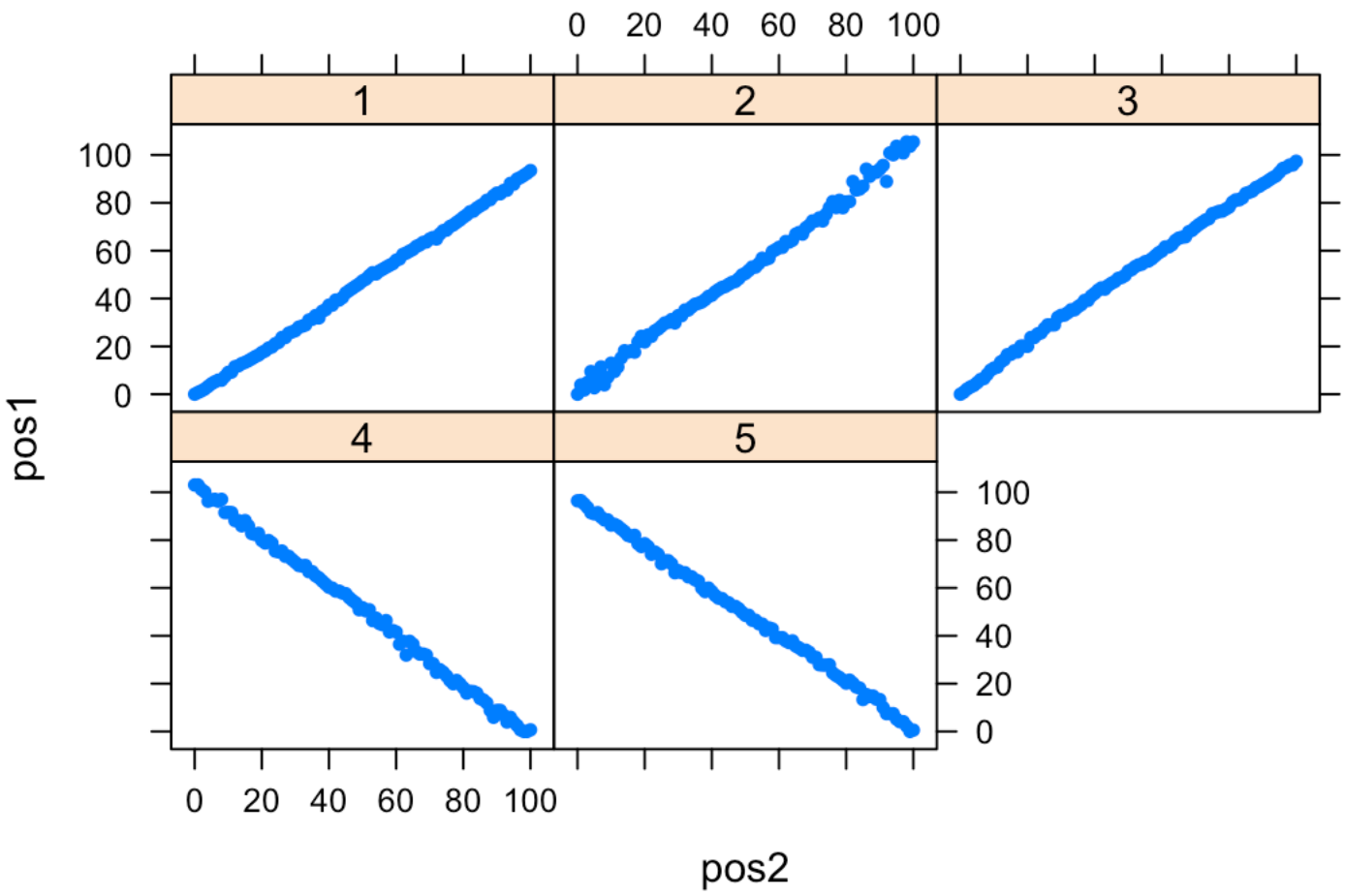
Validation summary

```
datfinal$map <- datfinal$map[c(5, 1:4)]
names(datfinal$map) <- names(dat$map)
mc <- mapcomp(datfinal, dat)
summary(mc)
```

```
## Number of markers in map1 is  505
## Number of markers in map2 is  505
## Number of common markers is  505
## Number of duplicated markers in map1 is  0
## Number of duplicated markers in map2 is  0
## Number of markers with differing chromosomes between maps
## Correlations between chromosomes are:
## 0.9996895 0.9974151 0.9992988 -0.9989776 -0.9992373
```

Validation plot

```
plot(mc)
```



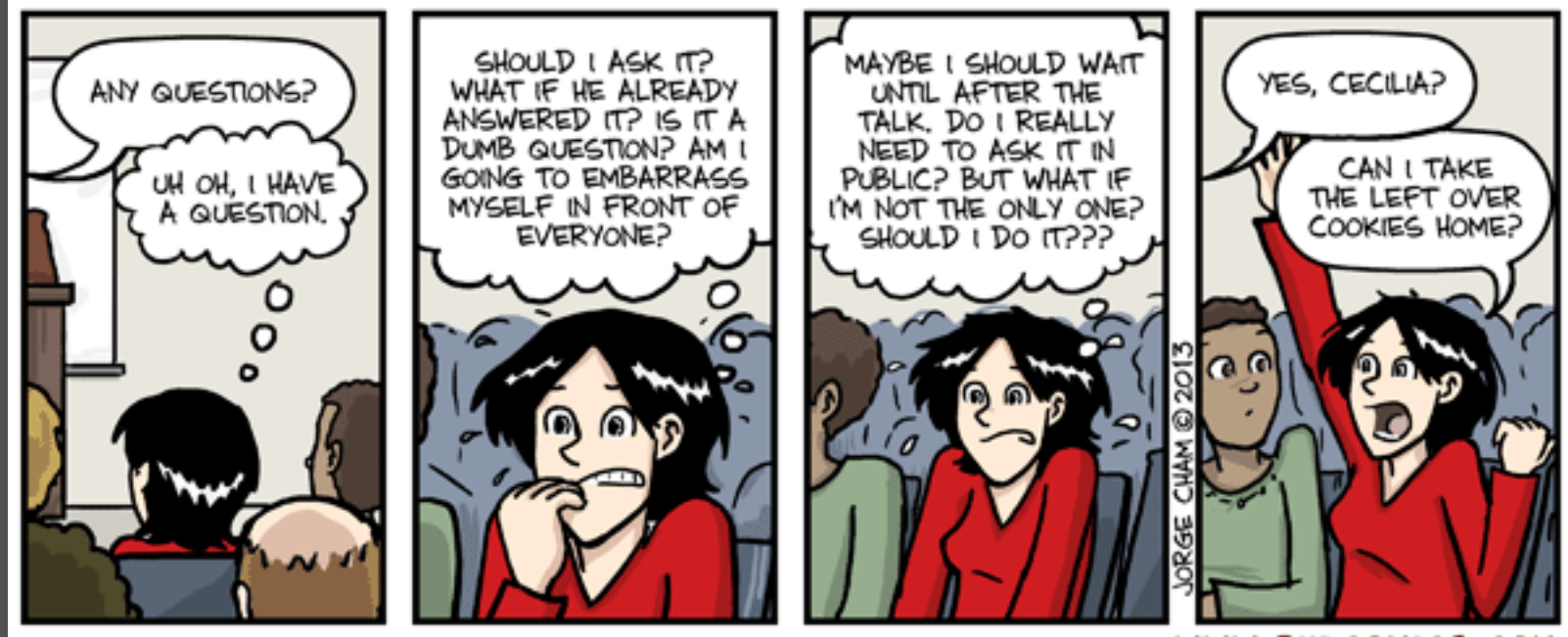
Exercises

Dataset sim2.1

- How many markers are there? per chr?
- Plot the genetic map
- Estimate recombination fractions
- Plot the heatmap
- Estimate the map using the correct order
- What's the length of each chromosome?
- What commands did I use to simulate this data?

Dataset sim2.2

- Correct the map


ANY QUESTIONS?
UH OH, I HAVE A QUESTION.
SHOULD I ASK IT? WHAT IF HE ALREADY ANSWERED IT? IS IT A DUMB QUESTION? AM I GOING TO EMBARRASS MYSELF IN FRONT OF EVERYONE?
MAYBE I SHOULD WAIT UNTIL AFTER THE TALK. DO I REALLY NEED TO ASK IT IN PUBLIC? BUT WHAT IF I'M NOT THE ONLY ONE? SHOULD I DO IT???
YES, CECILIA?
CAN I TAKE THE LEFT OVER COOKIES HOME?
JORGE CHAM © 2013
WWW.PHDCOMICS.COM

Questions

Exercise sim2.1

```
map <- sim.map(len=rep(150, 3), n.mar=51, eq.spacing=T, incl
ped <- sim.mpped(4, 1, 400)
dat2.1 <- sim.mpcross(map, ped)
save(dat2.1, file="sim2.1.RData")
```

Exercise sim2.2

```
load('sim2.1.RData')
fou <- apply(dat2.1$founders, 2, as.integer)
rownames(fou) <- rownames(dat2.1$founders)
colnames(fou) <- colnames(dat2.1$founders)
dat2.1$founders <- fou
fin <- apply(dat2.1$finals, 2, as.integer)
rownames(fin) <- rownames(dat2.1$finals)
colnames(fin) <- colnames(dat2.1$finals)
dat2.1$finals <- fin
dat2.1 <- mpestrf(dat2.1)
dat2.2 <- qtPlot(dat2.1)
dat2.2$lg <- list()
dat2.2$lg$all.groups=1:2
dat2.2$lg$groups <- c(rep(1, 51), rep(2, 102))
sub <- subset(dat2.2, markers=52:153)
sub <- mporder(sub, type="2", criterion="path_length")
dat2.2 <- subset(dat2.2, markers=c(colnames(dat2.2$finals)[1
                                   colnames(sub$finals)))
save(dat2.2, file="sim2.2.RData")
```