R / Bioconductor: A Short Course

James H. Bullard
Sandrine Dudoit
Division of Biostatistics, UC Berkeley
www.stat.berkeley.edu/~bullard
www.stat.berkeley.edu/~sandrine

Cuernevaca, Mexico
January 21-25, 2008

Exploring the HapMap Data Set

Whats a SNP?

- Two human DNA sequences are, on average, 99.5% identical. The majority of the remaining 0.5% of differences among individuals are Single Nucleotide Polymorphisms (SNPs).
- A **SNP** is a position in a genome at which two or more different bases occur in the population, each with a frequency $>1\%$.

```
CCAGCTAGT[T]AAGCTAGAA
CCAGCTAGT[C]AAGCTAGAA
```

- Nowadays there are more than 11 million validated SNPs for the human genome catalogued in public databases like dbSNP
- Due to their widespread distribution in the genome they are useful as genetic markers.

Terminology

- **Allele** is one of a number of alternative forms of the same gene occupying a given locus. If we are considering SNPs, an allele is one of two alternative forms. For the example above we have two alleles, often denoted as: T/C
- **Locus** (plural: loci) is the physical location of alleles on the chromosome.
- For each SNP, the combination of alleles a person has is called a **genotype**. Genotyping refers to any method that can uncover a genotype at a particular site or collection of sites. For example, the possible genotypes for the example SNP above are: TT, TC, CC.

Terminology

- A **haplotype** is a series of consecutive alleles on a particular chromosome. Haplotype variation exist due to a mechanism called recombination, which tends to occur preferentially in specific regions, thus recombination "creating hotspots" and "recombination cold spots."
- Example: Consider 2 loci, each with two possible alleles, the first locus being either A or a, the second locus being B or b. Then the genotype of an individual has 4 possible haplotypes: AB, Ab, aB, ab.

Haplotypes

- A haplotype is a combination of alleles at multiple linked loci that are transmitted together. Haplotype may refer to as few as two loci or to an entire chromosome.
- In a second meaning, haplotype is a set of SNPs on a single chromatid that are statistically associated.
- Through these associations, the identification of a few alleles of a haplotype can unambiguously identify the other polymorphic sites in the region.
- Such information is deemed valuable for investigating the genetic origin of common diseases and is collected by the International HapMap Project.

The HapMap Project

- The International HapMap Project is an organization which goal is to develop a haplotype map of the human genome, which will describe the common patterns of human genetic variation.
- The project is a collaboration among researchers at academic centers, non-profit biomedical research groups and private companies in Canada, China, Japan, Nigeria, UK and the United States.
- Populations sampled (n = 270 people)
  - 30 adult-and-both-parents trios from Ibadan, Nigeria (YRI)
  - 30 trios of U.S. residents of European ancestry (CEU)
  - 45 unrelated individuals from Tokyo, Japan (JPT)
  - 45 unrelated individuals from Beijing, China (CHB)

The HapMap Project

- The International HapMap Project officially started with a meeting on October 2002. It comprised two phases, and the analysis of the entire dataset was published in October 2007.
- The HapMap is expected to be a key resource for researchers to use to find genes affecting health, disease and responses to drugs and environmental factors. The information produced by the project will be made freely available to researchers around the world.

HapMap Data

- The data is publicly available at the HapMap website. I have pre-processed a little of the data and included it in the data directory of the course.
- Files contain 2,000 HapMap genotyped SNPs for chromosome 19, in CEU, CHB, JPT and YRI populations.
- After we familiarize ourselves with the pre-processed data which I have provided we will use R to do a more extensive example where we fetch the data ourselves and look at a larger set of SNPs on chromosome 19.

Reading in The Data

Unfortunately, when dealing with R often it is difficult to read in data sets using read.table where $p >> n$, or there are significantly more columns than rows. For instance, try the following code:

```
> hapmaps <- list.files("../../data/hapmap",
+     pattern = "\\.dta", full.names = T)
> hapmapData <- lapply(hapmaps, function(x) {
+     read.table(x, row.names = NULL)
+ })
```

Compare that to the following code!

Reading in The Data

```
> hapmaps <- list.files("../../data/hapmap",
+     pattern = "\\.dta", full.names = T)
> hapmapData <- lapply(hapmaps, function(hm) {
+     a <- scan(hm, what = character(0),
+         skip = 1, quiet = TRUE)
+     nm <- scan(hm, what = character(0),
+         nmax = NSNP, quiet = TRUE)
+     a <- matrix(a, ncol = NSNP,
+         byrow = T)
+     b <- as.data.frame(a)
+     colnames(b) <- nm
+     b
+ })
> names(hapmapData) <- c("CEU", "CHB",
+     "JPT", "YRI")
```

Looking at the Data

- The following situation occurs quite frequently in R - We want to look at the data which we just read in. We recall the following functions: head, tail, and very importantly in this case the subset operators "["
- Here we have data for the following SNP: rs2193502

	V1
AA	1
AG	16
GG	73

Table 1: CEU

Looking at the Data

	V1
AA	24
AG	17
GG	4

Table 2: CHB

	V1
AA	19
AG	15
GG	11

Table 3: JPT

Looking at the Data

	V1
CC	57
CT	30
NN	1
TT	2

Table 4: YRI

Data Storage

- The HapMap data has been stored as a data.frame where the SNPs are in the columns and the subjects are in the rows.
- This is the typical format. It is normal to store data as the independent observations in the rows and the (potentially) dependent observations on each subject int columns.
- The reason for this is that typically in most traditional statistical analysis we have $n >> p$ that is we have a lot more subjects than covariates. This is important when we want to estimate a parameter proportional to each "covariate" (think of the β parameter vector in linear regression)

Data Storage

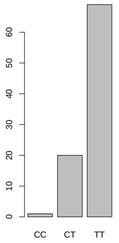
- As mentioned in the intro lecture data.frames are not really anything more than a list where each element in the list is a column. Therefore, we cannot store a factor in a row, because the columns don't share the same data type. This is confusing, but perform a transpose on one of the SNP matrices and try to do a table on the row.
- Finally, to be complete `R` stores matrices in column major format - which means that columns are contiguous in memory, this isn't really a big deal, but maybe important if you are dealing with huge data sets or if you need to write something in `C`.

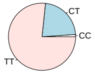
Visualizing the Data: Frequencies

- We are certainly interested in the different frequencies of genotypes in each population. We can display this using a barplot or a pie chart as we have seen before.
- Can we add some labels to the plots below to make things look nicer? How would we add text which told the percentages directly in either the pie chart or the barplot? (hint: look at the help page for pie and barplot)

```
> par(mfrow = c(1, 2))
> barplot(table(hapmapData[[1]][,
+     900]))
> pie(table(hapmapData[[1]][, 900]))
```

Visualizing the Data: Frequencies



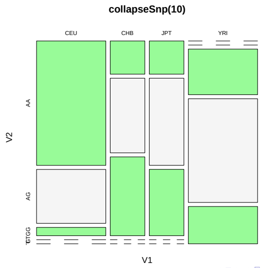


Visualizing the Data: Associations

Example

In this example we want to investigate the distribution of SNPs in certain populations. Todo this we are going to use mosaic plots (mosaicplot) and the function table. Write a function that takes a SNP name and then produces a mosaic plot for all the populations for that particular SNP. First, try to construct a data.frame containing the columns (SNP, subpop) and then on that two column data frame we can construct a table. In addition allow your function to take a second argument which determines which populations to include.

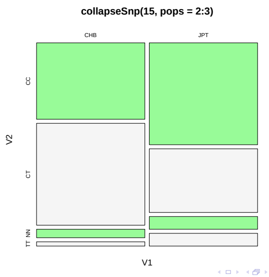
Visualizing the Data: Associations



Mosaic Plot

```
> mosaicplot(collapseSnp(15, pops = 2:3),
+     col = c("palegreen", "whitesmoke"))
```

Mosaic Plot



do.call

- The R function do.call is an essential tool when using R
- We can collapse a list using this function and aggregate the elements of a list using this function with a number of R functions, such as: cbind, rbind, c
- We can also use the Reduce to do similar things

```
> a <- list(a = c(1, 2), b = c(3,
+     4), c = c(5, 6))
> do.call("c", a)
> do.call("cbind", a)
> do.call("rbind", a)
> do.call("+", a)
> Reduce("+", a)
```

Statistical Tests

- A natural test to perform on our SNP data is a χ^2 test for no association between allele and population.
- We might want to determine whether a given loci is associated with a population.
- How can we take the code written above and perform a χ^2 test?

```
> chisq.test(collapseSnp(1, pops = 2:3))

Pearson's Chi-squared test


data:  collapseSnp(1, pops = 2:3)
X-squared = 3.9731, df = 2, p-value
= 0.1372
```

Lots of Tests

Example

Now that we have the machinery in place to make one statistical test we can make lots of tests. One thing we might be asked to do is obtain a ranking of the SNPs which are "most" important as a marker for population. One thing we might think to do is compute a χ^2 statistic across all SNPs and then rank the SNPs in order of the largest χ^2 values. Write a function which computes a χ^2 statistic for all 2000 SNPs and then ranks them. Also, compute a histogram of the χ^2 values. Our collaborators might ask us if this is a sensible thing to do, what might our response be? Along the way the functions order and sort might be useful.

Clustering and Classification

- Two common related problems in data analysis are classification and clustering.
- *Classification* is the setting where we are interested in assigning known class labels to subsets of our data based on knowledge of class-membership for some of our data.
- *Clustering* is the setting where we want to group together like subjects and potentially "learn" the clusters.
- In bioinformatics research clustering and classification continually appear - many methods exist to facilitate this.
  1. Clustering expression profiles of genes
  2. Classifying tumours (A natural classification problem is faced in the ALL data set)

The Data

- In cluster analysis we are going to be working with N observational units or N samples. For instance, experimental subjects or tissue samples.
- On each experimental subject we have made P observations. For instance, we have P gene expression measures. The P covariates could also be weight, height, race or other phenotypic variables. These covariates can be continuous, binary, or categorical.
- The goal is to assign the N subjects to one of K clusters where K is either chosen *a priori* or selected by an algorithm data-adaptively.

Distance

- Given our goal of constructing K clusters from our N observations there is an implicit notion of closeness between certain subjects. That is, we imagine that the each member of the k'th cluster will somehow be "close" to one another.
- This implies that we have a notion of distance or dissimilarity between our each of our N subjects.
- We construct a distance $d_{i,j}$ which will denote the distance between subject i and subject j. This number will have the following properties:
  1. Positive: $d_{i,j} \geq 0$
  2. Identity of Indiscernables: $d_{i,j} = 0 \iff i = j$
  3. Symetry: $d_{i,j} = d_{j,i}$
  4. Triangle inequality: $d_{i,j} \leq d_{i,k} + d_{k,j}$

Distance Matrices

- A distance matrix is a symetric matrix which measures the distance between each of our observational units or subjects. In the case of a microarray experiment we could compute a distance matrix in between all of our subjects based on all of their gene expression measures.

$$\text{distance}_{l_2}(i,j) = \sqrt{\sum_{n=1}^{N_{genes}} (\text{gene}_{i,n} - \text{gene}_{j,n})^2} \qquad (1)$$

In the microarray setting where we have continuous gene expression measures it may be sensible to use Euclidean distances between each subject. One important thing to note about this distance is that it implies that the measurements are on the same scale. We might be

Distance Matrices

interested in standardizing the measurements before hand in order to accomplish this.

- What might we do with categorical data? What if we have mixed data?

Example

Distance Matrices

Write an R function to compute the distance matrix for a given data.frame where the rows are subjects and the columns are covariates. This function should take two arguments: a data.frame and a function which actually computes the distance between *i* and *j*. Use Euclidean distance (1) for starters. To test your example, first, simulate some data in the following fashion:

```
> dta <- rbind(matrix(rnorm(100),
+     ncol = 2), matrix(rnorm(100,
+     2, sd = 2), ncol = 2), matrix(rnorm(100,
+     3, sd = 2), ncol = 2))
```

Then use the dist to test whether or not the distance matrix which you compute is the same as that computed by the R function - you'll need to look at the help file for dist.

Interlude: Timing

- As you might have noticed your distance function is considerably lower than R's distance function.
- There are a number of reasons for this. They have optimized, we have not. They have written in C we have not. They have a better algorithm than us.
- In any case, we might be interested in benchmarking this more precisely and to do this we can use R's function system.time.

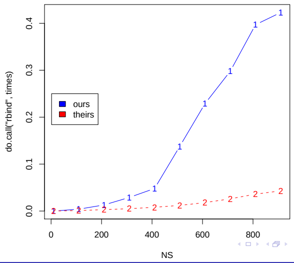
```
> system.time(dist(dta))

   user  system elapsed
  0.001   0.001   0.004
```

Interlude: Timing

Example

Use the system.time to compute a scaling plot like the one below. On the x axis is N and on the y axis is time from the system.time function.

Interlude: Timing



Distances Everywhere

- Distance matrices pop up quite frequently in Bioinformatics and machine learning in general.
- They are often used in phylogentics applications to infer the the likely evolutionary tree that gave rise to a set of species. In these applications the distances represent and evolutionary distance or dissimilariy between two species based on the sequence content of a particular gene or set of genes. Two well known methods are UPGMA and neighbor-joining.
- One must be quite careful when choosing particular distance metrics because certain metrics have certain undesirable properties. Euclidean distance is not scale invariant.

Take a look at the R function dist and see what distances are available.

Back to Clustering

- Now that we have our distance matrix we can get back to clustering. It is important to realize that we have to make $n(n-1)/2$ computations so we better do each computation really fast!
- Clustering methods are divided generally into two types:
  1. Partitioning : k-means, partitioning around medoids
  2. Hierarchical : Hopach, SOTA, diana
- First we will have a look at partitioning methods.

Partitioning method PAM

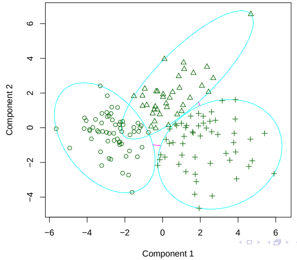
- PAM or partitioning around medoids (Kaufman & Rousseeuw 1990) is a very popular partitioning based clustering method which begin with a distance matrix and a number k indicating the number of clusters to divide the subjects into.
- The goal is to find a set of k medoids (k observations) which are representative of the remaining data points. These are subjects which are central in a group, like an average, but it is an actual data point.
- Formally, PAM seeks to minimize the following criteria:

$$\mathcal{M}_k \triangleq \underset{\mathcal{M}_k \subset x_1,\ldots x_n, |\mathcal{M}|=k}{\arg\min} \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(x_i, m) \qquad (2)$$

Partitioning method PAM

- PAM seeks to find a set of points which minimizes the sum of the distances to each point in that cluster, a point is in a cluster if it is closer to that cluster's medoid than any others.

Partitioning method PAM



K-means clustering

- K-means clustering is very similar to PAM however it does not use actual observations as the cluster centers, but rather an average of points within a given cluster.

$$\underset{\mu_k}{\arg\min} \sum_{k=1}^{K} \sum_{x_n \in \mathcal{C}_k} (x_n - \mu_k)^2 \qquad (3)$$

Here C_k denotes the kth cluster, and μ_k denotes the kth cluster mean. We are looking for the k values of μ which produce the lowest overall loss.

- One thing to note is that k-means is not as general as PAM in the sense that we cannot use arbitrary distance measures.
- Additionally, k-means is more sensitive to outliers than is PAM. Why?

Hierarchical Clustering

- The next class of clustering algorithms which we want to take a look at are hierarchical clustering methods.
- These take a distance matrix as we have seen before and construct a hierarchy or tree from the entries. At each leaf is an observation and the further away two observations are on the tree the more distant.
- Hierarchical clustering algorithms do not take a prespecified number of clusters which is often appealing when we don't know how many clusters are in our data.
- We can distinguish two types of hierarchical clustering algorithms **agglomerative** and **divisive**.

Hierarchical Clustering

- Agglomerative algorithms build a tree from the leaves up, whereas divisive algorithms build a tree from the root down.
- Tree based methods are often appealing because of the ease of visualization. See heatmap.

Clustering in R

- R has a number of clustering tools located in the cluster package.
  1. kmeans : Perform k-means clustering on a data matrix.
  2. pam : Partitioning the data around k-medoids. PAM takes a distance matrix and computes a clustering, alternatively it can take a data matrix and implicitly compute a distance matrix.
  3. agnes : Agglomerative hierarchical clustering.
  4. hclust : An agglomerative hierarchical clustering algorithm.
  5. diana : Computes a divisive hierarchical clustering.

A Simple Example

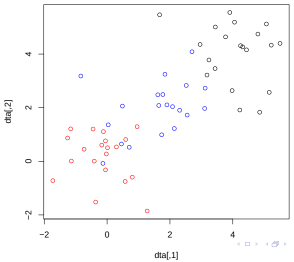
Below we make a "three" population example. In the example below each population contains 20 "subjects" with two measurements on each subject. We use both pam and kmeans to take a look at how the data can be clustered.

```
> library(cluster)
> N <- 20
> dta <- rbind(matrix(rnorm(N * 2),
+     ncol = 2), matrix(rnorm(N *
+     2, 2, sd = 1), ncol = 2), matrix(rnorm(N *
+     2, 4, sd = 1), ncol = 2))
```

First, it is a good idea to take a quick look at the data.

```
> plot(dta, col = rep(c("red", "blue",
+     "black"), each = 20))
```

A Simple Example



PAM and k-means

```
> pClust <- pam(dta, k = 3)
> kClust <- kmeans(dta, centers = 3)
> xtable(table(pClust$clustering))
```

	V1
1	25
2	15
3	20

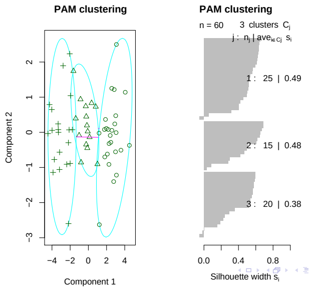
```
> xtable(table(kClust$cluster))
```

	V1
1	16
2	23
3	21

pam plot

```
> par(mfrow = c(1, 2))
> plot(pClust, main = "PAM clustering")
```

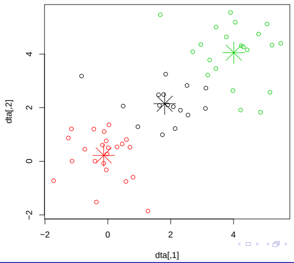
pam plot



kmeans plot

```
> plot(dta, col = kClust$cluster)
> points(kClust$centers, col = 1:3,
+     pch = 8, cex = 4)
```

kmeans plot



Assessing Performance

- Before we move on to more complicated examples and clustering methods we want to make sure that we understand how to assess performance of a particular clustering method.
- In this situation where we know the true class labels we can decide how many misclassifications we made and compute a misclassification rate.

```
> pClustMis <- sum(abs(table(pClust$clustering) -
+     N))/(20 * 3)
> kClustMis <- sum(abs(table(kClust$cluster) -
+     N))/(20 * 3)
> xtable(data.frame(pClustMis, kClustMis))
```

Assessing Performance

	pClustMis	kClustMis
1	0.17	0.13

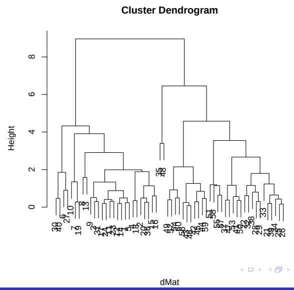
Hierarchical Clustering

We will use both hclust and diana to perform a hierarchical clustering of our sample data set.

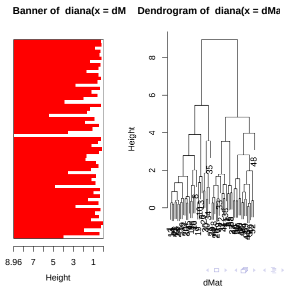
```
> dMat <- dist(dta)
> hClust <- hclust(dMat)
> dClust <- diana(dMat)
```

Take a quick look at these objects. What is their class? What do they contain?

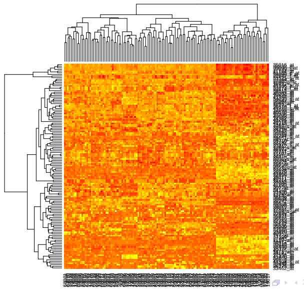
Plotting the Hierarchical Clustering: hclust



Plotting the Hierarchical Clustering: diana



Plotting the Hierarchical Clustering: heatmap



Partitioning vs. Hierarchical Methods

- Partitioning
  1. *Advantage* Provide clusters that satisfy an optimality criteria.
  2. *Disadvantage* Must specify K.
- Hierarchical
  1. *Advantage* Fast for agglomerative clustering.
  2. *Disadvantage* Can be rigid; imposes some structure on your data.

Clustering the ALL Data

Example

Clustering the ALL Data

Before we get back to the HapMap data we want to have a quick look at the ALL dataset which we have investigated previously. We want to see how well cluster analysis performs on this data. First, filter genes such that all genes retained are in the to 10% in terms of expression. Alternatively, filter genes based on the 10% most variable, finally, have a look at 1000 random genese. We want to determine if we can get PAM to cluster the data into the appropriate cancer types. Note: We have more than just B-Cell and T-Cell. First, see how well your method does on these two clusters then determine how well you do clustering into the more particular classifications:

```
10 Levels: B B1 B2 B3 B4 T T1 T2 ... T4
```

Things to think about are standardizing the data, scale and also which distances you might use.

Distances with Categorical Data

- After we have gone through our small analysis with the HapMap data and the ALL data set we are ready to start tackling clustering in the case of the HapMap data set.
- First, it should be noted that the HapMap data is hardly continous, so naively calculating a distance measure is probably not what we want to do.
- What kinds of distances can we think about using?

Clustering the HapMap Data

We first need to collapse the HapMap data into a single matrix. As it stands now we have a list with four data.frames.

Example

Write code to take these four data frames and combine them into a single matrix or data.frame. The rows should be the 270 subjects and the columns will be the SNPs. R can be a bit tricky when working with factors so you might want to first convert the data.frame to a matrix and then work with that instead. At the end we can convert back to a data.frame.

A Categorical Distance Measure

- One measure we could think about is the following:

$$\mathrm{d}_{\mathrm{SNP}}(i,j) = \sum_{k=1}^{N_{SNP}} |SNP_{i,k} \cup SNP_{j,k}| - |SNP_{i,k} \cap SNP_{j,k}| \quad (4)$$

Where $SNP_{i,k}$ is the set of alleles at that locus. For instance, we have TT and TG the intersection would be of size 1 and the union would be of size 2 - thus we would have 1 for this term in the sum. What properties does this distance satisfy?

1. Symmetric?
2. Positive?
3. Equal 0 $\iff SNP_{i,.} = SNP_{j,.}$?
4. Triangle inequality? $\mathrm{d}_{\mathrm{SNP}}(i,j) \leq \mathrm{d}_{\mathrm{SNP}}(i,k) + \mathrm{d}_{\mathrm{SNP}}(j,k)$

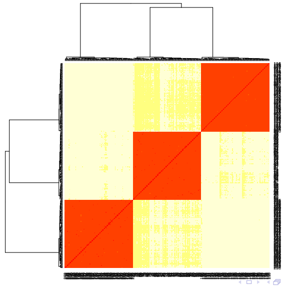
Running PAM on the Distance Matrix

Example

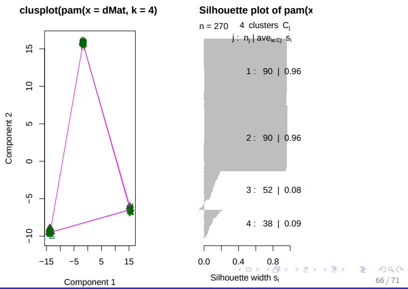
Running PAM on the Distance Matrix

Modify our previous distance function to compute our newly defined dissimilarity matrix. After generating the dissimilarity matrix do some summary statistics and brief exploratory data analysis to determine what it contains. For instance, knowing that the clusters are more or less broken down into the 90-90-90 are there clear patterns in the data? After you have satistfied yourself that you have generated the distance matrix correctly we want to run pam on the matrix to find a clustering. Make some plots of the output clustering also assess how well we did. What do we find? I recommend making the problem *much* smaller by selecting 10 of each subpopulation and maybe 200 SNPs to start off with! Once you have made your distance matrix try an image and/or heatmap plot.

The Distance Matrix



Silhoutte Plot



System Calls with R

In order to cover broadly the types of things which can be done with R, I wanted to present the code I used to get the HapMap data. This will be necessary in order to do some more interesting EDA and clustering analysis with the HapMap data. One problem with the data sets which I presented was the lack of chromosomal location for the SNPs - One thing which we might be really interested in investigating is the degree of linkage around particular loci. In order to do this we need to go get the HapMap data ourselves.

System Calls with R

```
> baseURL <- "http://www.hapmap.org/genotypes/latest_
> chr19 <- list(CEU = "genotypes_chr19_CEU_r22_nr.b36
+     CHB = "genotypes_chr19_CHB_r22_nr.b36_fwd.txt.g
+     JPT = "genotypes_chr19_JPT_r22_nr.b36_fwd.txt.g
+     YRI = "genotypes_chr19_YRI_r22_nr.b36_fwd.txt.g
> urls <- paste(baseURL, chr19, sep = "")
> sapply(urls, function(url) {
+     system(paste("wget", url))
+ })
> files <- list.files(pattern = ".txt.gz")
> sapply(files, function(f) {
+     system(paste("gunzip", f))
+ })
> infiles <- strsplit(files, split = "\\.gz")
> infiles <- list.files(pattern = ".txt")
```

System Calls with R

```
> readSubset <- function(files, ss = 1:54237) {
+     smple <- ss
+     i <- 1
+     l <- lapply(1:length(files),
+         function(i) {
+             a <- read.table(files[[i]],
+                 skip = 1)
+             snpNames <- as.character(a[,
+                 1])
+             snpLocations <- as.numeric(a[,
+                 4])
+             a <- a[smple, -(1:11)]
+             a <- t(a)
+             colnames(a) <- rownames(a) <- NULL
+             colnames(a) <- snpNames[smple]
```

System Calls with R

```
+             list(snps = as.data.frame(a),
+                 snpLoc = snpLocations[smple])
+         })
+     names(l) <- names(chr19)
+     return(l)
+ }
> snpTables.Chr19 <- readSubset(infiles)
> snpTables.Chr19.2000 <- readSubset(infiles,
+     sample(1:54237, size = 2000))
> save(snpTables.Chr19, file = "../../data/snpTables.
> load("../../data/snpTables.rda")
```

LD

The recombination hotspot data is available from the URL below, lets take a look at it for chromsome 19.
Recombination hotspots: recombination hotspots