Basic programing for R

GBIO0009

Kridsadakorn Chaichoompu

University of Liege

Important Links

- R Project

https://www.r-project.org/

- Bionconductor

https://www.bioconductor.org/

- RStudio

https://www.rstudio.com/

R : First impression

plot3D: Plotting Multi-Dimensional Data

- Functions for viewing 2-D and 3-D data, including perspective plots, slice plots, surface plots, scatter plots, etc. Includes data sets from oceanography

https://cloud.r-project.org/web/packages/plot3D/vignettes/plot3D.pdf

Basic commands

- q() To quit R environment
- x = 5 Assignment operator
- y <- 5 Assignment operator
- ls() To list objects in R environment
- ?ls() To check how to use a function
- getwd() To get a working directory
- setwd("New/Directory")
  To set a new working directory
- save(x,y,file="mydata.RData")
  To save objects as the R data file
- save.image(file="alldata.RData")
  To save all objects as the R data file
- load("mydata.RData")
  To load the R data file to the working space

Arithmetic operators

5+7	Addition
8-3	Subtraction
5*2	Multiplication
9/2	Division
(8+3)*4	Parentheses
2^4	Power
exp(4)	Exponential function
log(8)	Natural Logarithm
log10(8)	Logarithm in base 10
pi	Pi number

Logical operators

The values can be T, TRUE, F, FALSE

5<6	less than
5<=6	less than or equal to
5>6	greater than
5>=6	greater than or equal to
5==6	exactly equal to
5!=6	not equal to
!a	NOT a
a|b	a OR b
a&b	a AND b
xor(a,b)	a XOR b
isTRUE(a)	test if X is TRUE

Expression statement

- if (a == 5 && b > 5)
- if (a == 5 || b > 5)

Basic data types

class() - to check class of object

- Logical TRUE, T, FALSE, F
  ```
  class(TRUE)
  ```
- Numeric 2.4, 10, 200
  ```
  class(6.5)
  ```
- Integer 1L, 0L, -7L
  ```
  class(-8L)
  ```
- Complex 6 + 3i
  ```
  class(6 + 3i)
  ```
- Character 'hello', "I", "like", 'R'
  ```
  class('hello')
  ```
- Factor
  ```
  a = as.factor(1)
  a = as.factor('hello')
  class(a)
  ```

Vector

To create vectors

- `a = c(1,2,0,6.6,-2.5)`
- `b = c("a","b","c")`
- `c = c(F,T,TRUE,FALSE)`

Vectors and operators

- `a + 5`
- `a * 2`
- `c & TRUE`
- `c | FALSE`
- `1:5              Vector of 1 to 5`
- `c(a,1:5)         Concatenate 2 vectors`

Matrix

To create matrices

matrix(vector, nrow=r, ncol=c, byrow=FALSE)

- `a = matrix(1:12, nrow=3, byrow=F)`
- `b = matrix(1:12, nrow=3, byrow=T)`
- `c = matrix(runif(12,min=0,max=1), nrow=3, byrow=T)`
- `d = matrix(sample(c(TRUE,FALSE),12,replace=TRUE), nrow=3, byrow=T)`

Matrices and operators

- `a + 5`
- `a + b`
- `t(b)`   Transpose of matrix
- `a * b`   Element-wise multiplication
- `a %*% t(b)`   Matrix multiplication

Matrix (2)

To access elements of matrix

- `a[1,1]`
- `a[,1]`
- `a[1,]`
- `a[,2:3]`

To name row and columns

- `colnames(a) = c("a","b","c","d")`
- `rownames(a) = c("1","2","3")`

To combine 2 matrices

- `cbind(a,b)Combine by column`
- `rbind(a,b)Combine by row`

Data frame

“data.frame” is the collections of variables which share many of the properties of matrices and of lists

To create data.frame

- `x = c("Kris", "Jack", "Steve", NA)`
- `y = c(50,20,60,40)`
- `z = c(FALSE,TRUE,TRUE,FALSE)`
- `df = data.frame(x,y,z)`
- `colnames(df) <- c("name","paid","registered")`

Useful functions

- `df$name`
- `is.na(df$name) Check all elements if they are NA?`
- `anyNA(df$name) Is there any NA?`
- `df$paid * 1.21`
- `dim(df)         Check dimension`
- `df[which(df$name=="Kris"),]    Get specific row`

Data frame (2)

To name row and columns

- `colnames(df) = c("1","2","3”)`
- `rownames(df) = c("a","b","c","d")`

To combine 2 matrices

- `cbind(df,df)  Combine by column`
- `rbind(df,df)  Combine by row`

List

A collection of objects which can be in different length

```
• m = list(car=c("Toyota","Honda","Nissan"),
age=c(23,67),single=TRUE)
```

To access objects

```
• m$car
• m$age
• m[[1]]
• m[[2]]
```

Conversion functions

- `as.matrix(df)`
- `as.data.frame(a)`
- `as.list(1:5)`
- `as.integer(1:5)`
- `as.logical(c(0,1,1,0))`
- `as.factor(1:5)`

Concatenation functions

- `c()` To combine vectors
- `list()` To combine lists
- `cbind()` To combine matrices and data frames by column
- `rbind()` To combine matrices and data frames by row
- `paste("Hello","my","name","is","Kris")`
  To combine strings
- `paste0("Hello","my","name","is","Kris")`
  To combine strings without space

Trick to display text on screen

- `str = paste("Hello","my","name","is","Kris","\n")`
- `cat(str)` To display text
- `print(str)` To display all values as they are

Control Flow

- if(condition) ...
- if(condition) ... else ...

- for(variable in sequence) ...
- while(condition) ...

- break To stop iteration
- next To skip to next iteration

IF

Examples:

```
age = 10
if (age > 18){
   cat("Old\n")
}else{
   cat("Young\n")
}

age = 20
if ((age>18) && (age<25)){
   cat("Teenager\n")
}else{
   cat("Other type\n")
}
```

FOR

Examples:

```
for (i in 1:10){
  cat(paste(i,"\n"))
}

name =
c("Hello","my","name","is","Kris")
for (i in name)
  cat(paste0(i," "))
```

WHILE

Examples:

```
i = 0
while (i<5){
   print(i)
   i = i+1
}

i = 0
while (i<10){
   if (i>5) next
   print(i)
   i = i+1
}
```

Import delimited text file

- The formatted text files can be imported to R by these functions:
  - Read.table()
  - read.csv(), read.csv2()
  - read.delim(), read.delim2()
- Important parameters:
  - file : the name of input file
  - header : to indicate whether the first line contains the names of the variables or not
  - sep = the separator character

- Try to import *orange.csv*

  Download from the course website:

  http://bio3.giga.ulg.ac.be/archana_bhardwaj
- Example:

```
mydata=read.table(file="orange.csv",sep=",",header=TRUE)
head(mydata)
```

Export as delimited text file

- You can use these functions to export to file
  - write.table(x, file = "")
  - write.csv()
- Important parameters:
  - file : the name of input file
  - row.names : to indicate whether row names will be exported or not
  - col.names : to indicate whether column names will be exported or not
  - sep: the separator character
  - quote: to indicate whether text will be quoted (“hello”)

- Example:

```
write.table(mydata,file="newfile.csv",quote=T,sep="\t",
row.name=T,col.name=T)
```

Text display

To display text on screen

- `print(x, ...)`
- `cat(...)`

Concatenate variables

- `paste (...)`
- `paste0(...)`

Example:

- `dd <- 28`
- `mm <- "October"`
- `yy <- 2016`
- `cat(paste0(dd,mm,yy))`
- `cat(paste(dd,mm,yy,sep="-"))`

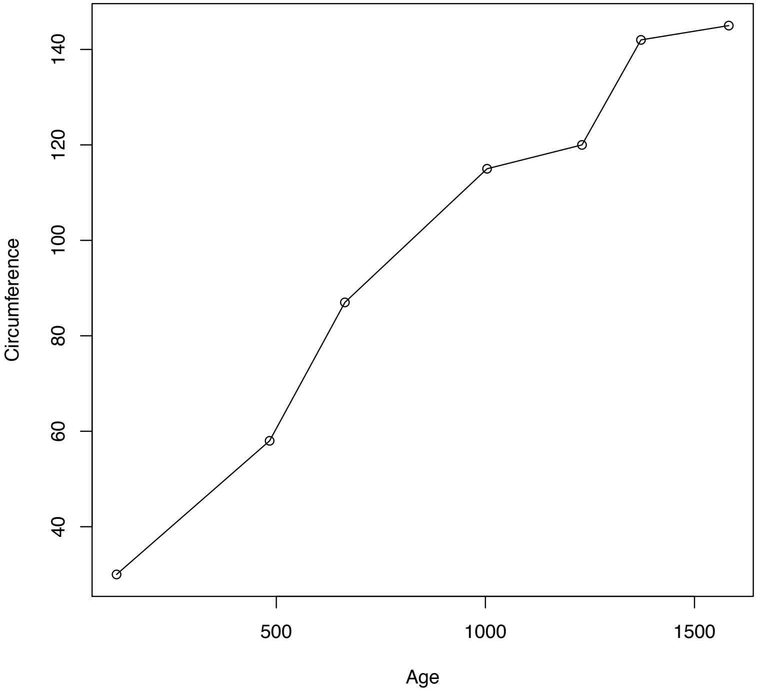
Plots

- Use plot() to create a simple XY plot
  - plot(rnorm(10))
- In the computing servers, we need to save plots as files and transfer to a local computer to view
  - pdf(file="./xyplot.pdf") ➔ create a pdf file in the current working directory
  - plot(rnorm(10))
  - points(rnorm(2),col="red") ➔ add 2 red dots to the plot
  - dev.off() ➔ close the graphical session, all graphical functions called before *dev.off()* will be saved to pdf file
- R also supports the other types of graphical files
  - Check: jpeg(), tiff(), png(), bmp()

Plotting for multiple data series

Single line:

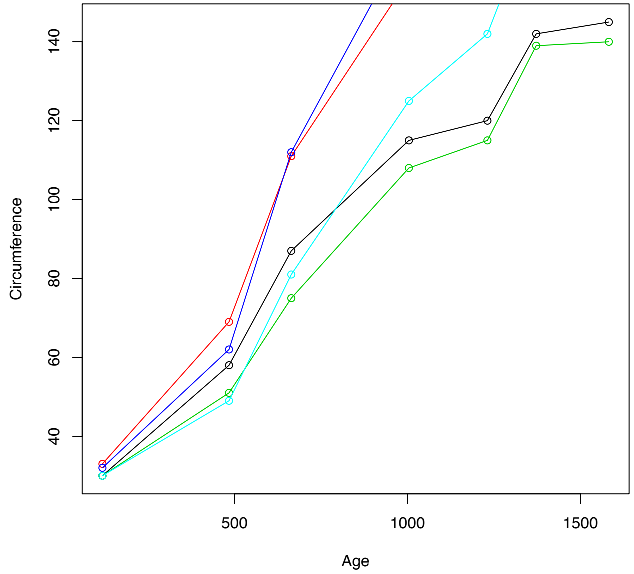
```
age=mydata$age[which(mydata$Tree==1)]
cir=mydata$circumference[which(mydata$Tree==1)]
plot(age,cir,type="o",xlab="Age",ylab="Circumference",
col=1)
```



Plotting for multiple data series (2)

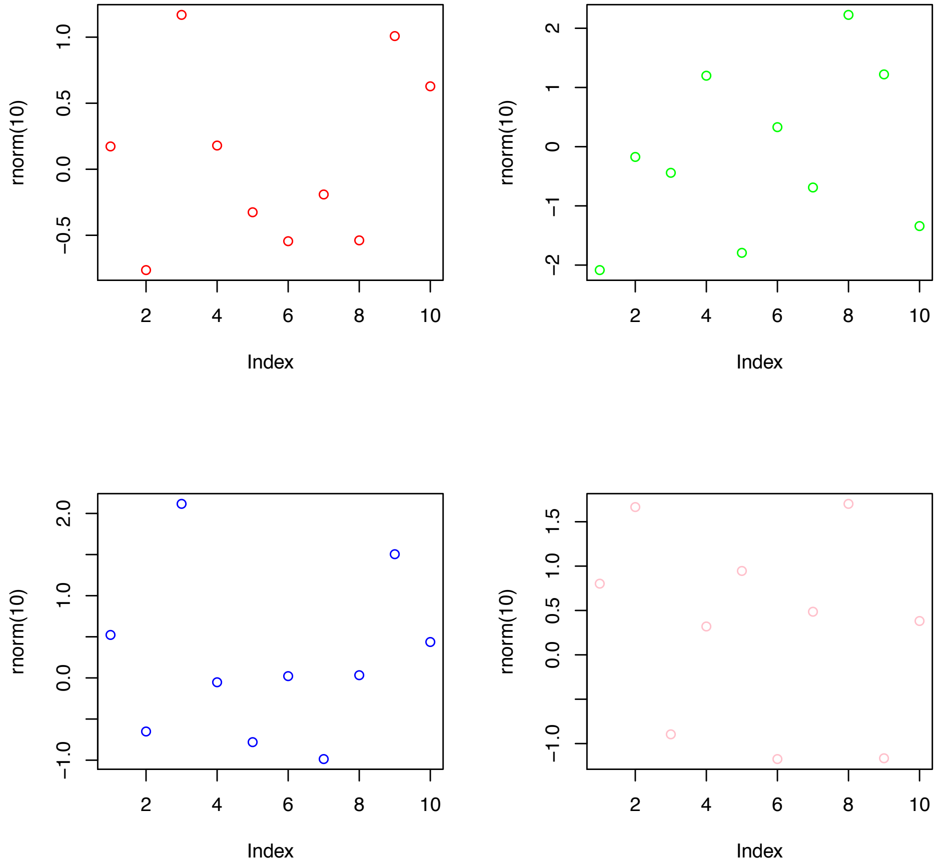
Add more lines:

```
trees=sort(unique(mydata$Tree))
subtrees=trees[-1]
for (item in subtrees){
  age=mydata$age[which(mydata$Tree==item)]
  cir=mydata$circumference[which(mydata$Tree==item)]
  lines(age,cir,col=item,type="o")
}
```



Multiple plots

```
par(mfrow=c(2,2))
plot(rnorm(10),col="red")
plot(rnorm(10),col="green")
plot(rnorm(10),col="blue")
plot(rnorm(10),col="pink")
```



Writing your own function

To define function:

```
f1 <- function(param1, param2, ... ){
   print(param1)
   return(param2)
}
```

Nested Function:

```
f2 <- function(p2,...){
   f1 <- function(p1,...){
      var1 <- log10(p1)
      return(var1)
   }
   var2 <- f1(p2)
   return(var2)
}
```

Population stratification

Population stratification

Population stratification is the presence of a systematic difference in allele frequencies between subpopulations in a population possibly due to different ancestry, especially in the context of association studies. Population stratification is also referred as population structure, in this context.

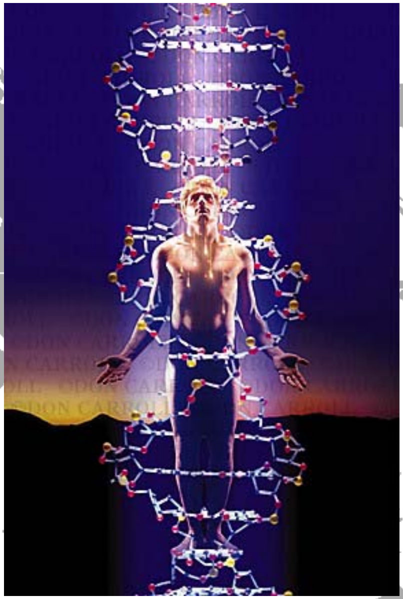
WIKIPEDIA
The Free Encyclopedia

Diversity

- Human
- Plants
- Animals
- Bacteria
- etc

Human Diversity

Medicine and Treatment



Public databases and tools

- The National Center for Biotechnology Information

https://www.ncbi.nlm.nih.gov/

- Ensembl

https://www.ensembl.org/index.html

- Gene Expression Omnibus

https://www.ncbi.nlm.nih.gov/geo/

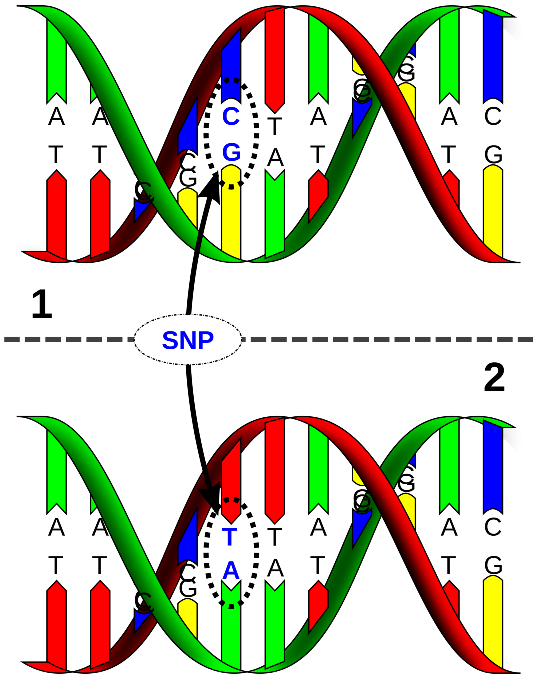
- UCSC Genome Browser

https://genome.ucsc.edu/

Single Nucleotide Polymorphisms (SNPs)

What are they?

How can we detect?



© David Kawai

SNP encoding

- Additive Encoding

Major Allele/Minor Allele	Encoding
A/A	0
A/B	1
B/B	2

- Try to load these files in to R working space
  - simSNP_rep1_data_numMark_rowInd_colVar.txt
  - simSNP_rep1_individuals_with_header.txt
- How many individuals?
- How many SNPs?

Principal Component Analysis (PCA)

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (PCs).

WIKIPEDIA
The Free Encyclopedia

PCA in R

- prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE, tol = NULL, ...)
- princomp(formula, data = NULL, subset, na.action, ...)
- eigen(x, symmetric, only.values = FALSE, EISPACK = FALSE)
- svd(x, nu = min(n, p), nv = min(n, p), LINPACK = FALSE)

library(rARPACK)

- svds(A, k, nu = k, nv = k, opts = list(), ...)
- eigs(A, k, which = "LM", sigma = NULL, opts = list(), ...)

PCA for SNPs

- X is the M x N matrix, where M is a number of individuals and N is a number of SNPs.

$$XX^T = UDV^T$$

U is the matrix of eigenvectors or PC scores.

$$B^T = D^{-1/2}U^TX$$

B is the factor loadings

$$PCs = X.B$$

Normalization

- Zero means

If X is a vector

$$M = X - \mathrm{mean}(X)$$

- Unit variance

$$Y = M / \mathrm{sd}(X)$$

- In R, it is more efficient to use apply() with mean() and sd()

Quality Control

- Select only founders

  PLINK option: --filter-founders
- Select only chromosome 1-22

  PLINK option: --not-chr 0,x,y,xy,mt
- Filter out SNPs in the Linkage disequilibrium (LD) blocks

  PLINK option: --indep-pairwise 50 5 0.2 (then use --extract to extract only the selected SNPs)
- Remove SNPs that disagree with the Hardy–Weinberg equilibrium (HWE) testing

  PLINK option: --hwe 0.001
- Allow individuals with call rate at least 95%

  PLINK option: --mind 0.05
- Filter out missing genotypes >2%

  PLINK option: --geno 0.02
- Remove SNPs with low minor allele frequency (MAF)

  PLINK option: --maf 0.05

Link: http://pngu.mgh.harvard.edu/~purcell/plink/

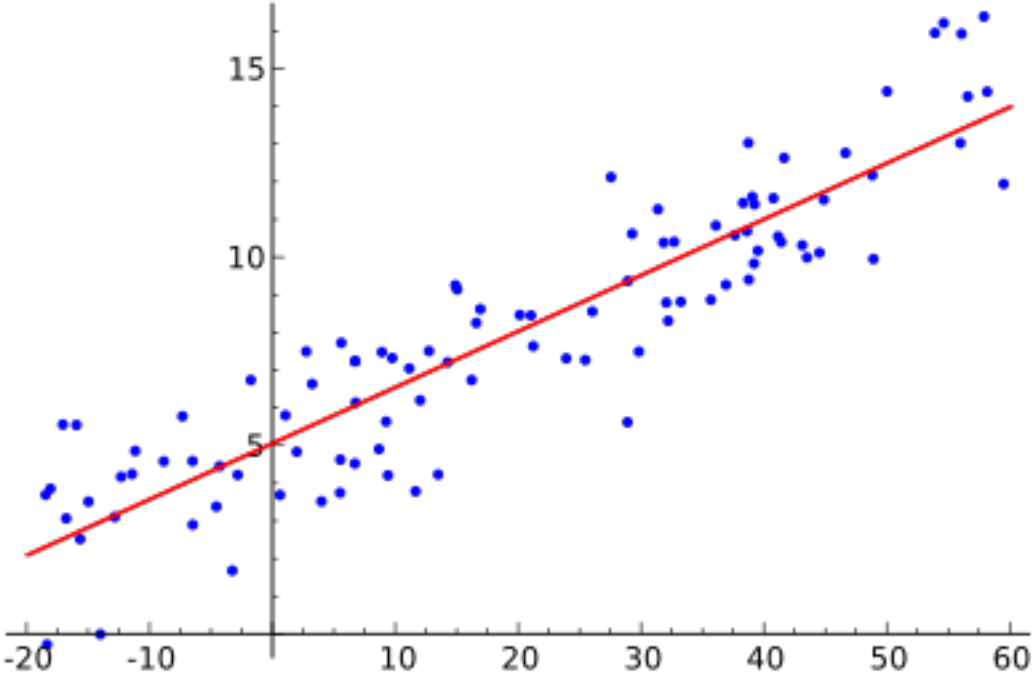
Exercise - PCA

- Calculate PCs for the example data - simSNP_rep1, more information
- Plot the first two eigenvectors
- Plot the first two PCs
- Is it possible to calculate PCs with PLINK? If so, please perform.
- Compare the PCs calculated from R and PLINK

Regression Models

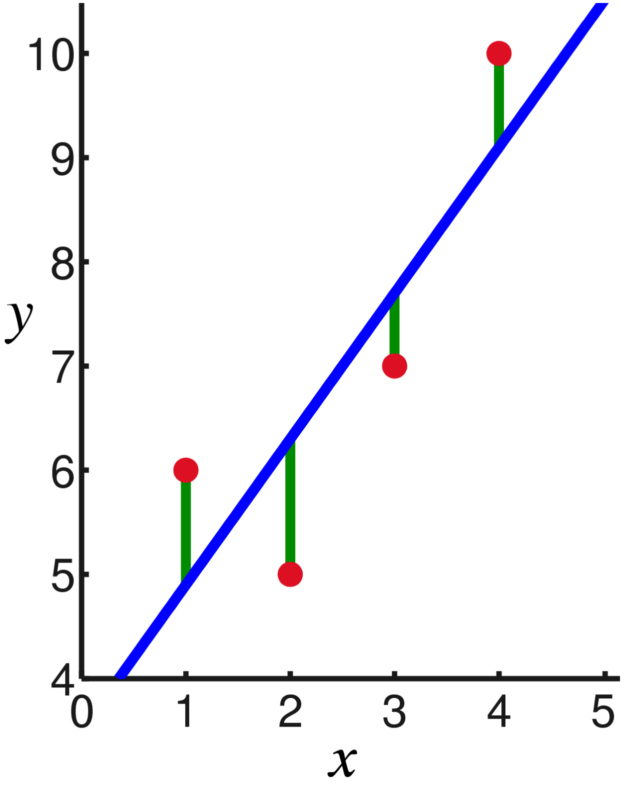
- Linear regression
- Logistic regression

Linear Regression



lm(formula, data)

Residuals



Residuals are the difference between any data point and the regression line

lm(formula, data)$residuals

Linear Regression in R

Linear models

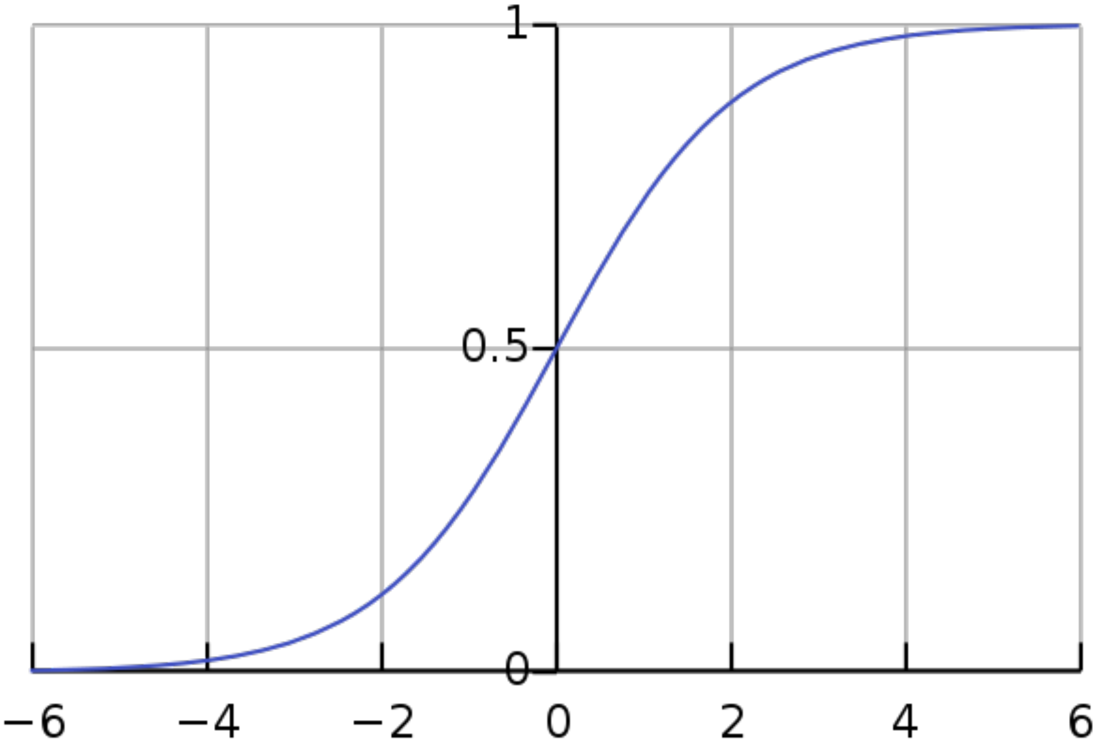
lm(formula, data, subset, ...)

Example in help page:

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
plot(lm.D9)
```

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html

Non-linear model: Logistic Regression



Generalized Linear Models - GLM

glm(formula, family = gaussian, data, weights, ...)

Example from help page:

```
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family =
binomial())
```

http://stat.ethz.ch/R-manual/R-patched/library/stats/html/glm.html

Models for GLM

glm(formula, family=familytype(link=linkfunction), data=)

Family	**Default Link Function**
binomial	(link = "logit")
gaussian	(link = "identity")
Gamma	(link = "inverse")
inverse.gaussian	(link = "1/mu^2")
poisson	(link = "log")
quasi	(link = "identity", variance = "constant")
quasibinomial	(link = "logit")
quasipoisson	(link = "log")

http://www.statmethods.net/advstats/glm.html

Exercise – Regression models

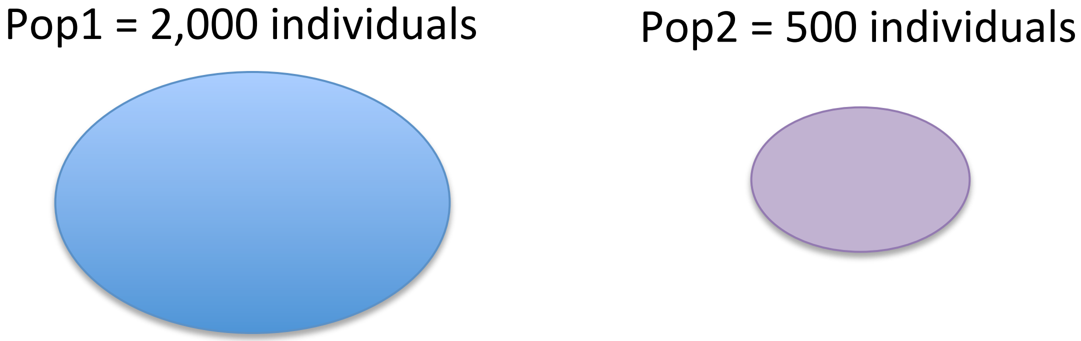
- Load the data from *simSNP_rep1.RData*, then perform linear regression using the following equation:

  SNP ~ PC1 + PC2 + PC3

- Create the plot of PC1 and PC2 using residuals
- Try with logistic regression

Fixation index (F_{ST})

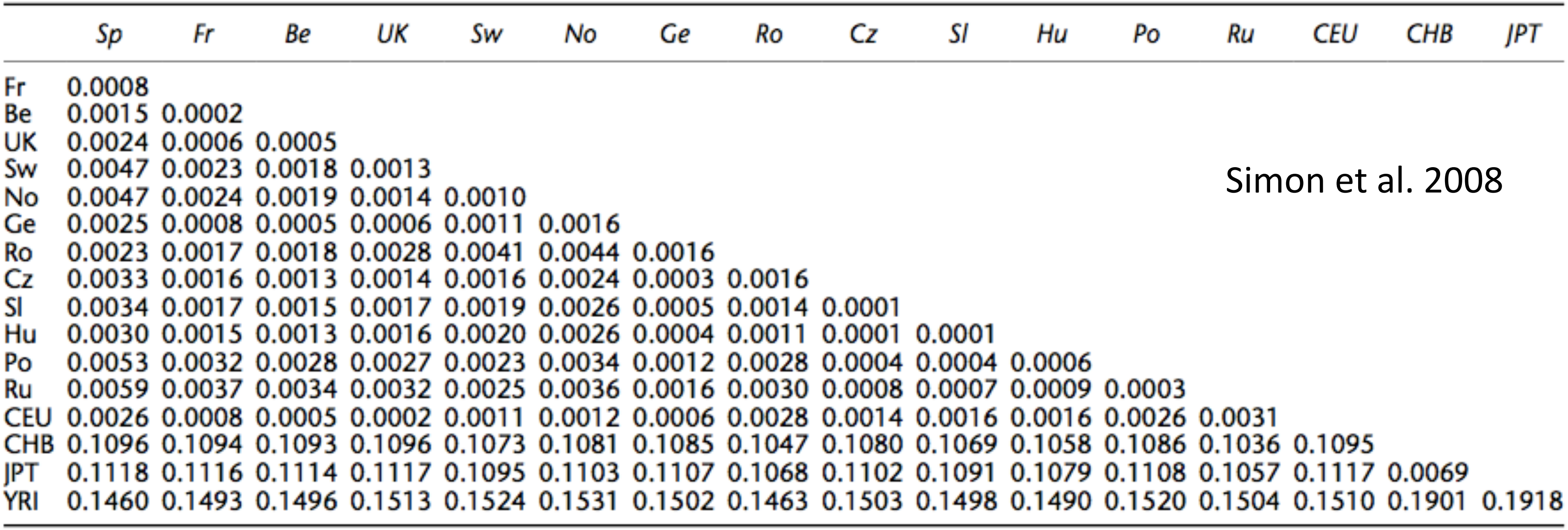
- F_{ST} can be used to describe a distance among population.
- F_{ST} can be biased due to the allele frequencies and the number of independent SNPs.



Pop1 = 2,000 individuals

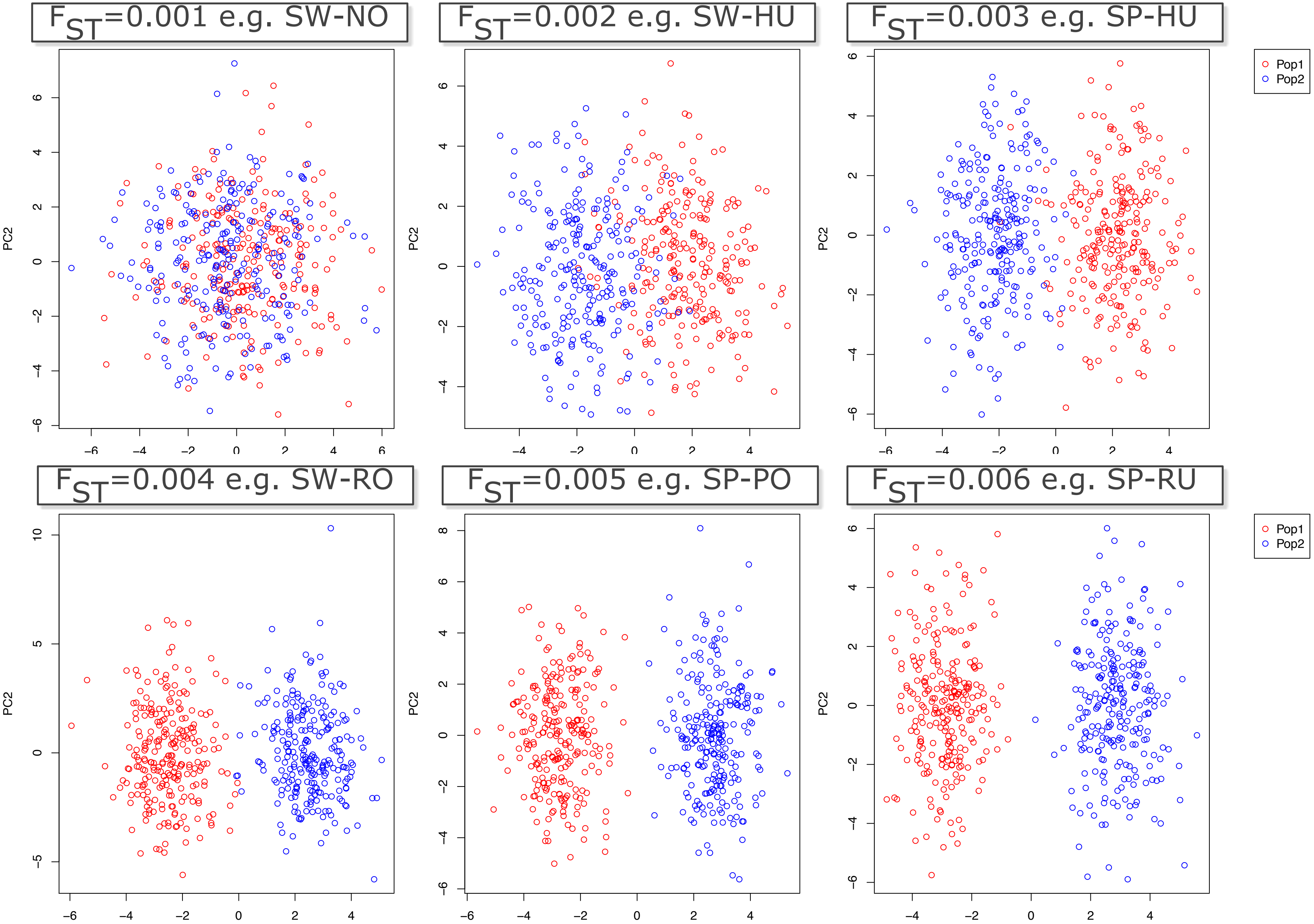
Pop2 = 500 individuals

F_{ST} among European populations



	Sp	*Fr*	*Be*	*UK*	*Sw*	*No*	*Ge*	*Ro*	*Cz*	*Sl*	*Hu*	*Po*	*Ru*	*CEU*	*CHB*	*JPT*
Fr	0.0008															
Be	0.0015	0.0002														
UK	0.0024	0.0006	0.0005													
Sw	0.0047	0.0023	0.0018	0.0013												
No	0.0047	0.0024	0.0019	0.0014	0.0010											
Ge	0.0025	0.0008	0.0005	0.0006	0.0011	0.0016										
Ro	0.0023	0.0017	0.0018	0.0028	0.0041	0.0044	0.0016									
Cz	0.0033	0.0016	0.0013	0.0014	0.0016	0.0024	0.0003	0.0016								
Sl	0.0034	0.0017	0.0015	0.0017	0.0019	0.0026	0.0005	0.0014	0.0001							
Hu	0.0030	0.0015	0.0013	0.0016	0.0020	0.0026	0.0004	0.0011	0.0001	0.0001						
Po	0.0053	0.0032	0.0028	0.0027	0.0023	0.0034	0.0012	0.0028	0.0004	0.0004	0.0006					
Ru	0.0059	0.0037	0.0034	0.0032	0.0025	0.0036	0.0016	0.0030	0.0008	0.0007	0.0009	0.0003				
CEU	0.0026	0.0008	0.0005	0.0002	0.0011	0.0012	0.0006	0.0028	0.0014	0.0016	0.0016	0.0026	0.0031			
CHB	0.1096	0.1094	0.1093	0.1096	0.1073	0.1081	0.1085	0.1047	0.1080	0.1069	0.1058	0.1086	0.1036	0.1095		
JPT	0.1118	0.1116	0.1114	0.1117	0.1095	0.1103	0.1107	0.1068	0.1102	0.1091	0.1079	0.1108	0.1057	0.1117	0.0069	
YRI	0.1460	0.1493	0.1496	0.1513	0.1524	0.1531	0.1502	0.1463	0.1503	0.1498	0.1490	0.1520	0.1504	0.1510	0.1901	0.1918

Simon et al. 2008



F_{ST}=0.001 e.g. SW-NO

F_{ST}=0.002 e.g. SW-HU

F_{ST}=0.003 e.g. SP-HU

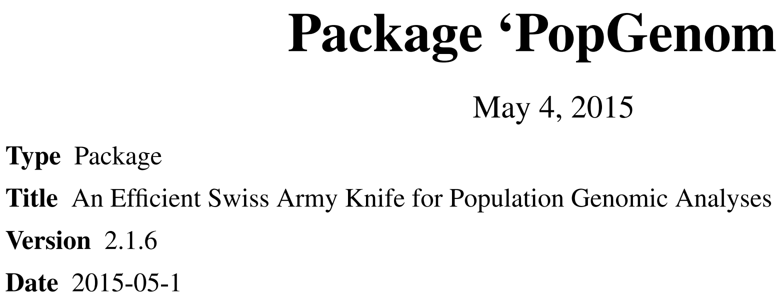
F_{ST}=0.004 e.g. SW-RO

F_{ST}=0.005 e.g. SP-PO

F_{ST}=0.006 e.g. SP-RU

To understand F_{ST}, here are simulated data using Balding method and the examples of EU populations as reported in (Simon et al. 2008)

F_{ST} – R Packages



Package ‘PopGenome’

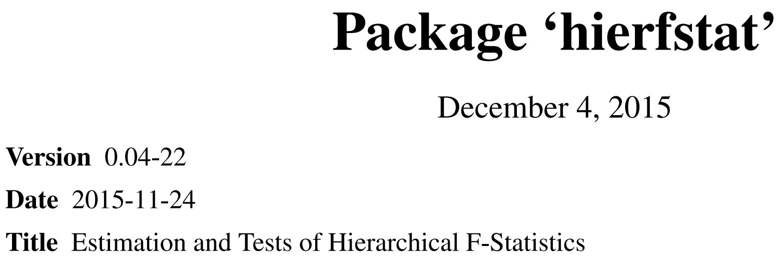
May 4, 2015

Type Package

Title An Efficient Swiss Army Knife for Population Genomic Analyses

Version 2.1.6

Date 2015-05-1



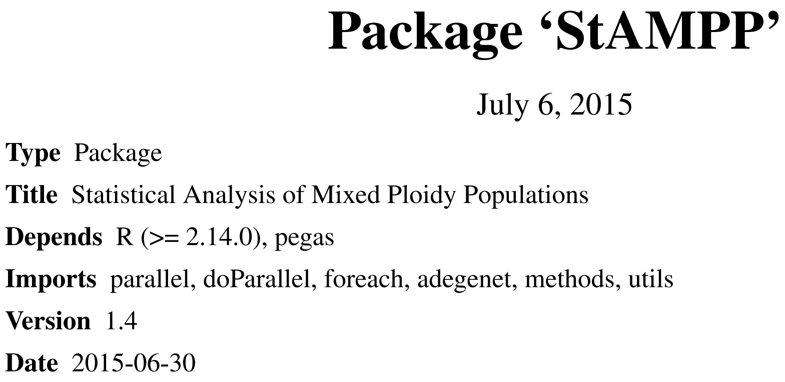
Package ‘hierfstat’

December 4, 2015

Version 0.04-22

Date 2015-11-24

Title Estimation and Tests of Hierarchical F-Statistics



Package ‘StAMPP’

July 6, 2015

Type Package

Title Statistical Analysis of Mixed Ploidy Populations

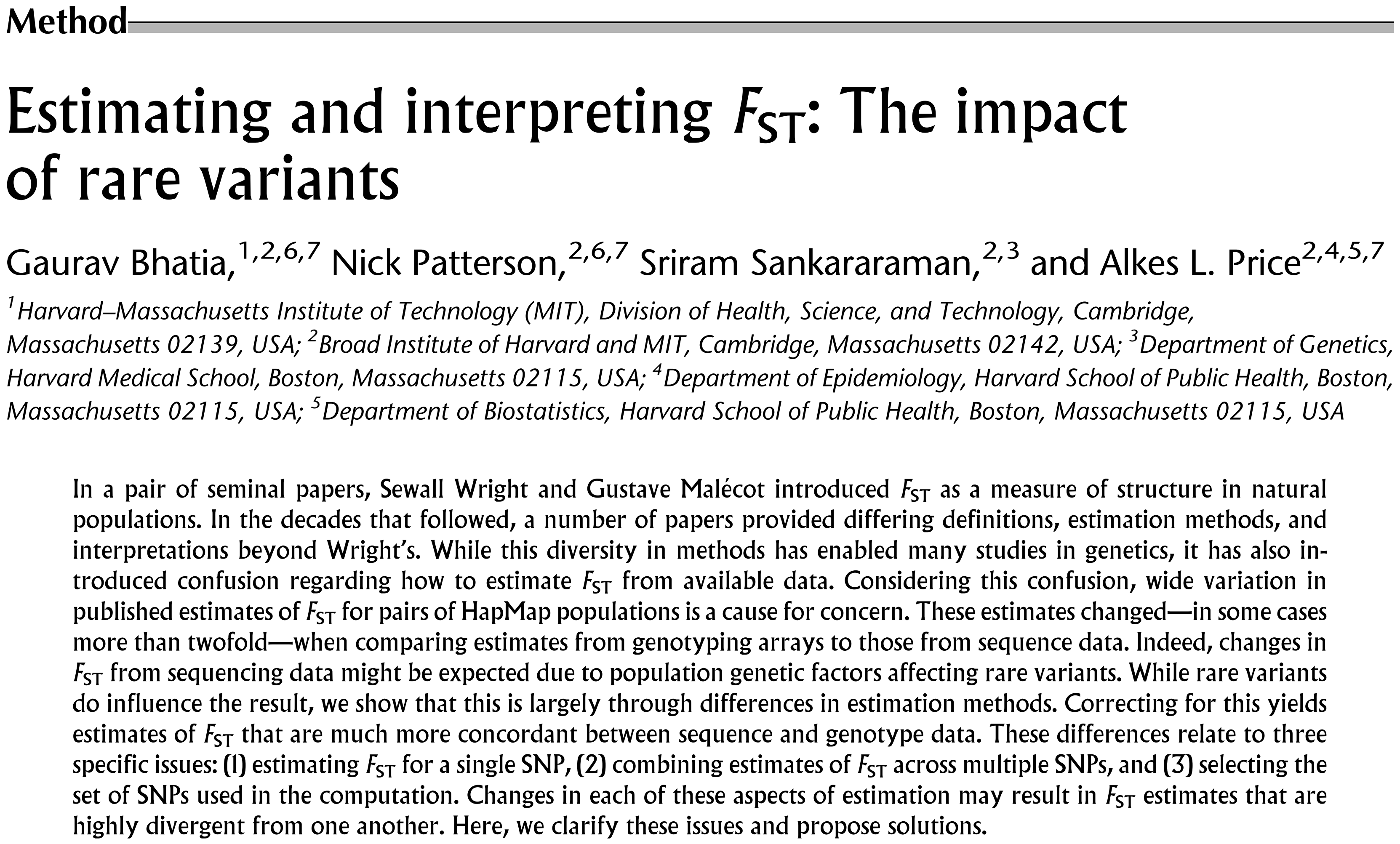
Depends R (>= 2.14.0), pegas

Imports parallel, doParallel, foreach, adegenet, methods, utils

Version 1.4

Date 2015-06-30

Estimating F_{ST}



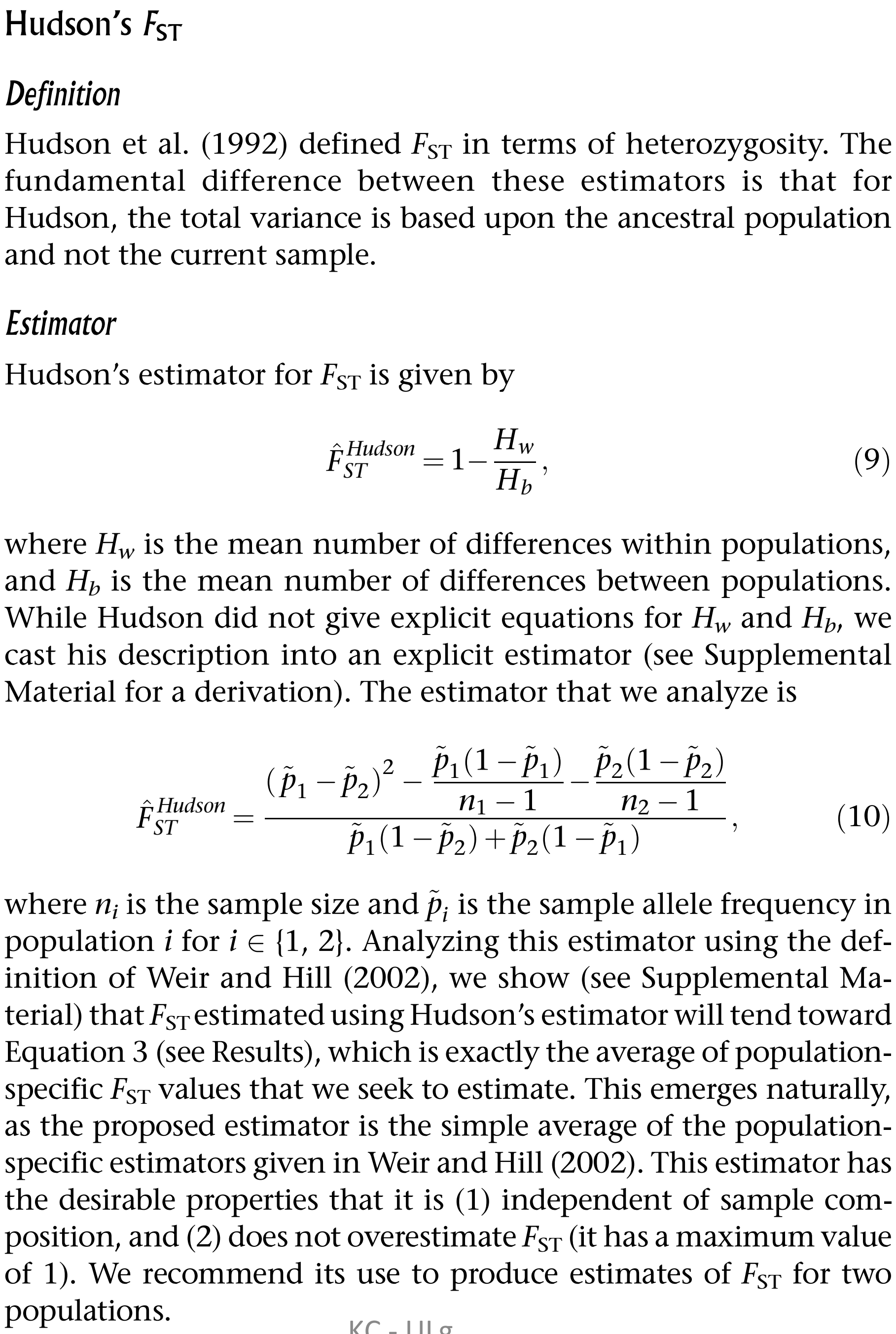
Method

Estimating and interpreting F_{ST}: The impact of rare variants

Gaurav Bhatia,[1,2,6,7] Nick Patterson,[2,6,7] Sriram Sankararaman,[2,3] and Alkes L. Price[2,4,5,7]

[1] Harvard–Massachusetts Institute of Technology (MIT), Division of Health, Science, and Technology, Cambridge, Massachusetts 02139, USA; [2] Broad Institute of Harvard and MIT, Cambridge, Massachusetts 02142, USA; [3] Department of Genetics, Harvard Medical School, Boston, Massachusetts 02115, USA; [4] Department of Epidemiology, Harvard School of Public Health, Boston, Massachusetts 02115, USA; [5] Department of Biostatistics, Harvard School of Public Health, Boston, Massachusetts 02115, USA

In a pair of seminal papers, Sewall Wright and Gustave Malécot introduced F_{ST} as a measure of structure in natural populations. In the decades that followed, a number of papers provided differing definitions, estimation methods, and interpretations beyond Wright's. While this diversity in methods has enabled many studies in genetics, it has also introduced confusion regarding how to estimate F_{ST} from available data. Considering this confusion, wide variation in published estimates of F_{ST} for pairs of HapMap populations is a cause for concern. These estimates changed—in some cases more than twofold—when comparing estimates from genotyping arrays to those from sequence data. Indeed, changes in F_{ST} from sequencing data might be expected due to population genetic factors affecting rare variants. While rare variants do influence the result, we show that this is largely through differences in estimation methods. Correcting for this yields estimates of F_{ST} that are much more concordant between sequence and genotype data. These differences relate to three specific issues: (1) estimating F_{ST} for a single SNP, (2) combining estimates of F_{ST} across multiple SNPs, and (3) selecting the set of SNPs used in the computation. Changes in each of these aspects of estimation may result in F_{ST} estimates that are highly divergent from one another. Here, we clarify these issues and propose solutions.



Hudson's F_{ST}

Definition

Hudson et al. (1992) defined F_{ST} in terms of heterozygosity. The fundamental difference between these estimators is that for Hudson, the total variance is based upon the ancestral population and not the current sample.

Estimator

Hudson's estimator for F_{ST} is given by

$$\hat{F}_{ST}^{Hudson} = 1 - \frac{H_w}{H_b}, \qquad (9)$$

where H_w is the mean number of differences within populations, and H_b is the mean number of differences between populations. While Hudson did not give explicit equations for H_w and H_b, we cast his description into an explicit estimator (see Supplemental Material for a derivation). The estimator that we analyze is

$$\hat{F}_{ST}^{Hudson} = \frac{(\tilde{p}_1 - \tilde{p}_2)^2 - \frac{\tilde{p}_1(1-\tilde{p}_1)}{n_1 - 1} - \frac{\tilde{p}_2(1-\tilde{p}_2)}{n_2 - 1}}{\tilde{p}_1(1-\tilde{p}_2) + \tilde{p}_2(1-\tilde{p}_1)}, \qquad (10)$$

where n_i is the sample size and $\tilde{p}_i$ is the sample allele frequency in population i for $i \in \{1, 2\}$. Analyzing this estimator using the definition of Weir and Hill (2002), we show (see Supplemental Material) that F_{ST} estimated using Hudson's estimator will tend toward Equation 3 (see Results), which is exactly the average of population-specific F_{ST} values that we seek to estimate. This emerges naturally, as the proposed estimator is the simple average of the population-specific estimators given in Weir and Hill (2002). This estimator has the desirable properties that it is (1) independent of sample composition, and (2) does not overestimate F_{ST} (it has a maximum value of 1). We recommend its use to produce estimates of F_{ST} for two populations.

Exercise – F_{ST} estimation

- Implement Hudson's method
- Estimate the average pairwise F_{ST} values for Pop1-6.