# PLINK and R

Archana Bhardwaj

# Motivation

- What is PLINK?
  - A software to analyse phenotype/genotype data
  - It is run from the command line
- Why should we use PLINK?
  - Perhaps the most common tool to analyse genome-wide genotyping data
  - It is free and open source
  - Designed to perform a wide range of basic, large-scale analyses in computationally efficient manner
  - Can  be used on several platforms

# Do I need to be afraid of PLINK?

# NO!

- It is not necessary to know how to program to use PLINK
- This presentation will provide you with available documentation for PLINK
- PLINK commands have a clear and intuitive structure

# Introduction

PLINK primarily aimed at genotype data

- SNPs
- "short" indels
- Some support for CNV

A leading tool for GWAS, structure analysis – many other tools support format.

- Not appropriate for many SVs, or when great variability

# Introduction

**Standard tool for manipulating genotype data**

- Vcftools
- PLINK/ PSEQ

**PLINK works with multiple file format**

- http://zzz.bwh.harvard.edu/plink/res.shtml

# How to get PLINK?

- Obtaning PLINK

http://zzz.bwh.harvard.edu/plink/res.shtml

- For Windows, choose MS-DOS

## Download

**PLINK** is now available for free download. Below are links to ZIP files containing binaries compilied on various platforms as well as the C/C++ source code. Linux/Unix users should download the source code and compile (see notes below).

These downloads also contain a version of gPLINK, an (optional) GUI for PLINK. Please see these pages for instructions on use of gPLINK.

Remember This release is considered a *stable* release, although please remember that we cannot guarantee that it, just like most computer programs, does not contain bugs...

| Platform | File | Version |
|---|---|---|
| Linux (x86_64) | plink-1.07-x86_64.zip | v1.07 |
| Linux (i686) | plink-1.07-i686.zip | v1.07 |
| MS-DOS | plink-1.07-dos.zip | v1.07 |
| Apple Mac (Intel) | plink-1.07-mac-intel.zip | v1.07 |
| C/C++ source (.zip) | plink-1.07-src.zip | v1.07 |

One more thing... If you download PLINK please either join the very low-volume e-mail list (link from Introduction page) or drop an e-mail to plink AT chgr dot mgh dot harvard dot edu letting me know you've downloaded a copy.

For old versions of PLINK please visit the archive.

Debian users PLINK is available as a Debian package, see these notes. Note, the executable is named snplink in the Debian plink package.

# PLINK Versions

PLINK in transition to PLINK 2

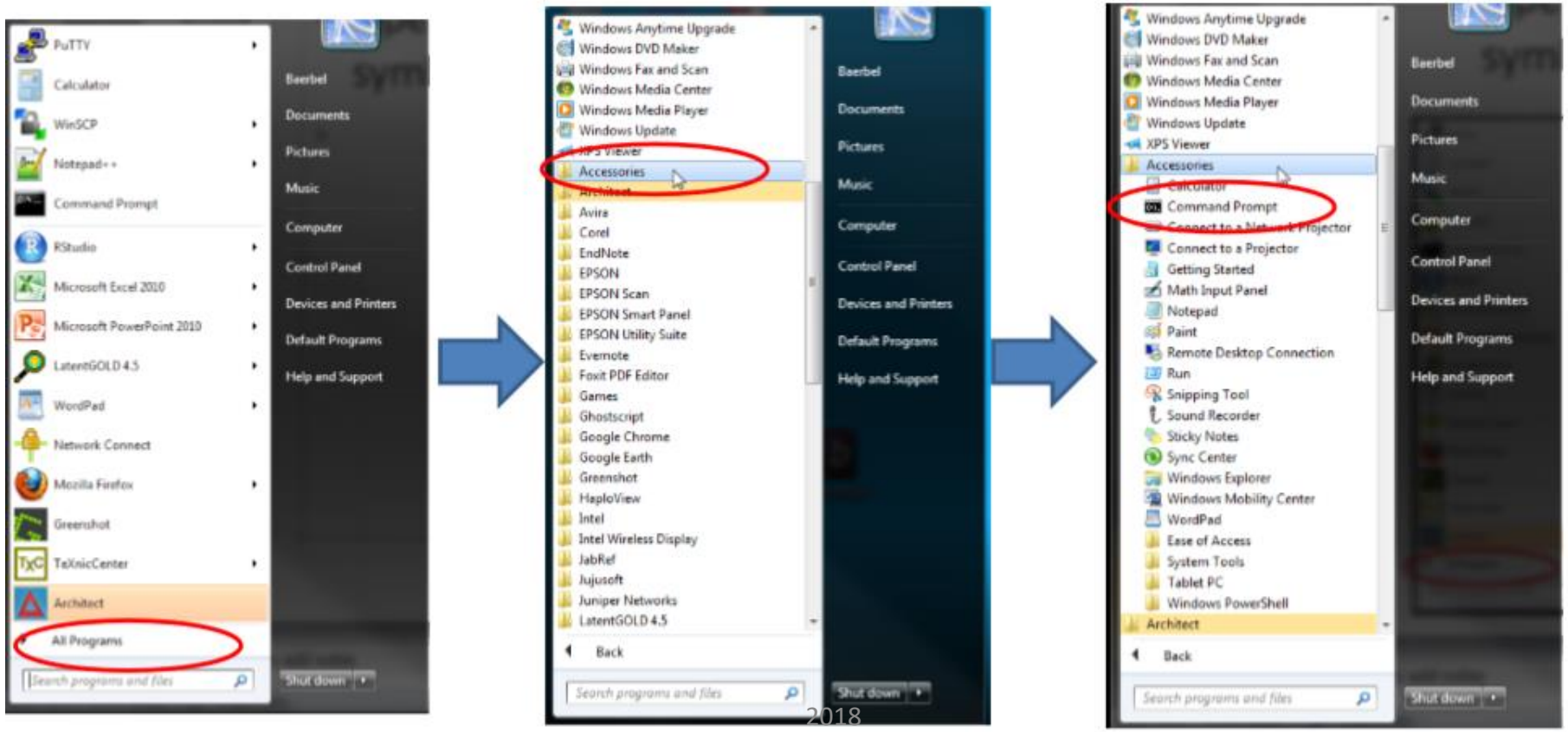
- Current version of Plink 1.90b2
- Previous version: 1.07

New version:

- Much faster
- Has more features
- Data compatible

# How to open command prompt

- Open start menu by clicking on window symbol in left corner 

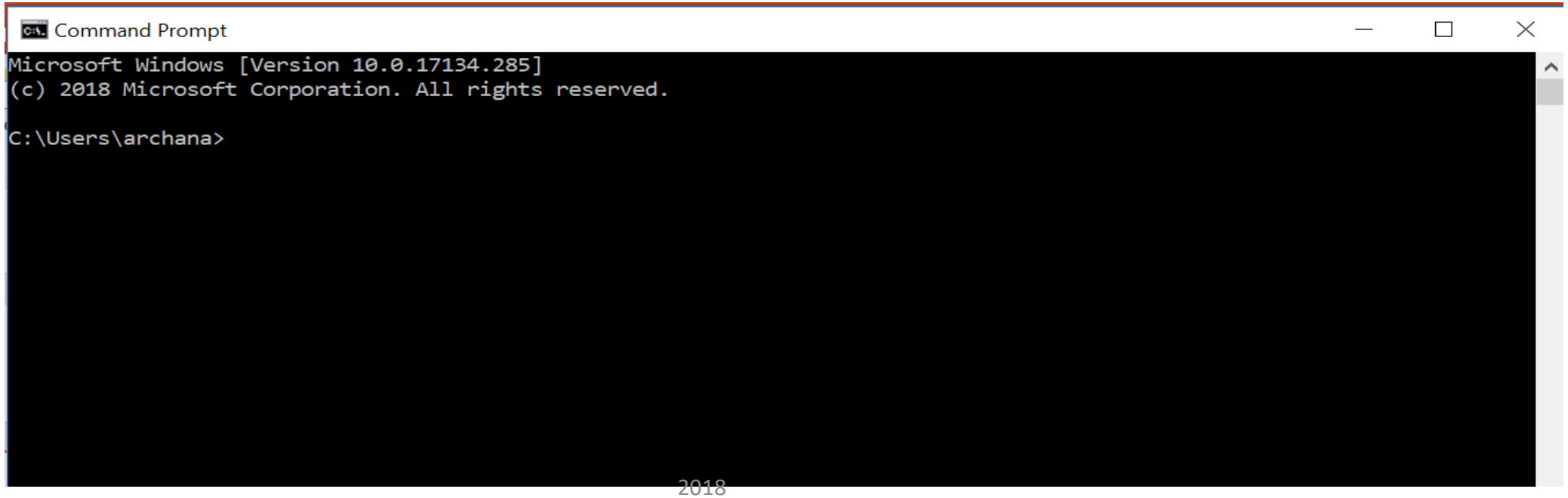


# How to get PLINK

Unzip zip into directory, eg :

C:\Users\archana\Desktop\plink_win64

You should be ready to go



# plink...

*Last original PLINK release is **v1.07** (10-Oct-2009); **PLINK 1.9** is now available for beta*

## Whole genome association analysis toolset





## Resources available for download

This page contains links to several freely-available resources, mostly generated by other individuals. All these resources are provided "as is", without any guarantees regarding their correctness or utility.

### The Phase 2 HapMap as a PLINK fileset

The HapMap genotype data (the latest is release 23) are available here as PLINK binary filesets. The SNPs are currently coded according NCBI build 36 coordinates on the forward strand. Several versions are available here: the entire dataset (a single, very large fileset: you will need a computer with at least 2Gb of RAM to load this file).

The *filtered* SNP set refers to a list of SNPs that have MAF greater than 0.01 and genotyping rate greater than 0.95 in the 60 CEU founders. This fileset is probably a good starting place for imputation in samples of European descent. Filtered versions of the other HapMap panels will be made available shortly.

| ***Description*** | ***File size*** | ***File name*** |
|---|---|---|
| Entire HapMap (release 23, 270 individuals, 3.96 million SNPs) | 120M | hapmap_r23a.zip |
| CEU (release 23, 90 individuals, 3.96 million SNPs) | 59M | hapmap_CEU_r23a.zip |
| YRI (release 23, 90 individuals, 3.88 million SNPs) | 65M | hapmap_YRI_r23a.zip |
| JPT+CHB (release 23, 90 individuals, 3.99 million SNPs) | 58M | hapmap_JPT_CHB_r23a.zip |
| CEU founders (release 23, 60 individuals, filtered 2.3 million SNPs) | 31M | hapmap_CEU_r23a_filtered.zip |
| YRI founders (release 23, 60 individuals, filtered 2.6 million SNPs) | 38M | hapmap_YRI_r23a_filtered.zip |
| JPT+CHB founders (release 23, 90 individuals, filtered 2.2 million SNPs) | 33M | hapmap_JPT_CHB_r23a_filtered.zip |

| ***Description*** | ***File size*** | ***File name*** |
|---|---|---|
| Entire HapMap (release 22, 270 individuals, 3.96 million SNPs) | 110M | hapmap_r22.zip |

# PLINK : PED Format

**The PED file is a white-space (space or tab) delimited file: the first six columns are mandatory:**

- Family ID
- Individual ID
- Paternal ID
- Maternal ID
- Sex (1=male; 2=female; other=unknown)
- Phenotype

**Test.ped**

```
1 1 0 0 1  1  A A  G T
2 1 0 0 1  1  A C  T G
3 1 0 0 1  1  C C  G G
4 1 0 0 1  2  A C  T T
5 1 0 0 1  2  C C  G T
6 1 0 0 1  2  C C  T T
```

**A PED file must have 1 and only 1 phenotype in the sixth column.**

# PLINK : MAP Format

**By default, each line of the MAP file describes a single marker and must contain exactly 4 columns:**

- chromosome (1-22, X, Y or 0 if unplaced)
- rs# or snp identifier
- Genetic distance (morgans)
- Base-pair position (bp units)

**Test.map**

**1 snp1 0 1**
**1 snp2 0 2**

**PED and MAP files can be specified separately, if they have different names**

# PLINK : Other FORMAT

- Binary format:
  -> BED, BIM, and FAM

- Transposed text format :
  ->TPED and TFAM

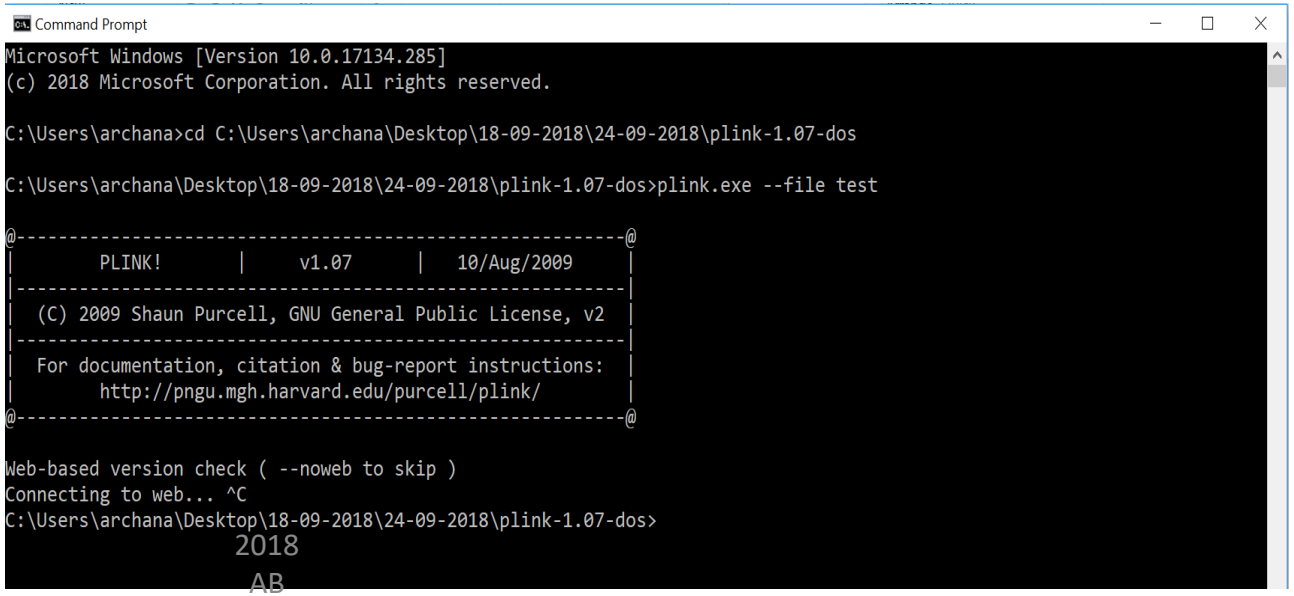
<- *trans.tped* ->

```
1 snp1 0 5000650 A A A C C C A C C C C C
1 snp2 0 5000830 G T G T G G T T G T T T
```

<- *trans.tfam* ->

```
1  1  0  0  1  1
2  1  0  0  1  1
3  1  0  0  1  1
4  1  0  0  1  2
5  1  0  0  1  2
6  1  0  0  1  2
```

- When PLINK starts it will attempt to contact the web, to check whether there is a more up-to-date version available or not.
- This option can be disabled with the --noweb option on the command line.



# Format Conversion

- To convert or to indicate output as text format (PED and MAP )

  Plink --file test --recode --out test_ped

- To convert or to indicate output as TPED and TFAM

  Plink --file test --transpose --recode --out test_tp

- To convert or to indicate output as Binary format TPED and TFAM

  Plink –file –make-bed –out test_bin

# PLINK: Command Line Run

- Type command :

  plink --file test

- For binary format (BED, BIM, and FAM)

  plink --bfile test

- For transposed text format(TPED, and TFAM).

Note that all files must have the same name, otherwise we need to clearly indicate by using --tped and –tfam

plink --tfile test

# Example data

- Download the example data from the course website
  - TSI_JPT_chr20_case_control.bed
  - TSI_JPT_chr20_case_control.bim
  - TSI_JPT_chr20_case_control.fam
  - TSI_JPT_chr20_pheno_header.txt
  - TSI_JPT_chr20_pheno.txt

# Data processing for SNPs(1/2)

- To get a set of SNPs, specify a single SNP and optionally, also ask for all SNPs in surrounding region, within the –window option

  plink --bfile mydata --snp rs652423 --window 20

- It will extract only SNPs within +/- 20kb of rs652423 based on multiple SNPs and ranges (--snps)
- To exclude some sets of SNPs

  Plink --bfile data –extract mysnps.txt

- Here, the file is mysnp.txt and –extract option will extract defined SNPs, one per line.

# Data processing for SNPs(2/2)

- The –snps command will accept a comma-delimited list of SNPs, including ranges based on physical position. For example,

plink --bfile mydata –snps rs273744, rs89883,rs12345- rs67890,rs999,rs222

- Based on physical position (--from-kb, etc)

plink --bfile mydata --chr 2 --from-kb 5000 --to-kb 10000

It will select all SNPs within this 5000kb region on chromosome 2.

# Quality control processes

- Missing genotype
- Hardy-Weinberg Equilibrium
- Minor Allele frequency
- Linkage disequilibrium pruning

# Missing Genotypes

- To generate a list genotyping/missingness rate statistics:

*plink --bfile data --missing This option creates two files:*

- plink.imiss
- plink.lmiss

- It provides the detail missingness by individual and by SNP (locus), respectively.

# Clustering based on Missing Genotypes

- Systematic batch effects that induce missingness in parts of the sample will induce correlation between the patterns of missing data that different individuals display.

- One approach to detect correlation in these patterns, that might possibly identify such biases, is to cluster individuals based on their identity-by-missingness (IBM).

  - plink --bfile data --cluster-missing

- which creates the files:

  - *plink.matrix.missing*
  - *plink.cluster3.missing*

which have similar formats to the corresponding IBS clustering files.

# Missing Rate Per Person

- The initial step in all data analysis is to exclude individuals with too much missing genotype data. This option is set as follows:

• plink --bfile mydata --mind 0.1

***which means exclude with more than 10% missing genotypes.***

- A line in the terminal output will appear, indicating how many individuals were removed due to low genotyping. If any individuals were removed, a file called **plink.irem** will be created, listing the Family and Individual IDs of these removed individuals.

# Missing Rate Per SNP

- Subsequent analyses can be set to automatically exclude SNPs on the basis of missing genotype rate, with the --geno option: the default is to include all SNPS (i.e. --geno 1).
- To include only SNPs with a 90% genotyping rate (10% missing) use

  - *plink --bfile mydata --geno 0.1*

- As with the --maf option, these counts are calculated after removing individuals with high missing genotype rates.

# Hardy-Weinberg Equilibrium (1/2)

- To generate a list of genotype counts and Hardy-Weinberg test statistics for each SNP, use the option:

• *plink --bfile data --hardy*

which creates a file: **plink.hwe.** The file has the following format

SNP SNP identifier
TEST Code indicating sample
A1 Minor allele code
A2 Major allele code
GENO Genotype counts:11/12/22
O(HET) observed hetrozygosity
E(HET) Expected hetrozygosity
P H-W p-value

# Hardy-Weinberg Equilibrium (2/2)

- To exclude markers that failure the Hardy-Weinberg test at a specified significance threshold, use the option:
  - plink --file mydata --hwe 0.001
- By default this filter uses an exact test. The standard asymptotic (1 df genotypic chi-squared test) can be requested with the --hwe2 option instead of --hwe.
- The following output will appear in the console window and in **plink.log,** detailing how many SNPs failed the Hardy-Weinberg test, for the sample as a whole, and (when PLINK has detected a disease phenotype) for cases and controls separately:

Writing Hardy-Weinberg tests (founders-only) to [ plink.hwe ]
30 markers failed HWE test ( p <= 0.05 ) and have been excluded
    34 markers failed HWE test in cases
    30 markers failed HWE test in controls

# Allele Frequency

- To generate a list of minor allele frequencies (MAF) for each SNP, based on all founders in the sample:
  - plink --file data --freq

- This will create a file: **plink.frq** with five columns:

| | |
|---|---|
| CHR | Chromosome |
| SNP | SNP identifier |
| A1 | Allele 1 code (minor allele) |
| A2 | Allele 2 code (major allele) |
| MAF | Minor allele frequency |
| NCHROBS | Non-missing allele count |

# Minor Allele Frequency

- Once individuals with too much missing genotype data have been excluded, subsequent analyses can be set to automatically exclude SNPs on the basis of MAF (minor allele frequency):
  - *plink --file mydata --maf 0.05*
- It means only include SNPs with MAF >= 0.05. The default value is 0.01. This quantity is based only on founders (i.e. individuals for whom the paternal and maternal individual codes and both 0).
- This option is appropriately counts alleles for X and Y chromosome SNPs.

# Linkage disequilibrium pruning (1/4)

- Sometimes it is useful to generate a pruned subset of SNPs that are in approximate linkage equilibrium with each other. This can be achieved via two commands:

--indep which prunes based on the variance inflation factor (VIF), which recursively removes SNPs within a sliding window;

-- indep-pairwise which is similar, except it is based only on pairwise genotypic correlation.

- The VIF pruning routine is performed:

plink --file data --indep 50 5 2

will create files **plink.prune.in** and **plink.prune.out**

# Linkage disequilibrium pruning (2/4)

- Each is a simlpe list of SNP IDs; both these files can subsequently be specified as the argument for a --extract or --exclude command.

- The parameters for --indep are: window size in SNPs (e.g. 50), the number of SNPs to shift the window at each step (e.g. 5), the VIF threshold. The VIF is 1/(1-R^2) where R^2 is the multiple correlation coefficient for a SNP being regressed on all other SNPs simultaneously.

- That is, this considers the correlations between SNPs but also between linear combinations of SNPs.

# Linkage disequilibrium pruning (3/4)

- The second procedure is performed:

• plink --file data --indep-pairwise 50 5 0.5

- This generates the same output files as the first option; the only difference is that a simple pairwise threshold is used.

- The first two parameters (50 and 5) are the same as above (window size and step); the third parameter represents the r^2 threshold.

# Linkage disequilibrium pruning (4/4)

- To give a concrete example: the command above that specify, 50 5 0.5 would

a) consider a window of 50 SNPs

b) calculate LD between each pair of SNPs in the window

c) remove one of a pair of SNPs if the LD is greater than 0.5

d) shift the window 5 SNPs forward and repeat the procedure.

# Association Analysis

- Case/control
- Fisher's exact Full model
- Multiple-testing correction

# Basic case/control association test

To perform a standard case/control association analysis, use the option:

plink --file mydata --assoc

which generates a file

plink.assoc

which contains the fields:

| | |
|---|---|
| CHR | Chromosome |
| SNP | SNP ID |
| BP | Physical position (base-pair) |
| A1 | Minor allele name (based on whole sample) |
| F_A | Frequency of this allele in cases |
| F_U | Frequency of this allele in controls |
| A2 | Major allele name |
| CHISQ | Basic allelic test chi-square (1df) |
| P | Asymptotic p-value for this test |
| OR | Estimated odds ratio (for A1, i.e. A2 is reference) |

# Fisher's Exact test (allelic association)

To perform a standard case/control association analysis using Fisher's exact test to generate significance, use the option:

plink --file mydata --fisher

which generates a file

plink.fisher

which contains the fields:

| | |
|---|---|
| CHR | Chromosome |
| SNP | SNP ID |
| BP | Physical position (base-pair) |
| A1 | Minor allele name (based on whole sample) |
| F_A | Frequency of this allele in cases |
| F_U | Frequency of this allele in controls |
| A2 | Major allele name |
| P | Exact p-value for this test |
| OR | Estimated odds ratio (for A1) |

As described below, if --fisher is specified with --model as well, PLINK will perform genotypic tests using Fisher's exact test.

# Adjustment for multiple testing

To generate a file of adjusted significance values that correct for all tests performed and other metrics, use the option:

plink --file mydata --assoc --adjust

which generates the file

plink.adjust

which contains the fields

| | |
|---|---|
| CHR | Chromosome number |
| SNP | SNP identifer |
| UNADJ | Unadjusted p-value |
| GC | Genomic-control corrected p-values |
| BONF | Bonferroni single-step adjusted p-values |
| HOLM | Holm (1979) step-down adjusted p-values |
| SIDAK_SS | Sidak single-step adjusted p-values |
| SIDAK_SD | Sidak step-down adjusted p-values |
| FDR_BH | Benjamini & Hochberg (1995) step-up FDR control |
| FDR_BY | Benjamini & Yekutieli (2001) step-up FDR control |

This file is sorted by significance value rather than genomic location, the most significant results being at the top.

# Working with PLINK

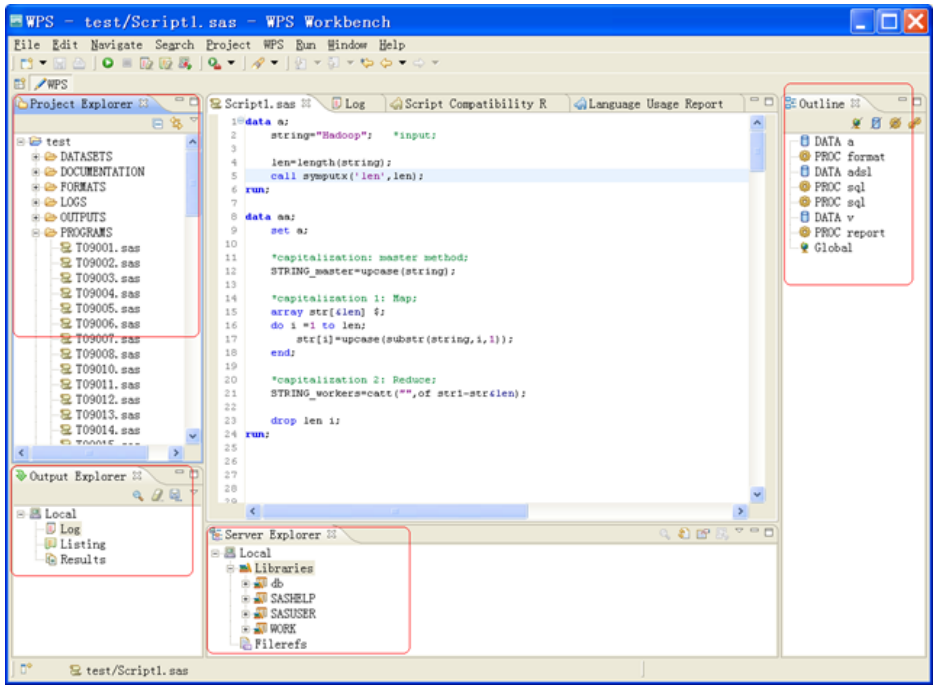
- Type all options on a single line
- Ensure exact syntax and spelling
- Always check the logfile!

# Introduction to 

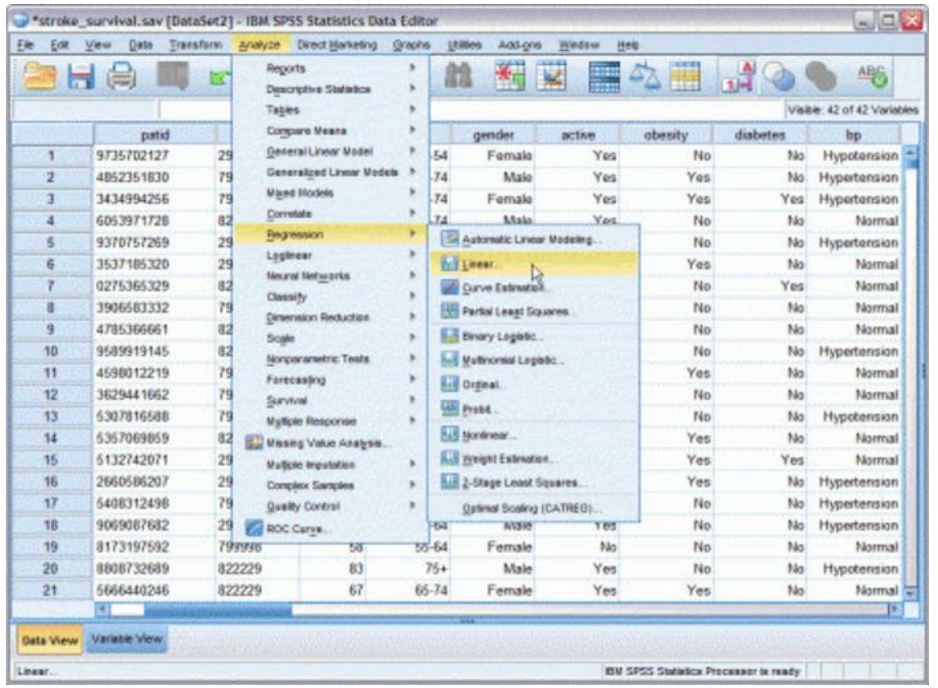
## A basic tutorial

2018
AB
GBIO0009

# Statistical languages GUIs

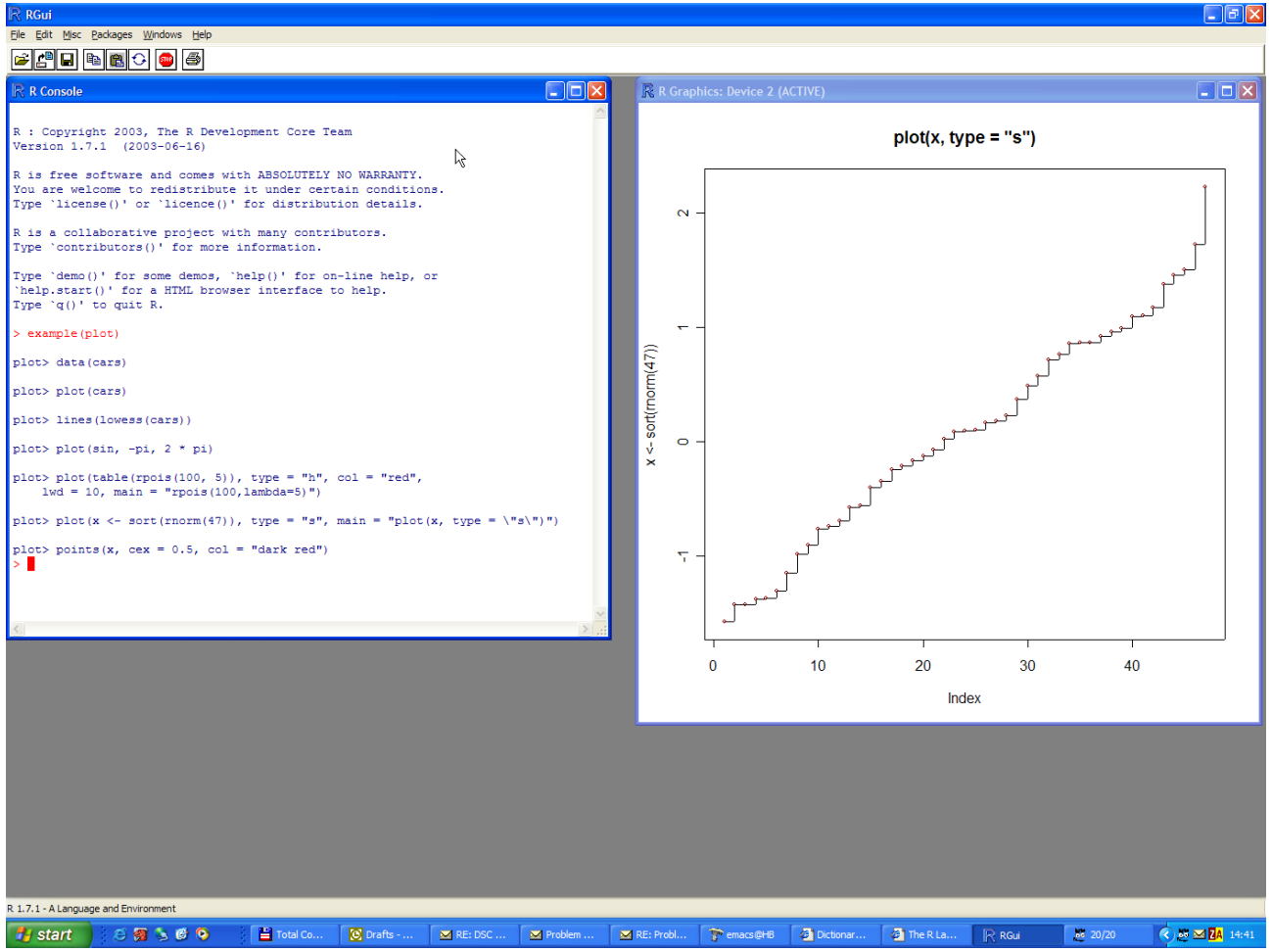



SAS




SPSS

# R GUI




Less fancy and no frills, but **free**!

# Definition

- “R is a free software environment for statistical computing and graphics”[1]
- R is considered to be one of the most widely used languages amongst statisticians, data miners, bioinformaticians and others.
- R is free implementation of S language
- Other commercial statistical packages are SPSS, SAS, MatLab



[1] R Core Team, R: A Language and Environment for Statistical Computing, Vienna, Austria (http://www.R-project.org/)

# Why to learn R?

- Since it is free and open-source, R is widely used by bioinformaticians and statisticians
- It is multiplatform and free
- Has wide very wide selection of additional libraries that allow it to use in many domains including bioinformatics
- Main library repositories CRAN and BioConductor

# Variables/Operators

- Variables store one element

```
x <- 25
```

Here x variable is assigned value 25

- Check value assigned to the variable x

```
>x
[1] 25
```

- Basic mathematical operators that could be applied to variables: (+),(-),(/),(*)
- Use parenthesis to obtain desired sequence of mathematical operations

# Arithmetic operators

- What is the value of small *z* here?

```
>x <- 25
> y <- 15
> z <- (x + y)*2
> Z <- z*z
> z
[1] 80
```

# Vectors

- Vectors have only 1 dimension and represent enumerated sequence of data. They can also store variables

```
> v1 <- c(1, 2, 3, 4, 5)
> mean(v1)
[1] 3
```

The elements of a vector are specified /modified with braces (e.g. [number])

```
> v1[1] <- 48
> v1
[1] 48 2 3 4 5
```

# Logical operators

- These operators mostly work on vectors, matrices and other data types
- Type of data is not important, the same operators are used for numeric and character data types

| Operator | Description |
|---|---|
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |
| == | exactly equal to |
| != | not equal to |
| !x | Not x |
| x \| y | x OR y |
| x & y | x AND y |

# Logical operators

- Can be applied to vectors in the following way. The return value is either True or False

```
> v1
[1] 48 2 3 4 5
> v1 <= 3
[1] FALSE TRUE TRUE FALSE FALSE
```

# R workspace

- Display all workplace objects (variables, vectors, etc.) via ls():

```
>ls()
[1] "Z" "v1" "x" "y" "z"
```

- Useful tip: to save “workplace” and restore from a file use:

```
>save.image(file = " workplace.rda")
>load(file = "workplace.rda")
```

# How to find help info?

- Any function in R has help information
- To invoke help use **?** Sign or help():

```
? function_name()

? mean

help(mean, try.all.packages=T)
```

- To search in all packages installed in your R installation always use  try.all.packages=T in help()
- To search for a key word in R documentation use help.search():

```
help.search("mean")
```

# Basic data types

- Data could be of 3 basic data types:
  - numeric
  - character
  - logical
- **Numeric** variable type:

```
> x <- 1
> mode(x)
[1] "numeric"
```

# Basic data types

- **Logical** variable type (True/False):

```
> y <- 3<4
> mode(y)
[1] "logical"
```

- **Character** variable type:

```
> z <- "Hello class"
> mode(z)
[1] "character"
```

# Data structures

- The main data objects in R are:
  - Matrices (single data type)
  - Data frames (supports various data types)
  - Lists (contain set of vectors)
  - Other more complex objects
- Matrices are 2D objects (rows/columns)

```
> m <- matrix(0,2,3)
> m
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
```

# Lists

- Lists contain various vectors. Each vector in the list can be accessed by double braces [[number]]

```
> x <- c(1, 2, 3, 4)
> y <- c(2, 3, 4)
> L1 <- list(x, y)
> L1
[[1]]
[1] 1 2 3 4
[[2]]
[1] 2 3 4
```

# Data Frames

- Data frames are similar to matrices but can contain various data types

```
> x <- c(1,5,10)
> y <- c("A", "B", "C")
> z <-data.frame(x,y)
   x y
1  1 A
2  5 B
3 10 C
```

# Input/Output

- To read data into R from a text file use read.table()
  - read help(read.table) to learn more
  - scan() is a more flexible alternative

```
raw_data <-read.table(file="data_file.txt")
```

- To write data into R from a text file use read.table()

```
> write.table(mydata, "data_file.txt")
```

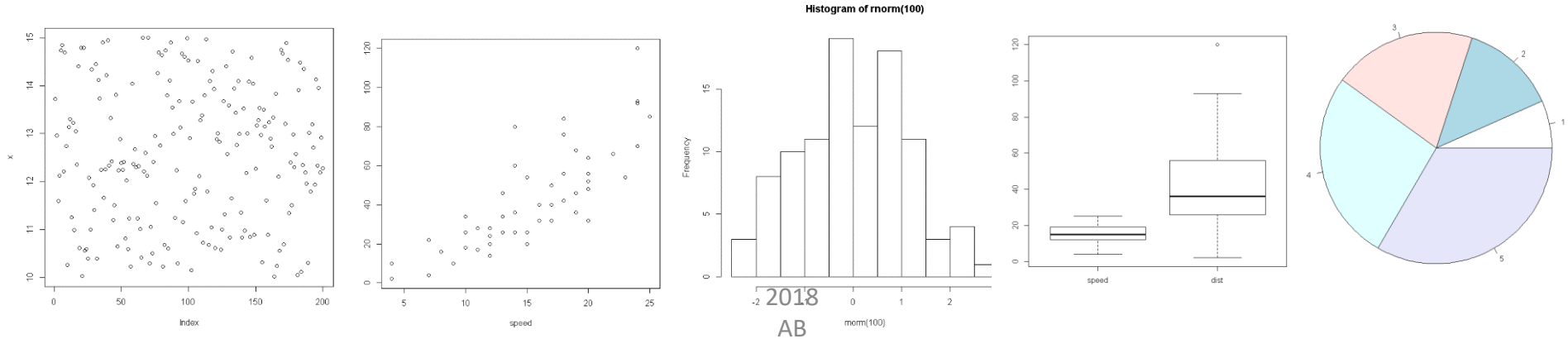
# Plots generation in R

- R provides very rich set of plotting possibilities
- The basic command is **plot()**
- Each library has its own version of plot() function
- When R plots graphics it opens “graphical device” that could be either a window or a file

# Plotting functions

- R offers following array of plotting functions

| Function | Description |
|---|---|
| **plot(x)** | plot of the values of x variable on the y axis |
| **plot(x,y)** | bi-variable plot of x and y values (both axis scaled based on values of x and y variables) |
| **pie(y)** | circular pie-char |
| **boxplot(x)** | Plots a box plot showing variables via their quantiles |
| **hist(x)** | Plots a histogram(bar plot) |



# Plot modification functions

- Often R plots are not optimal and one would like to add colors or to correct position of the legend or do other appropriate modifications
- R has an array of **graphical parameters** that are a bit complex to learn at first glance. Here is the full list
- Some of the graphical parameters can be specified inside **plot()** or using other graphical functions such as **lines()**

# Plot modification functions

| Function | Description |
|---|---|
| **points(x,y)** | add points to the plot using coordinates specified in x and y vectors |
| **lines(x,y)** | adds a line using coordinates in x and y |
| **mtext(text,side=3)** | adds text to a given margin specified by side number |
| **boxplot(x)** | this a histogram that bins values of x into categories represented as bars |
| **arrows(x0,y0,x1,y1, angle=30, code=1)** | adds arrow to the plot specified by the x0, y0, x1, y1 coordinates. Angle provides rotational angle and code specifies at which end arrow should be drawn |
| **abline(h=y)** | draws horizontal line at y coordinate |
| **rect(x1, y1, x2, y2)** | draws rectangle at x1, y1, x2, y2 coordinates |
| **legend(x,y)** | plots legend of the plot at the position specified by x and y vectors used to generate a given plot |
| **title()** | adds title to the plot |
| **axis(side, vect)** | adds axis depending on the chosen one of the 4 sides; vector specifying where tick marks are drawn |

# Installation of new libraries

- There are two main R repositories
  - CRAN
  - BioConductor
- To install package/library from CRAN

```
install.packages("seqinr")
```

To install packages from BioConductor

```
source("http://bioconductor.org/biocLite.R")
biocLite("GenomicRanges")
```

# Installation of new libraries

- Download and install latest R version on your PC. Go to http://cran.r-project.org/
- Install following libraries by running

```
install.packages(c("seqinr", "ape", "GenABEL"))
```

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("limma","affy","hgu133plus2.db","Biostrings", "muscle"))
```