

Basic programming for R

GBIO0009

Kridsakorn Chaichoompu

University of Liege

Important Links

- R Project

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

- Bioconductor

<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
- $\log_{10}(8)$ Logarithm in base 10
- π 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', "l", "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="-"))`

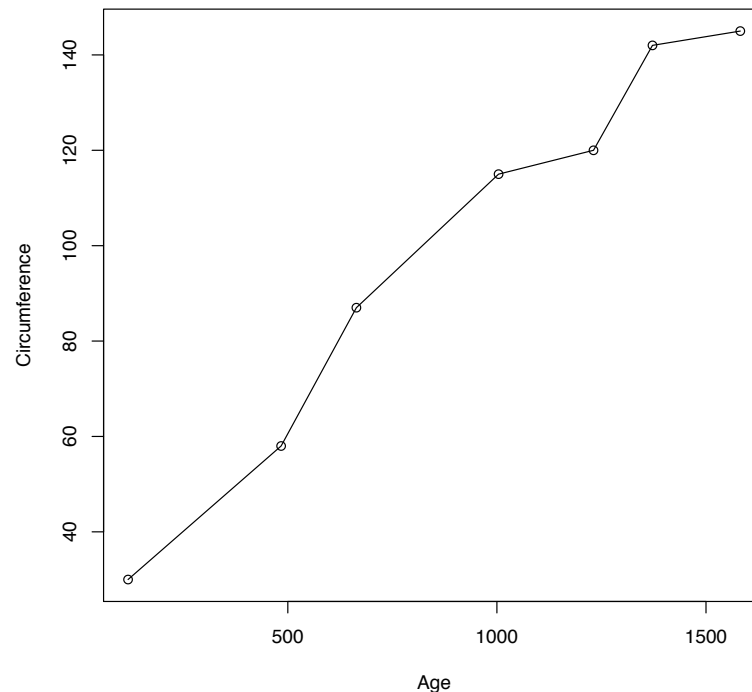
Plots

- 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:

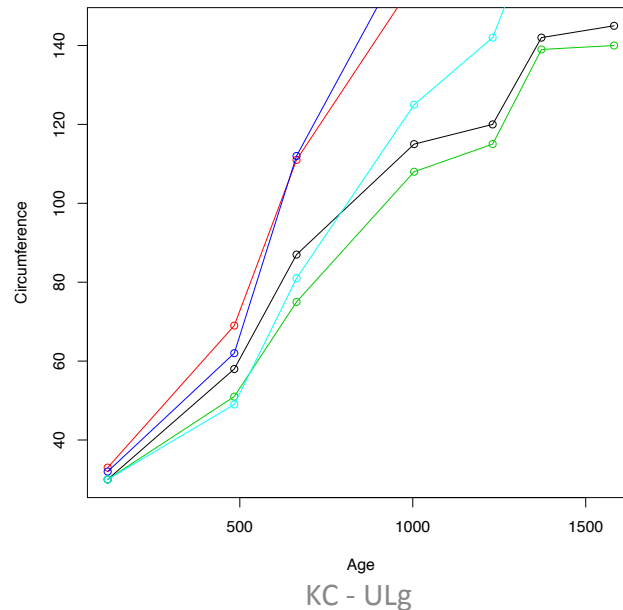
```
age=mydata$age[which(mydata$Tree==1)]  
cir=mydata$circumference[which(mydata$Tree==1)]  
plot(age,cir,type="o",xlab="Age",ylab="Circumference",  
col=1)
```



Plotting for multiple data series (2)

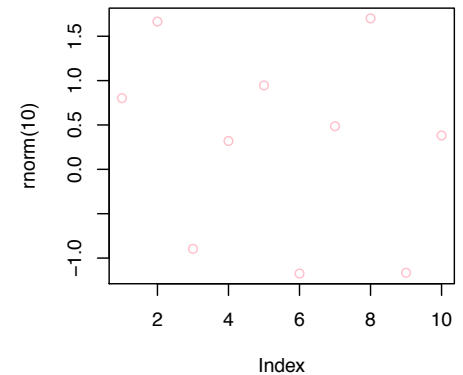
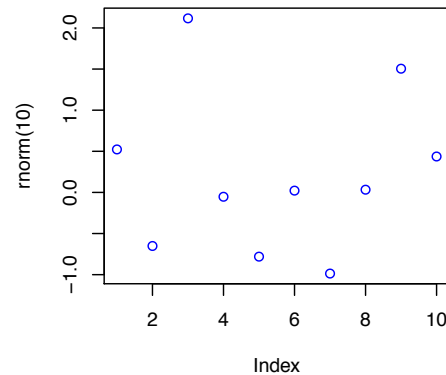
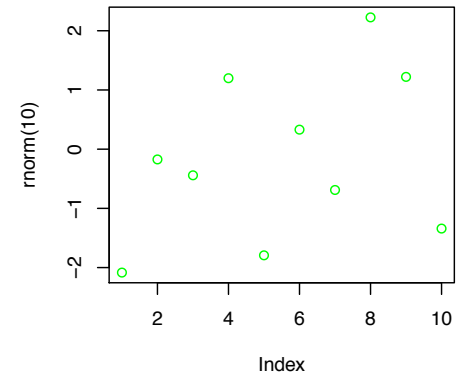
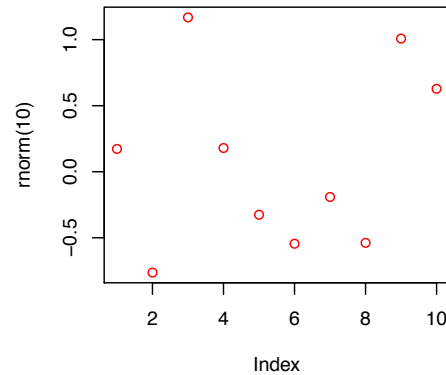
Add more lines:

```
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)  
}
```

R for Bioinformatics and Biostatistics



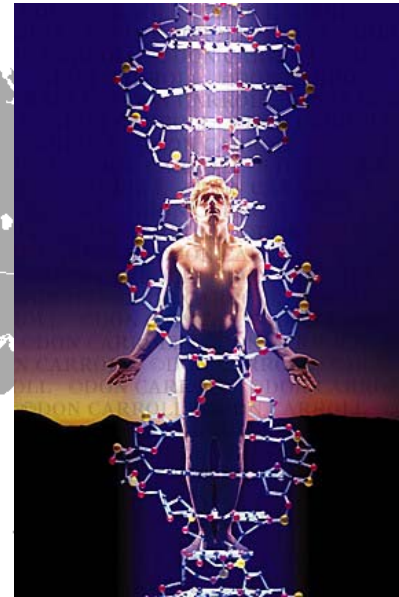
Diversity

- 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?

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)**.



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

$$\text{PCs} = X.B$$

Normalization

- Zero means

If X is a vector

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

- Unit variance

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

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

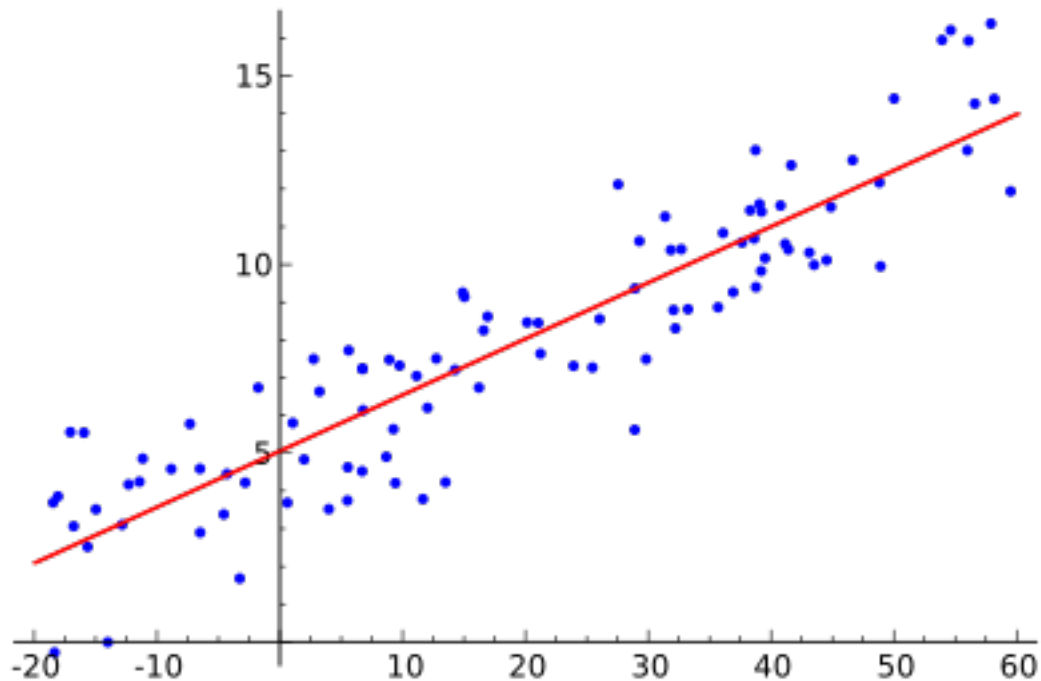
Exercise - PCA

- Calculate PCs for the example data - `simSNP_rep1`, more information
- Plot the first two eigenvectors
- Plot the first two PCs
- Plot the first three PCs using the package `plot3D`

Regression Models

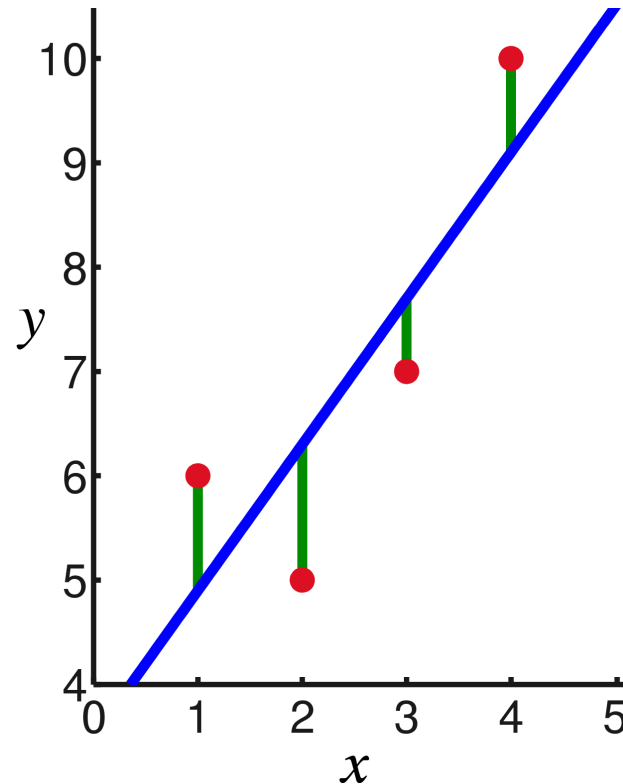
- Linear regression
- Logistic regression

Linear Regression



`lm(formula, data)`

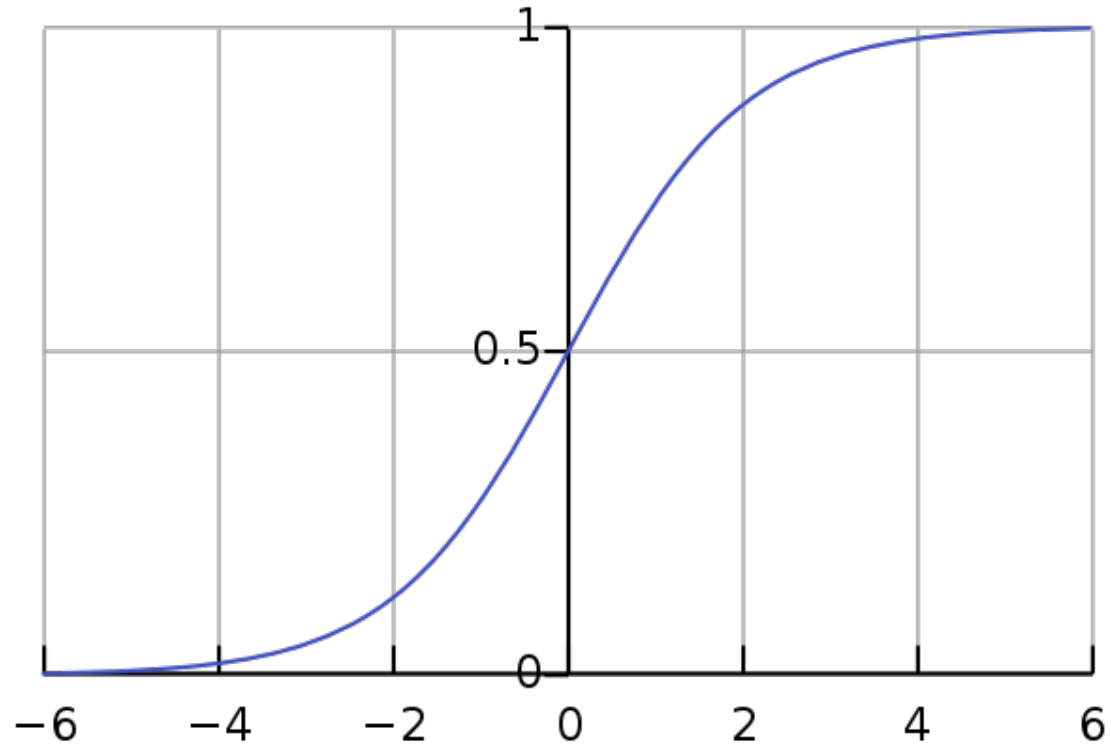
Residuals



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

```
lm(formula, data)$residuals
```

Non-linear model: Logistic Regression



Generalized Linear Models

`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")

Exercise – Regression models

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

$$\text{SNP} \sim \text{PC1} + \text{PC2} + \text{PC3}$$

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