Intro to

for Population Genetics 2024

Why?

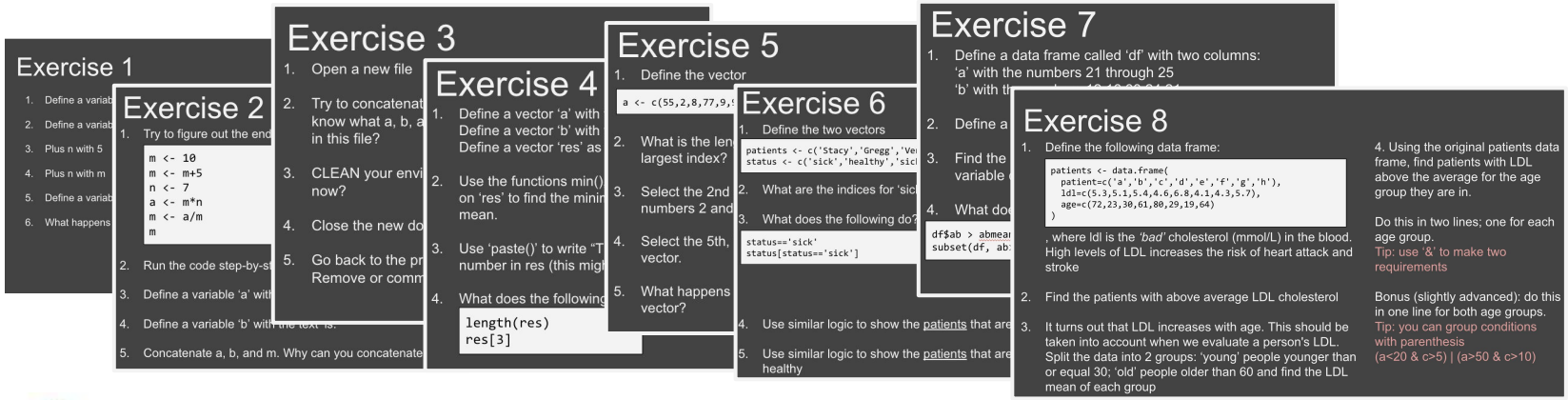
Why **NOT** to use Excel:

Why use **R**

- Correct and reproducible science!
- All analysis you will need in your career
- Beautiful figures
- Highly wanted skill in industry and academia

How?

Hands-on exercises is the only way to learn



stackoverflow & Google

are your friends

Recently, ChatGPT can also help a lot!

FR

What is wrong with this R code?:
a <- 1,2,4,8,3

This R code has a syntax error. The assignment operator " <- " should separate the object name (a) and the value, but in this case, the values are separated by commas, not spaces. Additionally, the values are not grouped into a vector. The correct syntax for assigning a vector to an object in R is as follows:

a <- c(1, 2, 4, 8, 3)

We encourage you to use it to learn, study, and explore.

But unfortunately, **<u>you are not allowed to use it for ANY handins/exam</u>**

Jyllands-Posten KØB MENU

Universiteter forbyder brugen af den omstridte chatrobot ChatGPT

Danske universiteter frygter, at studerende kan bruge kunstig intelligens til at snyde til eksamen.

Learning objective

Today you will get an intro to

1. Variables
2. Vectors
3. DataFrames
4. Reading data
5. Plotting data
6. Using Libraries

This will not cover everything, but you will use R throughout the course and get more used to it.

Don't be scared - it looks more difficult than it is

1. Variables

Defining variables is just giving stuff names for later use

Code:

```
x <- 2
y <- 204
b <- 'chimp'
```

Explanation:

'x' now contains the number 2
'y' now contains the number 204
'b' now contains the string 'chimp'

You can put anything into a variable
You can use the variable for calculations

You can't start the name with a number
You can't have spaces in the name

```
1a <- 102
my variable <- 'hello'
```

1. Variables

You can add, subtract, etc. with variables

example:

```
> n <- 2
> m <- 204
> n+m
[1] 206
> m-n
[1] 202
```

`n+m`	sum
`n-m`	subtract
`n*m`	times
`n/m`	divide
`n^m`	power
`sqrt(n)`	square root
`log(n)`	Natural logarithm
`log10(n)`	Base-10 logarithm

Exercise 1

1. Define a variable 'a' with the value 135
2. Define a variable 'b' with the value 73
3. Calculate a minus b
4. Calculate the product of a and b
5. Define a new variable 'absum' as the sum of a and b
6. Calculate the square root of absum
7. Explicitly tell R to print out absum with: `print(absum)`

Tip: login to the server with the command on absalon

Run `R`

If you want to close R, run `q()`

And say ‘n’ to saving workspace

2. Vectors

Programming starts to be very powerful when we do thousands or even millions of calculations at the same time. We do this with vectors:

Code:

```
a <- c(1,9,4,6,5)



b <- 1:10



f <- c('healthy', 'sick')
```

Explanation:

'a' is now a variable containing the vector of numbers: 1, 9, 4, 6, 5 in that specific order

'b' is now a variable containing the vector of numbers 1,2,3,4,5,6,7,8,9,10 in that specific order

'f' is now a variable containing the vector of strings/characters 'healthy', 'sick' in that specific order

2. Vectors

Majority of R functions also works on vectors:

Code:

```
a <- c(1,9,4,6,5)
a+10

b <- c(2,3,4,2,3)
a-b



paste('result:', a)
```

Explanation:

plus every number in a with 10

minus 1st number in a with 1st number in b, 2nd number in a with 2nd number in b, etc..

write 'result: ' before each number in a

Exercise 2

1. Define a vector 'a' with the numbers 4,2,9,7,8
   Define a vector 'b' with the numbers 0.5,10,200,-1,1
   Define a vector 'res' as the product of a and b

2. Use the functions min(), max(), sum(), and mean() on 'res' to find the minimum, maximum, sum, and mean.

3. Use 'paste()' to write “The result is ” before each number in res (this might take a few tries).

4. What does the following lines do?

```
length(res)
res[3]
```

2. Vectors

Vectors are ordered. To access a certain element you use indices:

Index: *1* *2* *3* *4*

```
a <- c('Stacy','Gregg','Vera','Kate')

a[4]

a[2:4]

a[c(1,4)]

a[c(T,F,F,T)]
```

Explanation:

Define vector with names

Access the 4th name

Access the 2nd through 4th name

Access the 1st and 4th name

Access the 1st and 4th name based on T/F (True/False) vector

2. Vectors

T/F vectors are called logical/boolean vectors: F=False and T=True

'!' in front inverts the vector så F->T and T->F

T/F vectors are often made from comparisons.

Code:

```
a <- c(1,9,4,6,5)
a>4


a[a>4]
```

Explanation:

logical vector which will be true for values in 'a' that are larger than 4

Only get the numbers in 'a' that is larger than 4

code	comparison
`a == b`	equal
`a != b`	Not equal
`a > b`	Greater than
`a < b`	Less than
`a >= b`	Greater or equal
`a <= b`	Less or equal

Exercise 3

1. Define the vector

```
a <- c(55,2,18,9,6)
```

2. What is the length of the vector, i.e. what is the largest index?

3. Select the 2nd and 5th value using a vector with the numbers 2 and 5

4. Select the 2nd and 5th value using a T/F vector.

5. Make a T/F vector from the comparison: a<9

6. Take out the elements in 'a' that are <9

7. Take out elements that are >=9

Today you will get an intro to

1. Variables `a<-10`
2. Vectors `a<-c(1,7,4,7)`
3. DataFrames
4. Reading data
5. Plotting data
6. Using Libraries

Too many related vectors can quickly become messy.
Instead, we use dataframes

15 min break… then Data Frames

3. Data frames

Data frames is just a bunch of vectors with the same length, i.e. what we usually call a table. This is going to be your main workhorse.

Table

x	y
1	2
2	8
3	7
4	6
5	4
6	1

Code:

```
df <- data.frame(
    x=c(1,2,3,4,5,6),
    y=c(2,8,7,6,4,1)
)


df$x
df$y

df$xy <- df$x + df$y
```

Explanation:

We define a data frame called 'df' (you can call it whatever you like).

df has two vectors/columns x and y.

Take out the column x
Take out the column y

Make a new column as the sum of column x and column y

3. Data frames

You can access specific parts of the table (index) as with vectors

Table

	x	y	z
	4	6	'chimp'
[2,1] →	5	4	'dog'
	6	1	'chimp'

```
df <- data.frame(
    x=c(4,5,6),
    y=c(6,4,1),
    z=c('chimp','dog','chimp')
)

df[2,1]
```

3. Data frames

You can access specific parts of the table (index) as with vectors

Table

	x	y	z
	4	6	'chimp'
[2,1] →	5	4	'dog'
	6	1	'chimp'

```
df <- data.frame(
     x=c(4,5,6),
     y=c(6,4,1),
     z=c('chimp','dog','chimp')
)

df[2,1]

df[df$y<6, ]
```

And you can use comparisons as well

Get the rows where y-column is <6

Leaving this part blank (<u>but remember the comma</u>) takes all columns

Exercise 4

1. Define a data frame called 'df' with columns:
   'a' with the numbers 21 through 25
   'b' with the numbers 12,16,33,24,21

2. Print out the data frame. How many rows and columns?

3. What do you think `dim(df)` does?

4. Manually take out the cell at row 3, column 2

5. Define a new column 'ab' as the sum of the two

6. Find the mean of the new 'ab' column and save in a new <u>variable</u> called 'abmean'

Tip: mean(df$a) calculates the mean of column a in df.

7. Using a comparison, take out the rows where column ab is larger than 'abmean'

4. Reading Data

You never write your data manually into R (as we have done previously).
You always read the data from a file.

You will mainly work with text files that represent tables/data frames with rows and columns:

Table

x	y	z
1	2	a
2	8	b
3	7	c
4	6	d
5	4	e
6	1	f

`filename.csv`
(columns separated by commas)

Header line →

```
x,y,z
1,2,a
2,8,b
3,7,c
4,6,d
5,4,e
6,1,f
```

`filename.tsv`
(columns separated by tabs '\t')

Header line →

```
x	y	z
1	2	a
2	8	b
3	7	c
4	6	d
5	4	e
6	1	f
```

`filename.txt`
(columns separated by tabs, commas, or other)

Header line →

```
x	y	z
1	2	a
2	8	b
3	7	c
4	6	d
5	4	e
6	1	f
```

Sometimes, there's a header line with the names of columns. Sometimes not.

4. Reading Data

We read data with a function called: read.table()

```
read.table('/path/to/data.csv', sep=',', header=T)
```

Path (where the file is located)

What separates the columns

Whether the file has a header line or not

The path is the 'address' of the file.

The address depends your user name and where you downloaded the file!

```
/home/mqr375/rIntro/data.csv
```

If you download a file to your own computer you <u>can't</u> read it on the server

Exercise 5

1. Go out of R `q()` Make a directory called ‘introR’ and go into that directory

2. Download the following file with wget:

```
wget http://popgen.dk/ffs/teaching/2024-02-PopGen/data.csv
```

3. Go in R `R` Start by writing read.table('/') and with the cursor here: `> read.table('/')` double-press the tab (⇆) button. This will show you the possibilities. Write '/home/' and double-press tab again. With these help messages, write the full path to the downloaded data.

Tip: look at the previous slide

4. Set sep=',' and header=T and read the file

5. Save the data in a variable named 'data'. What are the columns? How many rows are there?

6. What happens if you set header=F and rerun?

Today you will get an intro to

1. Variables `a<-10`
2. Vectors `a<-c(1,7,4,7)`
3. DataFrames `df<-data.frame(a=1:5,b=c(3,8,5,9,6))`
4. Reading data `read.table('/home/mqr375/data.csv', sep='\t', header=T)`
5. Plotting data
6. Using Libraries

Now we know how to read and handle data.
But usually we also want to plot it!

15 min break… then Plotting and libraries

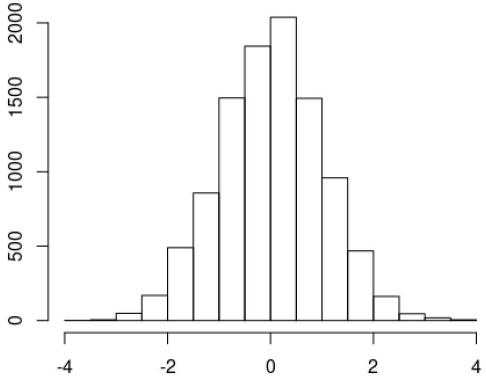
5. Plotting data

R can do all types of plots of your data

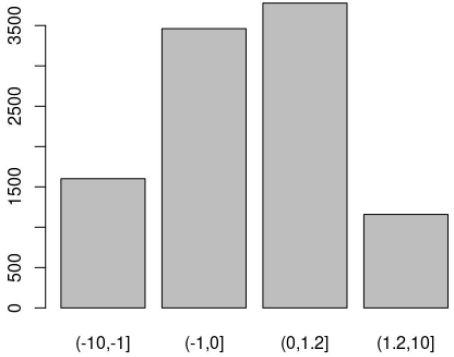
Making them “beautiful” takes a little effort

We’ll go through a few examples

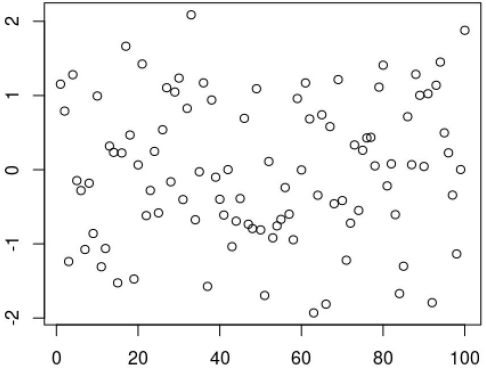
Histogram



Barplot



Scatterplot



Exercise 6

1. Go out of R and download the file `wget http://popgen.dk/ffs/teaching/2024-02-PopGen/heights.tsv`

2. Go into R and read the file with sep='\t' and header=T. Put into variable called ‘data’.

3. Use `head(data)  nrow(data)  table(data$sex)`

   Briefly describe the data: What’s in the columns? How many rows? How many individuals of the different sex?

4. Make a <u>histogram</u> of the height `hist(data$height)` What’s on the x- and y-axis?

5. `table(data$decadeofbirth)` Will give you number of individuals born in different decades. Make a <u>barplot</u> of these counts: `barplot( table(data$decadeofbirth) )` What decade has most indvs?

6. Make a new column in data called 'cmheight' (cm=2.54*inch) based on column 'height'. Make a new column in 'data' called 'age' based on the column 'yearofbirth'.

7. Make a scatterplot: `plot(data$age, data$cmheight, col=data$sex)` what’s on x and y? What do the colors represent?

6. Libraries

R has thousands of extra libraries that can help you with anything

We will often use the library snpMatrix to load genetic data

Instead of working with the alleles we convert it to numbers: 0,1,2

	Genotype	In R matrix
Homozygous for A	A / A	0
Heterozygous	A / B	1
Homozygous for B	B / B	2

Exercise 7

1. Go out of R and download the files:

```
wget http://popgen.dk/ffs/teaching/2024-02-PopGen/geno.bed
wget http://popgen.dk/ffs/teaching/2024-02-PopGen/geno.bim
wget http://popgen.dk/ffs/teaching/2024-02-PopGen/geno.fam
```

Open R and run:

```
library(snpMatrix)
data <- read.plink('geno')
geno <- matrix(as.integer(data@.Data),nrow=nrow(data@.Data))-1
geno[geno<0] <- NA
```

2. We now created a matrix ‘geno’. Rows are individuals, columns are SNPs - how many of each?

3. What is the genotype of individual 43 at snp 3395? Tip: look at indexing in slide 20

4. `SNPMiss <- colSums(is.na(geno))` calculates the number of missing genotypes for each SNP (column) and puts it in a vector called 'SNPMiss'. Make a histogram 'SNPMiss'. What’s the max missing for a SNP?

5. Similarly, there exist a rowSums function. Use this to calculate the number of missing genotypes for each individual and save in a vector called 'IndvMiss'. Make a histogram. What’s the max missing for an Indv?

6. Make a new matrix called 'geno_filt' with the individuals with <30 missing sites, using IndvMiss from above. How many individuals are left?

Tip: look at comparisons in slide 20

That’s it for today

Cheat-sheet from today

```
# Define a variable
n <- 5
m <- 10
a <- 'hello'

# Add two variables
absum <- a+b

# Define a vector
a <- c(1,2,3,4,5)
b <- 1:10

# Get 2nd element in vector
a[2]

# Define a data frame
df <- data.frame(
    a=c(1,5,8,3),
    b=c('chimp', 'dog', 'chimp', 'dog')
)

# Get row 4, column 2 in data frame
df[4,2]

# Get rows where column ‘a’ is larger than 4.
df[df$a>4, ]

# Read file into data frame
df <- read.table('path/to/data.csv', sep=',', header='T')
```

```
# Use snpMatrix to load genotype matrix
library(snpMatrix)
data <- read.plink('path/to/genofile')
geno <- matrix(as.integer(data@.Data),nrow=nrow(data@.Data))-1
geno[geno<0] <- NA

# Sum by column or by row
colNtot <- colSums(geno)
rowNtot <- rowSums(geno)

# Count number NA/missing by column or by row
colMiss <- colSums(is.na(geno))
rowMiss <- rowSums(is.na(geno))

# Only keep columns with no missingness
geno[,colMiss==0]

# Only keep rows with total count > 100
geno[rowNtot>100, ]

# Count occurrences of different values of column sex in df
table(df$sex)

# Barplot of number of occurences
barplot(table(df$sex))

# Scatterplot of x and y in df. Color by df$color
plot(df$x, df$y, col=df$color)

# Histogram of column 'height' in df
hist(df$height)
```