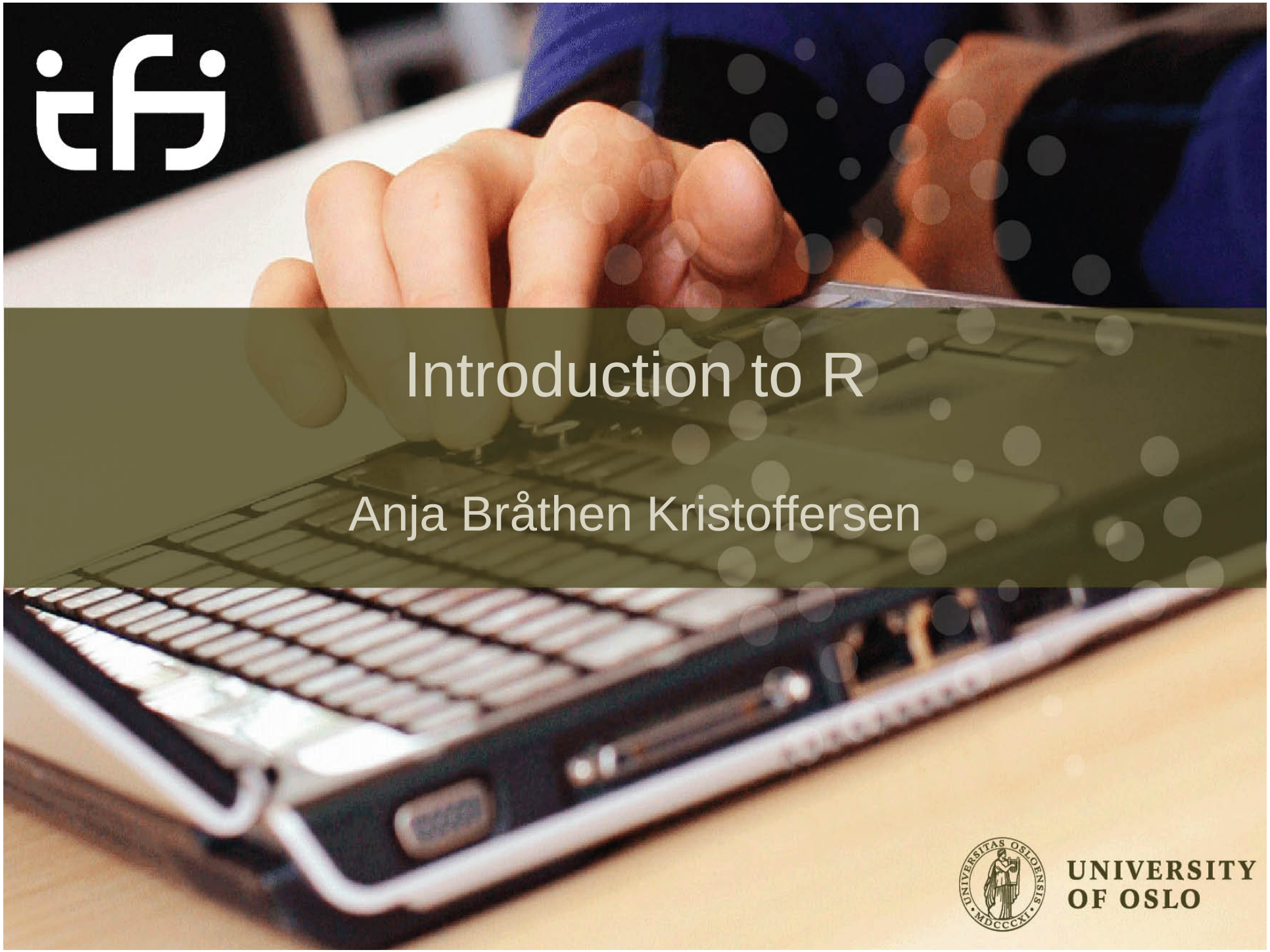

Introduction to R
Anja Bråthen Kristoffersen
UNIVERSITY
OF OSLO

R Session

- After R is started, there is a console waiting for input. At the prompt (>), you can enter numbers and perform calculations.

```
> 1 + 2
[1] 3
```

Variable Assignment

- We assign values to variables with the assignment operator "<-". Just typing the variable by itself at the prompt will print out the value.

```
> x <- 1
> x
[1] 1
```

Functions

- **Functions**
- R functions are invoked by its name, following by the parenthesis, and zero or more arguments. The following apply the function c to combine three numeric values into a vector.

```
> c(1, 2, 3)
[1] 1 2 3
```

Comments

- All text after the hash sign "#" is considered a comment.

```
> 1 + 1      # this is a comment
[1] 2
```

Getting Help

- R provides extensive documentation. For example, typing help(c) gives documentation of the function c in R.

  > help(c)

- If you are not sure about the name of the function you are looking for, you can perform a fuzzy search with the apropos function.

  > apropos("nova")
  [1] ".__C__anova" ".__C__anova.glm (and more)

Basic data types

- Numeric

```
> x <- 10.5       # assign a decimal value
> x               # print the value of x
[1] 10.5
> class(x)        # print the class name of x
[1] "numeric"
```

- Integer

```
> as.integer(3.14)    # coerce a numeric value
[1] 3
```

Basic data types

- Logical
  - A logical value is often created via comparison between variables.

```
> x  <-  1; y <- 2   # sample values
> z  <-  x > y      # is x larger than y?
> z                 # print the logical value
[1] FALSE
> class(z)       # print the class name of z
[1] "logical"
```

Basic data types (logical)

– Standard logical operations are "&" (and), "||" (or), and "!" (negation).

```
> u <- TRUE; v <- FALSE
> u & v        # u AND v
[1] FALSE
> u || v       # u OR v
[1] TRUE
> !u           # negation of u
[1] FALSE
```

Basic data types

- Character
  - A character object is used to represent string values in R. We convert objects into character values with the as.character() function:

```
> x <- as.character(3.14)
> x           # print the character string
[1] "3.14"
> class(x)      # print the class name of x
[1] "character"
```

Basic data types (character)

- Two character values can be concatenated with the paste function.

```
> fname <- "Anja"; lname <-"Kristoffersen"; mname <-"Bråthen"
> paste(fname, mname, lname, sep = " ")
[1] “Anja Bråthen Kristoffersen"
```

- To extract a substring, we apply the substr function. Here is an example showing how to extract the substring between the third and twelfth positions in a string.

```
> substr("Mary has a little lamb.", start=3, stop=12)
[1] "ry has a l"
```

Basic data types (character)

– And to replace the first occurrence of the word "little" by another word "big" in the string, we apply the sub function.

```
> sub("little", "big", "Mary has a little lamb.")
[1] "Mary has a big lamb."
```

– More functions for string manipulation can be found in the R documentation.

```
> help(sub)
```

Vector

- A vector is a sequence of data elements of the same basic type. Members in a vector are officially called components
- Here is a vector containing three numeric values 2, 3 and 5.
- > c(2, 3, 5)
  [1] 2 3 5

Vector

- Here is a vector of logical values.

```
> c(TRUE, FALSE, TRUE, FALSE, FALSE)
[1]  TRUE FALSE  TRUE FALSE FALSE
```

- A vector can contain character strings.

```
> c("aa", "bb", "cc", "dd", "ee")
[1] "aa" "bb" "cc" "dd" "ee"
```

- The number of members in a vector is given by the length function.

```
> length(c("aa", "bb", "cc", "dd", "ee"))
[1] 5
```

Combining vectors

- Vectors can be combined via the function c. For examples, the following two vectors n and s are combined into a new vector containing elements from both vectors.

```
> n <- c(2, 3, 5)
> s <- c("aa", "bb", "cc", "dd", "ee")
> c(n, s)
[1] "2"  "3"  "5"  "aa" "bb" "cc" "dd" "ee"
```

Part of a vector

```
> A <- c("aa", "AA", "bb", "BB", "cc", "CC")
> B <- c("aa", "bb", "cc", "dd", "ee", "ff")
> x <- A %in% B
> x
[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE
> A[x]
[1] "aa" "bb" "cc"
```

Vector arithmetics

- Arithmetic operations of vectors are performed member-by-member, i.e., memberwise.
  - suppose we have two vectors a and b.

```
> a <- c(1, 3, 5, 7)
> b <- c(1, 2, 4, 8)
```

  - multiply a by 5

```
> 5 * a
[1]  5 15 25 35
```

Vector arithmetics

```
> a <- c(1, 3, 5, 7)
> b <- c(1, 2, 4, 8)
```

– add a and b together, the sum would be a vector whose members are the sum of the corresponding members from a and b.

```
> a + b
[1]  2  5  9 15
```

– Similarly for subtraction, multiplication and division.

```
> a - b
[1]  0  1  1 -1
```

```
> a * b
[1] 1  6  20  56
```

```
>a/b
[1] 1.000  1.500  1.250  0.875
```

Vector arithmetics

- Recycling Rule
  - If two vectors are of unequal length, the shorter one will be recycled in order to match the longer vector. You will not get any warning or error from R.

```
> u <- c(10, 20, 30)
> v <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
> u + v
[1] 11 22 33 14 25 36 17 28 39
```

Vector index

- We retrieve values in a vector by declaring an index inside a single square bracket "[]" operator.

```
> s <- c("aa", "bb", "cc", "dd", "ee")
> s[3]
[1] "cc"
```

Vector index

- Negative Index
  - If the index is negative, it would remove the member whose position has the same absolute value as the negative index.

```
> s[-3]
[1] "aa" "bb" "dd" "ee"
```

- Out-of-Range Index
  - If an index is out-of-range, a missing value will be reported via the symbol NA.

```
> s[10]
[1] NA
```

Logical Index Vector

- A new vector can be sliced from a given vector with a logical index vector, which has the same length as the original vector.

```
> s <- c("aa", "bb", "cc", "dd", "ee")
> L <- c(FALSE, TRUE, FALSE, TRUE, FALSE)
> s[L]
[1] "bb" "dd"
```

- Or by indicating the positions:

```
>s[c(2,4)]
[1] "bb" "dd"
```

Matrix

```
>  A <- matrix(
+   c(2, 4, 3, 1, 5, 7),  # the data elements
+   nrow=2,               # number of rows
+   ncol=3,               # number of columns
+   byrow = TRUE)     # fill matrix by rows

> A                       # print the matrix
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

Matrix

```
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

- An element at the m^{th} row, n^{th} column of A can be accessed by the expression A[m, n]

```
> A[2, 3]     # element at 2nd row, 3rd column
[1] 7
```

- The entire m^{th} row A can be extracted as A[m,]

```
> A[2, ]      # the 2nd row
[1] 1 5 7
```

- The entire n^{th} column A can be extracted as A[,n]

```
> A[ ,3]      # the 3rd column
[1] 3 7
```

Matrix

```
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

- We can also extract more than one rows or columns

```
> A[ ,c(1,3)]  # the 1st and 3rd columns
     [,1] [,2]
[1,]    2    3
[2,]    1    7
```

- If we assign names to the rows and columns of the matrix, than we can access the elements by names.
-

```
> dimnames(A) = list(
+   c("row1", "row2"),        # row names
+   c("col1", "col2", "col3")) # column names
> A["row2",]    # print row 2 of  A
 col1 col2 col3
    1    5    7
```

Transpose

- Transpose of a matrix with the function t().

```
> t(A)        # transpose of A
     [,1] [,2]
[1,]    2    1
[2,]    4    5
[3,]    3    7
```

Combining Matrices

- The columns of two matrices having the same number of rows can be combined into a larger matrix.
- > B <- matrix(c(7, 4, 2), nrow=1, ncol=3)
  > B

```
     [,1] [,2] [,3]
[1,]    7    4    2
```

Combining Matrices

- Then we can combine the rows of A and B with rbind.
- > rbind(A, B)

```
     col1 col2 col3
row1    2    4    3
row2    1    5    7
        7    4    2
```

List

- A list is a generic vector containing other objects.

```
> n <- c(2, 3, 5)
> s <- c("aa", "bb", "cc", "dd", "ee")
> b <- c(TRUE, FALSE, TRUE, FALSE, FALSE)
> x <- list(n, s, b, 3)
> x
[[1]]
[1] 2 3 5
[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
[[3]]
[1]  TRUE FALSE  TRUE FALSE FALSE
[[4]]
[1] 3
```

List

- We retrieve a list slice with the single square bracket "[]" operator.

```
> x[2]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
```

- With an index vector, we can retrieve a slice with multiple members.

```
> x[c(2, 4)]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
[[2]]
[1] 3
```

List

- In order to reference a list member directly, we have to use the double square bracket "[[]]" operator

```
> x[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
```

- We can modify its content directly.

```
> x[[2]][1] = "ta"
> x[[2]]
[1] "ta" "bb" "cc" "dd" "ee"
> s                                   # s is unaffected
[1] "aa" "bb" "cc" "dd" "ee"
```

List

- Assign names to list members

```
> v <- list(bob=c(2, 3, 5),  john=c("aa", "bb"))
> v
$bob
[1] 2 3 5
$john
[1] "aa" "bb"
> v["bob"]
$bob
[1] 2 3 5
> v[["bob"]]
[1] 2 3 5
```

Data frame

```
> n <- c(2, 3, 5)
> s <- c("aa", "bb", "cc")
> b <- c(TRUE, FALSE, TRUE)
>  daf <- data.frame(n, s, b)
> daf
  n  s     b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
```

Read data from txt file

100	a1	b1
200	a2	b2
300	a3	b3
400	a4	b4

- Assume you have a file mydata.txt

```
> mydata <- read.table("mydata.txt")    # read text file
> mydata                                # print data frame
  V1 V2 V3
1 100 a1 b1
2 200 a2 b2
3 300 a3 b3
4 400 a4 b4
```

Read data from txt file

C1	C2	C3
100,1	a 1	b1
200,2	a 2	b2
300,3	a 3	b3
400,4	a 4	b4

- If the file is tab divided use sep="\t", especially important when space in text that should be read in one column.
- If the columns has names in the file, use header = TRUE

```
> mydata <- read.table("mydata.txt", sep="\t", header = TRUE, dec =",")
> mydata
    C1   C2   C3
1 100.1  a 1   b1
2 200.2  a 2   b2
3 300.3  a 3   b3
4 400.4  a 4   b4
```

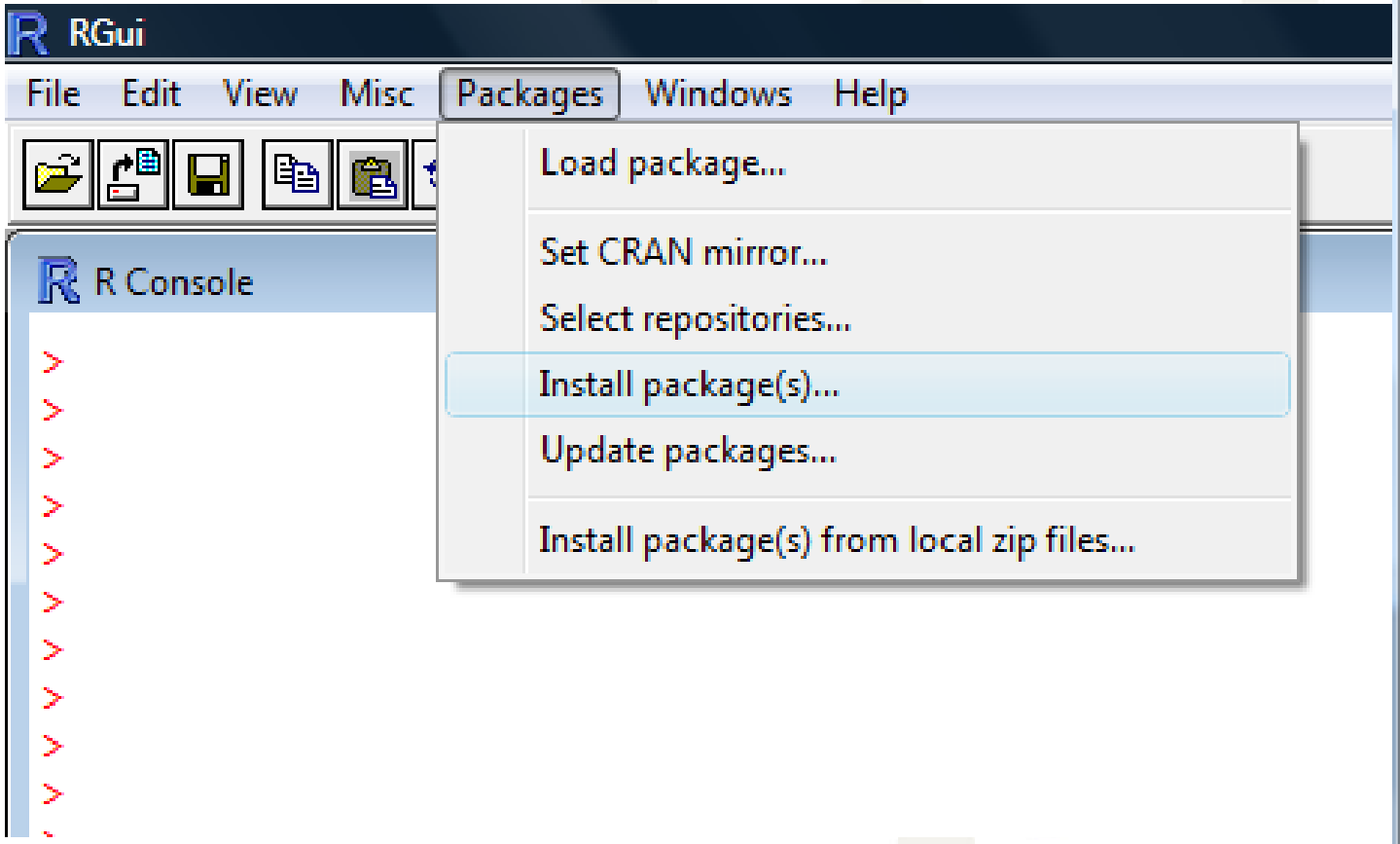
- For further optional arguments see help(read.table)

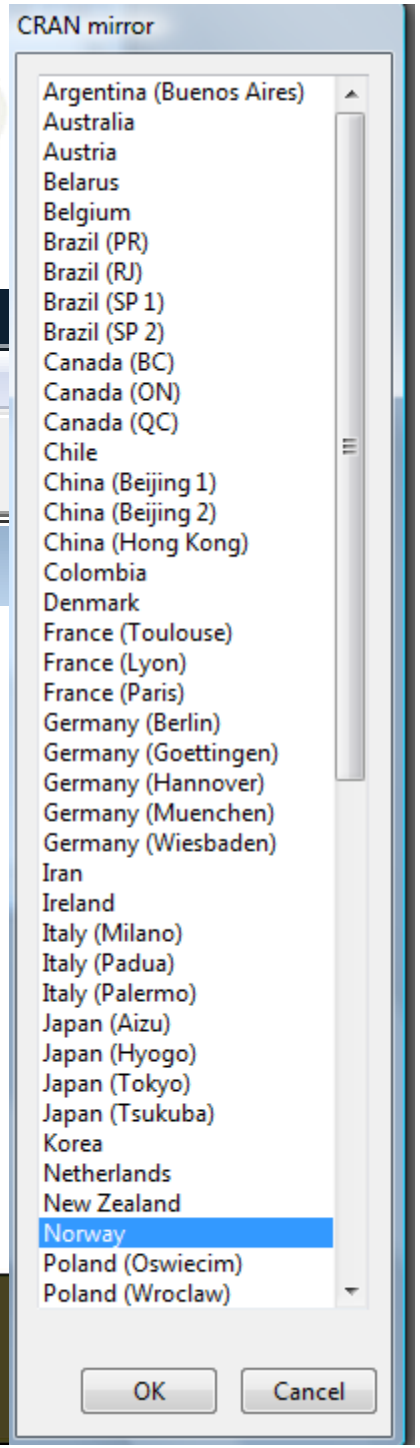
Read data from excel file

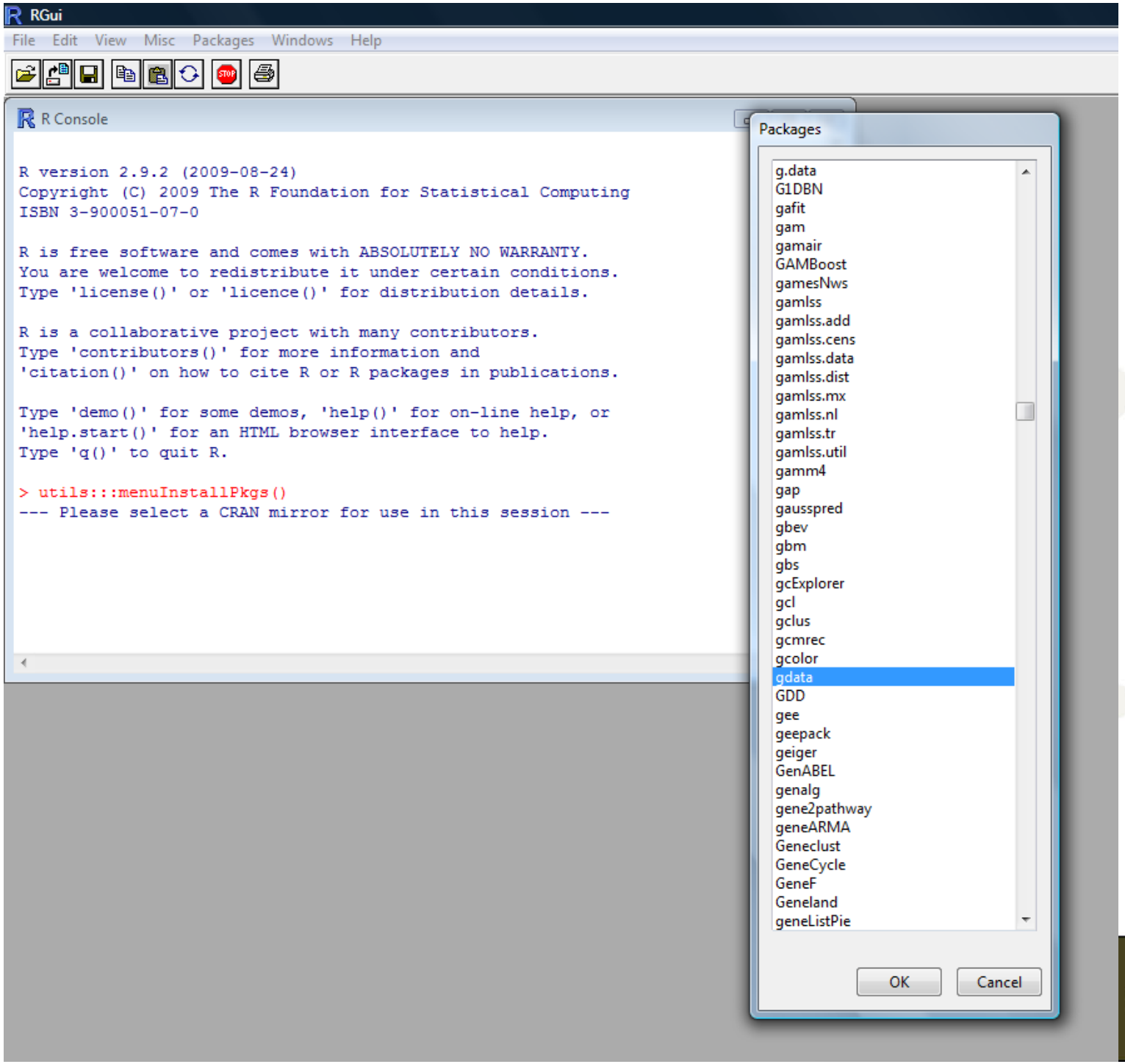
- Need the package gdata which is not in the core R library. The library must be installed and loaded into the R workspace before use, see next slide

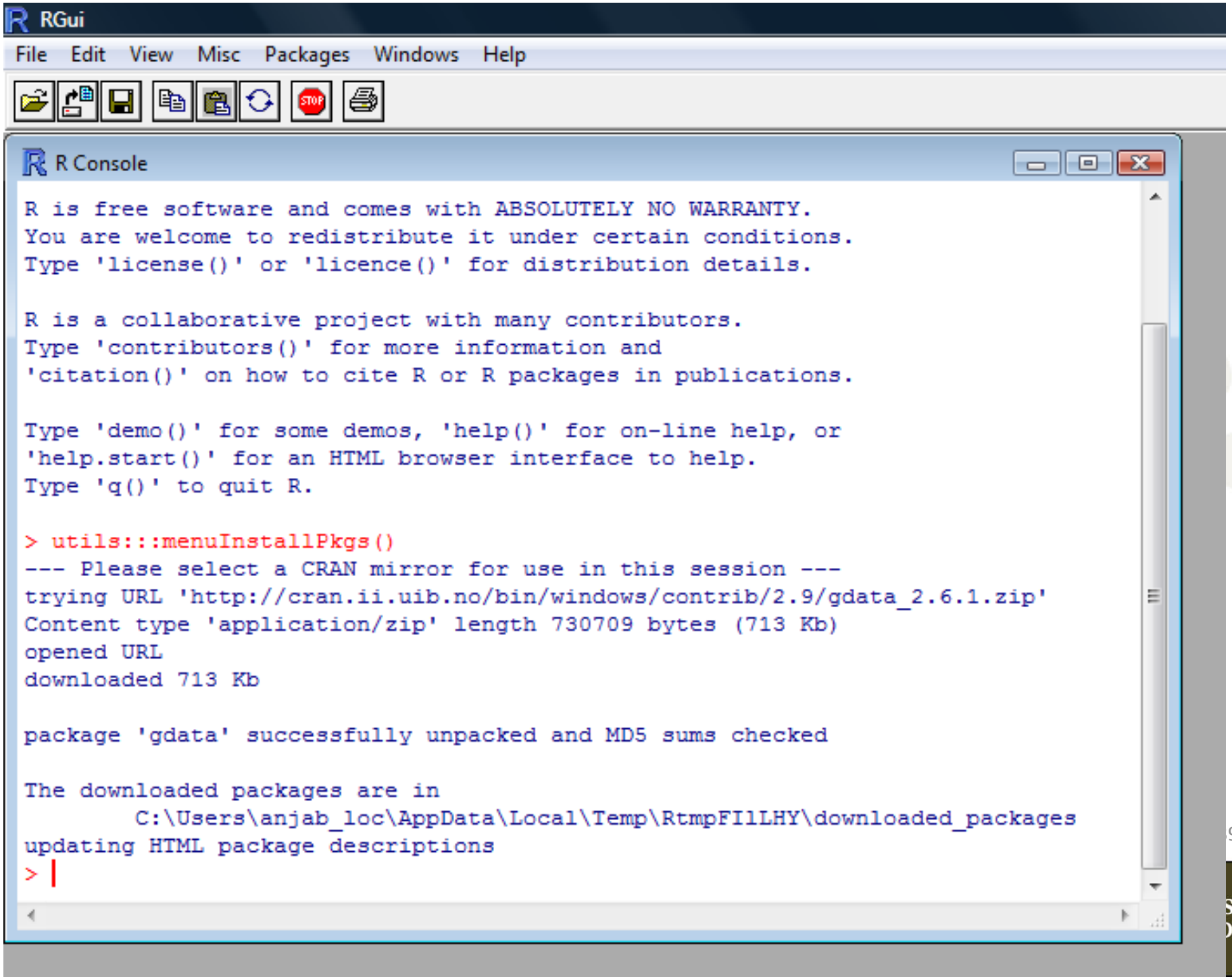
```
> library(gdata)                          # load the gdata package
> help(read.xls)                          # documentation
> mydata <- read.xls("mydata.xls")  # read from first sheet
```

Load packages





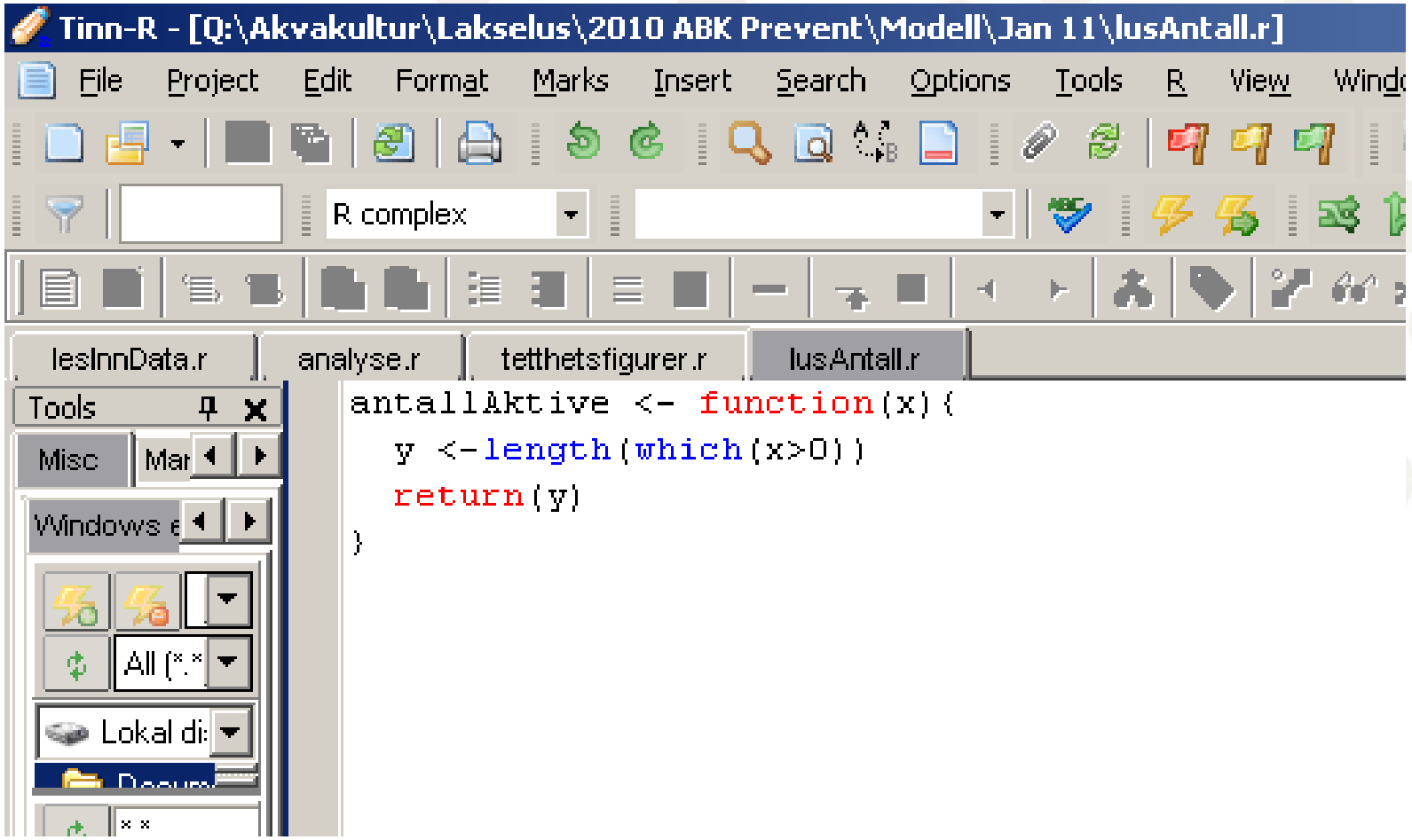



RGui
File Edit View Misc Packages Windows Help
R Console
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this session ---
trying URL 'http://cran.ii.uib.no/bin/windows/contrib/2.9/gdata_2.6.1.zip'
Content type 'application/zip' length 730709 bytes (713 Kb)
opened URL
downloaded 713 Kb
package 'gdata' successfully unpacked and MD5 sums checked
The downloaded packages are in
C:\Users\anjab_loc\AppData\Local\Temp\RtmpFI1LHY\downloaded_packages
updating HTML package descriptions
>

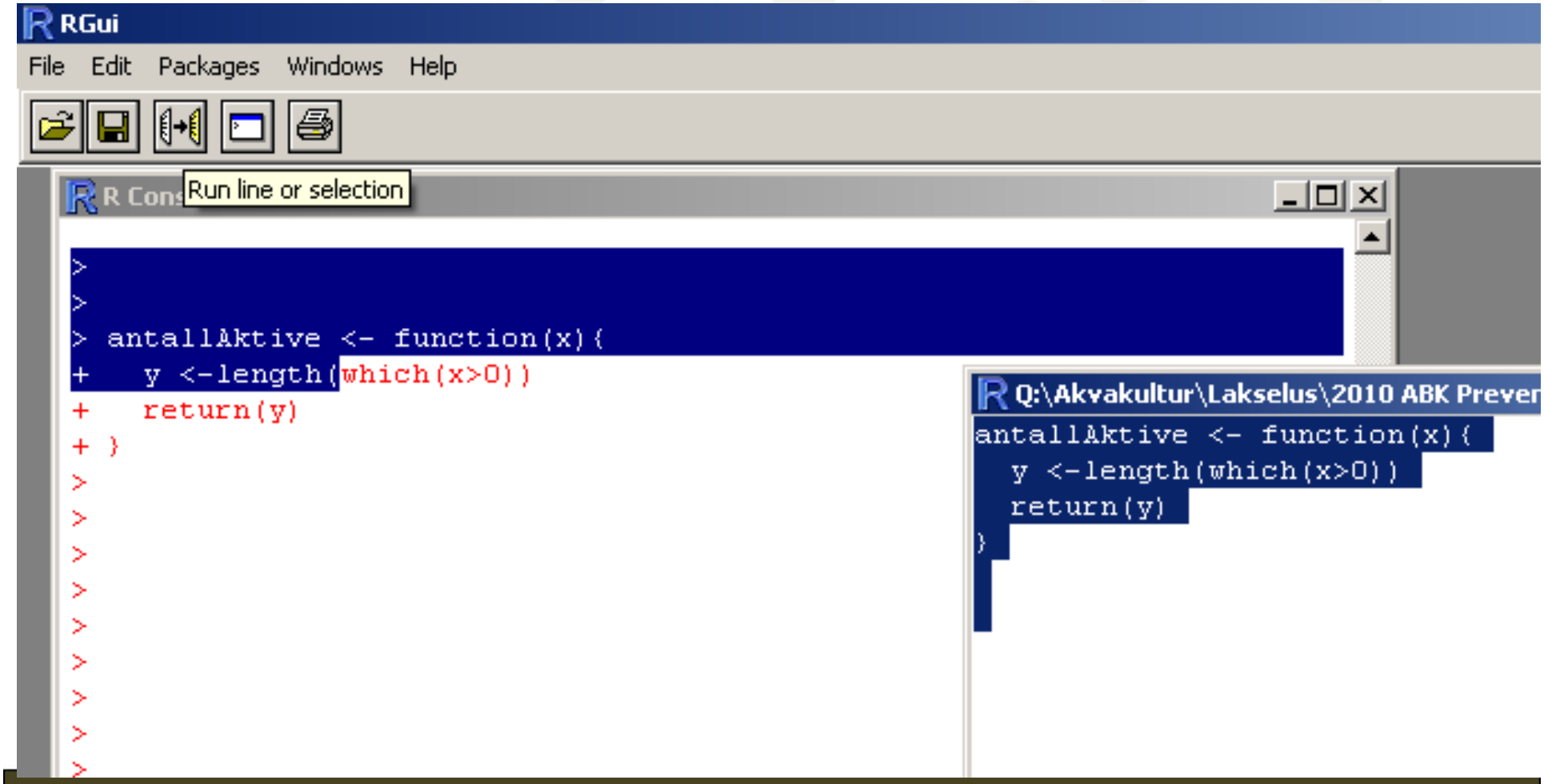
Save your script

- The name of your file should end with .r
- You could write your script in any text editor you like.

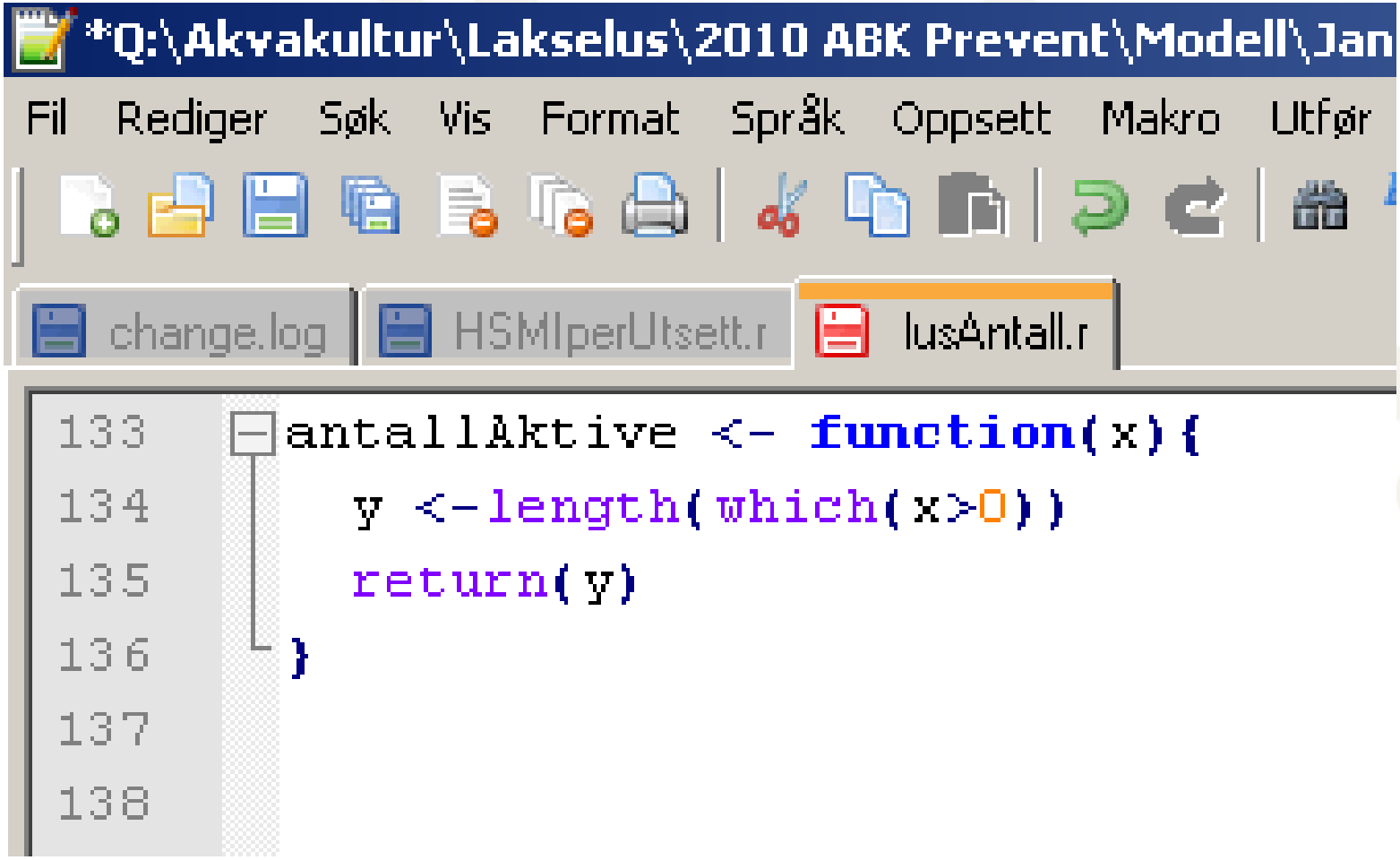
Tinn-R


Tinn-R - [Q:\Akvakultur\Lakselus\2010 ABK Prevent\Modell\Jan 11\lusAntall.r]
File Project Edit Format Marks Insert Search Options Tools R View
R complex
lesInnData.r analyse.r tetthetsfigurer.r lusAntall.r
Tools
Misc
antallAktive <- function(x){
y <-length(which(x>0))
return(y)
}

Open script in R: File – Open script



Notepad++


*Q:\Akvakultur\Lakselus\2010 ABK Prevent\Modell\Jan
Fil Rediger Søk Vis Format Språk Oppsett Makro Utfør
change.log
HSMIperUtsett.r
lusAntall.r
133 antallAktive <- function(x){
134 y <-length(which(x>0))
135 return(y)
136 }
137
138

Source of inspiration to this lecture:

- http://www.r-tutor.com/r-introduction/