These are notes to accompany the video series:
R for epidemiologists (beginners)
Watch it at: www.sva.se
And use this document as a guide to make your notes

For your reference: http://www.r-project.org/ (Search → R site search)

```
> citation()
```

1. Installing R and installing RStudio

R: http://cran.r-project.org/bin/windows/base/ (install first)

R Studio: http://www.rstudio.com/products/rstudio/download/

2. Introduction to the interface and R syntax

- → R as a big calculator
  - `1+1, 6*7`
  - > lines (command prompt) versus answers (bracket numbered lines)
  - Decimals with . `10.5` `10,5`
  - Assigning values to variables (we'll define ***objects*** more precisely later)
    - `var = 2` versus `var <- 2`
  - Breaking up over multiple lines
  - ESCape
  - Arrow up
- → Simple mathematical operators: + - * / ^ ()
- → Some other mathematical functions
  - Integer quotients: 119%/%13 [1] 9
  - Modulo (remainder): 119%%13 [1] 2
- → Object names
  - Case-sensitive: `var<-2` versus `Var<-24`
  - Cannot start with numbers
  - Cannot have spaces: `var_1; var.1`

3. Let us better stick to Rstudio…

- → Console
- → Bottom right: files, plots, packages
- → Top right: Environment
- → Editor – write and save scripts

4. Continuing with basic math in R:

➔ Logical values

Table 2.3. Logical operations.

Symbol	Meaning
!	logical NOT
&	logical AND
|	logical OR
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	logical equals (double =)
!=	not equal
&&	AND with IF
||	OR with IF
xor(x,y)	exclusive OR
isTRUE(x)	an abbreviation of identical(TRUE,x)

Source: Michael J. Crawley. The R Book.

- 15%%2==0 (test if odd or even)

➔ Simple mathematical functions

Table 2.1. Mathematical functions used in R.

Function	Meaning
log(x)	log to base e of x
exp(x)	antilog of x (e^x)
log(x,n)	log to base n of x
log10(x)	log to base 10 of x
sqrt(x)	square root of x
factorial(x)	$x!$
choose(n,x)	binomial coefficients $n!/(x!\ (n-x)!)$
gamma(x)	$\Gamma(x)$, for real x $(x-1)!$, for integer x
lgamma(x)	natural log of $\Gamma(x)$
floor(x)	greatest integer $< x$
ceiling(x)	smallest integer $> x$
trunc(x)	closest integer to x between x and 0 trunc(1.5) $=1$, trunc(-1.5) $=-1$ trunc is like floor for positive values and like ceiling for negative values
round(x, digits=0)	round the value of x to an integer
signif(x, digits=6)	give x to 6 digits in scientific notation
runif(n)	generates n random numbers between 0 and 1 from a uniform distribution
cos(x)	cosine of x in radians
sin(x)	sine of x in radians
tan(x)	tangent of x in radians
acos(x), asin(x), atan(x)	inverse trigonometric transformations of real or complex numbers
acosh(x), asinh(x), atanh(x)	inverse hyperbolic trigonometric transformations of real or complex numbers
abs(x)	the absolute value of x, ignoring the minus sign if there is one

Source: The R book, Michael J. Crawley.

Script used in **video1**:

```
#### Basic mathematical operations

119%/%13
119%%13
x <- 10
log(x)
exp(x)
sqrt(x)
12/7
floor(12/7)
ceiling(12/7)
y <- 12/7
floor(y)
ceiling(y)
round(12/7)
round(12/7, digits=1)
round(12/7, digits=2)

### Logical tests
3 < 4
248%%2==0
0 ==
0.000000000000001
a <- 6*7
(b = (a==42))
a <- 2
b <- -2
a > 0
(a>0)&(b>0)
(a>0)|(b>0)
a<2
a<=2
```

5. Number and vectors in R, simple manipulations

(The script used in the video is pasted in the next page)

- ➔ Create vectors
- ➔ Paste from clipboard (COLUMN format): x<- scan()
- ➔ Elements can be non-numeric; but all must be the same type
- ➔ Vector index
  - Single
  - Multiple
  - range
  - Duplicate
  - Out-of-order
  - Out of bond index
  - Negative index
- ➔ Basic operations
- ➔ Recycling
- ➔ Which x logical tests

Table 2.2. Vector functions used in R.

Operation	Meaning
max(x)	maximum value in x
min(x)	minimum value in x
sum(x)	total of all the values in x
mean(x)	arithmetic average of the values in x
median(x)	median value in x
range(x)	vector of min(x) and max(x)
var(x)	sample variance of x
cor(x,y)	correlation between vectors x and y
sort(x)	a sorted version of x
rank(x)	vector of the ranks of the values in x
order(x)	an integer vector containing the permutation to sort x into ascending order
quantile(x)	vector containing the minimum, lower quartile, median, upper quartile, and maximum of x
cumsum(x)	vector containing the sum of all of the elements up to that point
cumprod(x)	vector containing the product of all of the elements up to that point
cummax(x)	vector of non-decreasing numbers which are the cumulative maxima of the values in x up to that point
cummin(x)	vector of non-increasing numbers which are the cumulative minima of the values in x up to that point
pmax(x,y,z)	vector, of length equal to the longest of x, y or z, containing the maximum of x, y or z for the ith position in each
pmin(x,y,z)	vector, of length equal to the longest of x, y or z, containing the minimum of x, y or z for the ith position in each
colMeans(x)	column means of dataframe or matrix x
colSums(x)	column totals of dataframe or matrix x
rowMeans(x)	row means of dataframe or matrix x
rowSums(x)	row totals of dataframe or matrix x

Source: The R book, Michael J. Crawley.

Vectors: script used in video 2

```
#Creating vectors
(var1 <- 1:15)
(var2 <- c(2,3,7))
(x <- c(var1,var2))
(y <- c(1:5,rep(6,4),7:8))

#Pasting from clipboard
myExcel.data <- scan()

#Non numerical data types
(fruits <-
c("apples","oranges","grapes"))

(mix <- c(1,"apple",3,4))

numbers <- -5:5
test <- numbers>0

#Indexing
1:200
numbers[3]
numbers[3:5]
fruits[c(1,3)]

liked.fruits <- fruits[c(1,3)]
liked.fruits <- fruits[-2]

fruits
fruit[3] <- kiwis
fruit[3] <- "kiwis"

liked.fruits
liked.fruits[3] <- "kiwis"

cows.tested <-
c("daisy","nora","earTag1267")

cows.tested[4] <- "mimosa"

cows.tested <-
c(cows.tested,"princess")

cows.tested[10] <- "prima"
cows.tested[11]

liked.fruits

eaten.fruits <-
liked.fruits[c(1,3,1,1)]

liked.fruits[c(3,1)]

a<- 1:10
b<- 1:11
a+b

#Operations with vectors
x <- 1:15
x*2

a<- 1:10
b<- 11:20
a+b

b<-10:20
a+b
c <- a+b

numbers <- -5:5

numbers>0
which(numbers>0)

result1 <- numbers>0
result2 <- which(numbers>0)

result3 <-
numbers[which(numbers>0)]

result4 <- numbers[numbers>0]
result1==result2
result3==result4

#propertis of vectors
length(numbers)

class(numbers)
class(result1)
class(cows.tested)


#Functions applied to vectors
a <- 1:10
(b <- a/3)

floor(b)
max(b)
min(b)
mean(b)

(x <- rpois(15, 5))
sort(x)
order(x)

(x <- c(1,5,3,4,2))
(groups <-
c("a","b","c","d","e"))
groups[order(x)]
```

6. Matrices

- ➔ Building
  - Using matrix()
  - From vectors
- ➔ Indexes
- ➔ dim()
- ➔ cbind() rbind()
- ➔ Matrix math

7. Objects and their properties

- ➔ Character
- ➔ length() and dim()
- ➔ class()

8. Factors

- ➔ as.factor() x as.integer() x as.character()
- ➔ ordered factors: ordered()

Table 2.4. Functions for testing (is) the attributes of different categories of object (arrays, lists, etc.) and for coercing (as) the attributes of an object into a specified form. Neither operation changes the attributes of the object.

Type	Testing	Coercing
Array	is.array	as.array
Character	is.character	as.character
Complex	is.complex	as.complex
Dataframe	is.data.frame	as.data.frame
Double	is.double	as.double
Factor	is.factor	as.factor
List	is.list	as.list
Logical	is.logical	as.logical
Matrix	is.matrix	as.matrix
Numeric	is.numeric	as.numeric
Raw	is.raw	as.raw
Time series (ts)	is.ts	as.ts
Vector	is.vector	as.vector

Source: The R book, Michael J. Crawley.

9. Data frames

- ➔ data frames versus matrices
- ➔ categorical variables - factors
- ➔ data.frame(); as.data.frame()
- ➔ cbind(), rbind()
- ➔ Rownames(), colnames()
- ➔ str()
- ➔ summary()

➔ table()

10. Arrays

➔ array()
➔ dim(); dimnames()
➔ indexing

11. Lists

➔ list(); as.list()
➔ ~~dim();~~ length()
➔ indexing

Script used in video3:

```
## Building matrices in R
#Using matrix()
M = matrix(
   c(2, 4, 3, 1, 5, 7),
   nrow=2,
   ncol=3,
   byrow = TRUE)

matrix(1:12,3,4)
matrix(1:12,ncol=4,nrow=3)

#From vectors
ID <- 1:15
weight <- rnorm(15,mean=75,sd=4)
height <- norm(15,
          mean=1.65,sd=0.2)

study.group <-matrix(
          c(ID,weight,height),
          ncol=3,byrow=FALSE)
study.group <- cbind(
          ID,weight,height)

dim(study.group)

study.group <- rbind(study.group,
                     c(16,7,1.6))

dim(study.group)

#Indexing
study.group[3,2]
study.group[3,2:3]
study.group[3,c(1,3)]
study.group[,2]
study.group[3,]
study.group[16,2] <- 87
```

```
#Matrix math
M*2
M+N
M*N

# Objects and their properties
class(study.group)

cowName <- c("Daisy","Bia",
      "Mimosa","Princess",
      "lili","Nora","earTag1378",
      "ceci","kiki","Amelia")
Lactation <- c(1,2,5,3,2,2,4,1,4,3)
Vaccinated <- c(TRUE,FALSE,FALSE,
          TRUE,TRUE,TRUE,TRUE,
          FALSE, TRUE,TRUE)
ELISA <- c(0.1,0.5,0,2.5,
          1.2,0,1.5,0,0,2.5)

class(cowName)
class(Lactation)
  is.integer(Lactation)
  Lactation <-
as.integer(Lactation)
  is.integer(Lactation)
  class(Lactation)
class(Vaccinated)
class(ELISA)
  as.integer(ELISA)

v <- 1:5
v[10] <- 10
v
is.na(v)
which(is.na(v))
sum(is.na(v))
v[is.na(v)]
```

```
cow.study <- cbind(cowName,
          Lactation,Vaccinated)
class(cow.study)
cow.study


cow.study<-data.frame(
          cowName,Lactation,
          Vaccinated)
class(cow.study)
class(cow.study[,"cowName"])
cow.study[,"cowName"] <-
          as.character(
          cow.study[,"cowName"])

feed.group <- c(
          rep("A",5),rep("B",5))
class(feed.group)

cow.study<- cbind(
          cow.study,feed.group)
class(cow.study[,"feed.group"])

#Indexing
cow.study[3,"Lactation"]
cow.study[,2]
cow.study[,"Lactation"]

cow.study$Lactation

cow.study[cow.study$
          cowName=="Mimosa",]

#More properties of data frames and
matrices
dim(cow.study)
nrow(cow.study)
ncol(cow.study)
dim(cow.study)[1]
dim(cow.study)[2]

rownames(cow.study)
colnames(cow.study)
dimnames(cow.study)

original.names <-
colnames(cow.study)

colnames(cow.study) <- c(
          "name","lact","vacc",
          "feed")
cow.study

colnames(cow.study) <-
           original.names

#Remember, R is not meant to SEE
data
str(cow.study)
summary(cow.study)
head(cow.study,5)
tail(cow.study)
table(cow.study$Vaccinated)
table(cow.study$Vaccinated,cow.stud
y$Lactation)



#### Arrays

result <-
array(rpois(18,5),dim=c(2,3,3))
dimnames(result)<-list(
  day=1:2,
  cow=1:3,
  test=1:3)

result[2,2,3] #day2,cow2,test3

#### List

study.group
cow.study
(vector1 <- 3:8)
(vector2 <- c("a","b","c"))
(value1 <-5)

myList <- list(
          study.group,cow.study,
          vector1,vector2,value1)

dim(myList)
length(myList)

myList[[1]]
myList[[3]]
```

12. Built in datasets

- ➔ data()
- ➔ To access data from a particular package, use the package argument, for example:
  - data(package="rpart")
  - data(Puromycin, package="datasets")

If a package has been attached by library, its datasets are automatically included in the search.

13. Functions

- ➔ Function(arguments)
  - Default, optional
  - Order (x use of explicit argument name)
- ➔ ?sum
- ➔ apropos("nova")

14. Functions in arrays/data frames

- ➔ Any individual values or dimensions
- ➔ Specific functions for matrices
  - rowSums()
  - colSums()
  - rowMeans()
- ➔ apply()
- ➔ tapply → tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)
- ➔ sapply – to each unit in a vector
- ➔ lapply – to lists

15. Getting help to figure out which functions to use

?mean ??average apropos("nova") google it!

16. Explore a few functions on your own:

- ➔ paste()
- ➔ rep()
- ➔ sample()
- ➔ seq()
- ➔ union, intersect and setdiff

17. Packages

- ➔ ?? function needed
- ➔ install.packages("")
- ➔ library() require()
- ➔ help(package = "ggplot2")
- ➔ Package tab in R Studio

Script used in video4:

```
# Built in datasets
data(chickwts)
head(chickwts)
dim(chickwts)
str(chickwts)

data(trees)

## Functions
#finding
?average
??average
?anova
apropos("nova")

#mandatory
mean()
mean(chickwts$weight)

#optional
  chickwts$weight[35] <- NA
mean(chickwts$weight)
?mean
mean(chickwts$weight,
          na.rm=TRUE)

#order
mean(chickwts$weight,
          na.rm=TRUE,trim=0.05)
mean(chickwts$weight,0.05,TRUE)
mean(chickwts$weight,TRUE,0.05)
data(chickwts)

# Applying functions to >1D
objects
colSums(trees)
colMeans(trees)
?apply
apply(trees,2,mean)

#...
trees[1,1]<-NA
apply(trees,2,mean)
apply(trees,2,mean,na.rm=TRUE)

?aggregate
aggregate(chickwts$weight ~
          chickwts$feed,
          FUN="mean")
aggregate(weight ~ feed,
          FUN="mean",
          data=chickwts)
aggregate(chickwts$weight,
      by=list(chickwts$feed),
          FUN="mean")

?glm
glm(weight ~ feed,
       data=chickwts)
       #family = gaussian
model1 <- glm(weight ~ feed,
          data=chickwts)
class(model1)
str(model1)
summary(model1)
model1$residuals
model1$coefficients

#more classes
?ts
x <- 1:30
plot(x)
x <- ts(x, start=1,
          frequency=365)

#ts {stats}
#detach("package:stats",
unload=TRUE)
x <- ts(x,start=1,
          frequency=365)

#### Loading packages
install.packages("MASS")
library(MASS)
require(MASS)
help(package = "MASS")

data(painters)
tapply(painters$Composition,
       painters$School, mean)
aggregate(Composition~School,
      FUN="mean",data=painters)

sapply(3:7, seq)

#data(package="rpart")
data(Puromycin,
package="datasets")
head(Puromycin)

ts <- rnorm(500,mean=0,sd=2)
?cusum
??cusum
install.packages("qcc")
?cusum
library("qcc")
cusum(ts
```

18. Data from Excel

➔ The common mistake

control	preheated	prechilled
6.1	6.3	7.1
5.9	6.2	8.2
5.8	5.8	7.3
5.4	6.3	6.9

x

Response	Treatment
6.1	control
5.9	control
5.8	control
5.4	control
6.3	preheated
6.2	preheated
5.8	preheated
6.3	preheated
7.1	prechilled
8.2	prechilled
7.3	prechilled
6.9	prechilled

(Use of Pivot tables in Excel solves visualization issues with keeping it all in the right format (second)).

➔ Save as .csv
- English windows: it's truly a COMMA separated values file
- Others: the separator is usually ";"

➔ Common errors with reading table
- Column names cannot have spaces (for instance a column name "Sample pH" would cause the process to fail)
- Factors can only have spaces if you are not using a space as the separator (file is a csv or separator is ";")

➔ If you use space as a separator, then missing values must be substituted first to NA

19. Getting data in and out of R

➔ setwd(); getwd()
➔ DATA IN: read.table(); read.csv(); read.csv2()
- library(foreign) for other types

➔ DATA OUT: write.table(); save(); save.image()
-

➔ read.csv2
➔ read.table("file.name.xxx",sep=";", header=TRUE)
-

20. Object manipulation and the R environment

➔ The R environment : ls()
➔ Save environment: save.image(file="myfile.RData")
➔ Load environment
- load("Rmeeting.RData ")

➔ rm(), keep()

Script used in video5:

```
## Right data format in Excel
## Data in
getwd()
setwd() #set manually
        #projects in RStudio
    #pasted from Explorer:
    #C:\Users\dorea.meyer\mydata
#setwd("C:\\Users\\dorea.meyer\\mydata")
#setwd("C:/Users/dorea.meyer/mydata")

?read.table
weight.data <- read.csv("animal_weights.csv")
#  weight.data <- read.csv(
#     "C:/Users/dorea.meyer/mydata/animal_weights.csv")
#weight.data <- read.csv2("animal_weights.csv")
#weight.data <- read.csv("animal_weights.csv", sep=";", dec=",")
str(weight.data)
weight.data <- read.csv("animal_weights.csv",
                        stringsAsFactors=FALSE)
str(weight.data)
#row.names
#col.names

## NAs
is.na(weight.data)
weight.data[weight.data==""]<- NA
is.na(weight.data)

#install.packages("foreign")
require(foreign)
help(package="foreign")

unique(weight.data)
duplicated(weight.data)
  duplicated(weight.data[,2:4])
complete.cases(weight.data)

summary1 <- mean(weight.data$weight)
summary2 <- aggregate(Weight ~ Group,
                      FUN="mean",
                      data=weight.data)
#Data out
write.table(weight.data,file="weight.data.csv",
            sep=",") #getwd()
                      #row.names, col.names
save(weight.data,file="weight.data.RData")
save.image(file="weight.data.RData")

#Save workspace using RStudio
#clear workspace
rm(list=ls())
#Load saved files or workspace
    load("weight.data.RData")
#rm()
#keep()
```

21. Use of an editor to save scripts

- ➔ #comments
- ➔ Load a batch of code: warning x errors
- ➔ Head
  - rm(list=ls())
  - packages
  - setwd() ; getwd()
- ➔ Import data
- ➔ Analysis
  - Align commands
  - Organize levels
  - USE comments
- ➔ Export data
- ➔ Save image

22. RStudio and scrip management

- ➔ R PROJECT
- ➔ All types of files
- ➔ Sections, chunks, etc
- ➔ Collapsible code

23. Graphs

➔ Plots with two continuous variables

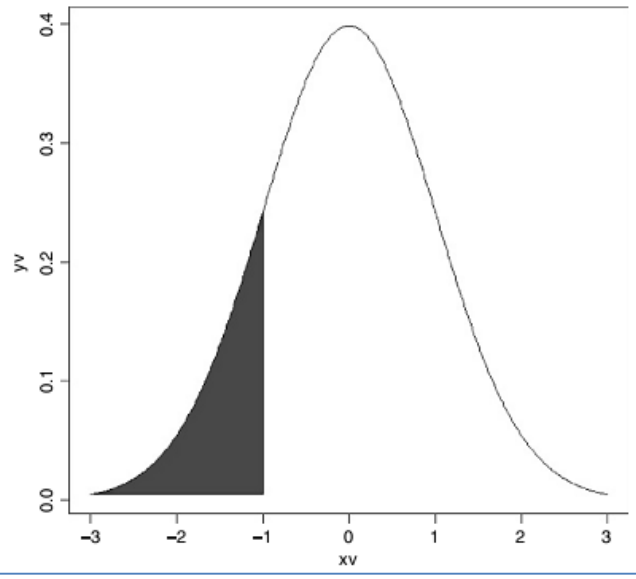
- plot()
- lines()
- labels
- points()
- text()
- arrows()
- polygon()
- pch
- legend
- col

The polygon function can be used to draw more complicated shapes, even curved ones. In this example we are asked to shade the area beneath a standard normal curve for values of x that are less than or equal to -1. First draw the line for the standard normal:

```
xv<-seq(-3,3,0.01)
yv<-dnorm(xv)
plot(xv,yv,type="l")
```

Then fill the area to the left of $xv \leq -1$ in red:

```
polygon(c(xv[xv<=-1],-1),c(yv[xv<=-1],yv[xv==-3]),col="red")
```



➔ Plots with a continuous and a categorical variables

- Box-plot - Plot(catX,y)
- Bar plot – barplot(y)

➔ Plot for a single variable

- hist()
  - breaks=
  - adding distribution line: (calculate parameters first – p.e. mean, stdev, then use dxxx) → lines (xrange,ys*xlength)

24. Working with dates

There are two basic classes of date/times. Class POSIXct represents the (signed) number of seconds since the beginning of 1970 as a numeric vector: this is more convenient for including in dataframes. Class POSIXltis a named list of vectors closer to human-readable forms, representing seconds, minutes, hours, days, months and years.

➔ Sys.time() date()

```
> class(date())    [1] "character"
> class(Sys.time())         [1] "POSIXct" "POSIXt"

> date<-as.POSIXlt(Sys.time())
> date                      [1] "2013-01-11 10:35:28 CET"
> date$sec                            [1] 28.45003
> date$min                            [1] 35
> date$hour                           [1] 10
> date$mday                           [1] 11
> date$wday                           [1] 5
> date$yday                           [1] 10
```

- Months: 0 to 11
- Weekdays: 0 to 6

➔ Type in time

➔ Recognizing time – strptime

%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time, locale-specific
%d	Day of the month as decimal number (01–31)
%H	Hours as decimal number (00–23) on the 24-hour clock
%I	Hours as decimal number (01–12) on the 12-hour clock
%j	Day of year as decimal number (001–366)
%m	Month as decimal number (01–12)
%M	Minute as decimal number (00–59)
%p	AM/PM indicator in the locale
%S	Second as decimal number (00–61, allowing for two 'leap seconds')
%U	Week of the year (00–53) using the first Sunday as day 1 of week 1
%w	Weekday as decimal number (0–6, Sunday is 0)
%W	Week of the year (00–53) using the first Monday as day 1 of week 1
%x	Date, locale-specific
%X	Time, locale-specific
%Y	Year with century
%Z	Time zone as a character string (output only)

➔ strptime(excel.dates,format="%d/%m/%Y")

25. Loops and conditional execution

- ➔ Standard logical operations are "&" (and), "|" (or), and "!" (negation)
  - help("&")
- ➔ if (expr_1) expr_2 else expr_3
- ➔ ifelse(condition, a, b)
- ➔ for (name in expr_1) expr_2
  - break
  - next
- ➔ while (condition) expr
- ➔ avoiding loops – use of vector functions
  - for (i in 1:length(y)) { if(y[i] < 0) y[i] <-0}
  - versus: y[y<0]<-0
- ➔ Do not 'grow' data sets in loops or recursive function calls