R tutorial / cheatsheet:

Getting help:
```
?functionname
??searchterm
Functionname       #prints code
```

Datastructure information:
```
class(), attr(), rownames(),
colnames(), dim(), length(), nrow(),
ncol(), is.vector(), is.numeric(),
is.data.frame()
```

Datastructure manipulation:
```
c(), rbind(), cbind()
as.vector(),as.numeric(),
as.data.frame()
```

Boolean operators:
```
==, !=, &&, ||
exists(), is.na(), is.null(),
is.finite()
```

Filesystem:
```
getwd(), setwd()
dir(), file()
```

Distribution functions:
```
runif(), rnorm(), rgamma(), rbeta()
```

Loading Libraries:
```
require(<lname>)
library(<lname>)
```

Printing loaded objects and functions:
```
 ls()
```

Reading/Writing datasets to disk:
```
data <- read.table(file="out.txt")
write.table(data, file="out.txt")
```

Handling large datasets:
```
head(), tail()
```

Matrices
```
testmatrix <- matrix(1:8,8,4) # Empty matrix 8 rows 4 cols
testmatrix[1,]                # First row
testmatrix[,1]                # First column
vector("list",10)             # Empty list with 10 NULL
testdataframe <- as.data.frame(testmatrix)
testdataframe$V1          #(First column, auto column on data.frame)
testmatrix[,-2]               # Remove second column
testmatrix[-c(1,4,5),]  # Print without row 1st , 4th and 5th column
```

Repeat and iteration:
```
for(var in seq) expr
while(cond) expr
apply(FUN,MARGIN,data)
#USE APPLY not FOR ! Margin: cols 1, rows 2, or both c(1,2)
lapply(FUN,data)              #Apply over a list
```

Missing data:
```
any(is.na(testmatrix))        # Any to all values in the matrix
testmatrix[2,1] <- NA
testmatrix[5,2] <- NA
any(is.na(testmatrix))        # Any to all values in the matrix
na.omit(testmatrix)
```

Sorting and matching
```
names1 <- paste("ind",1:10,sep="")   #Create names
names2 <- paste("ind",5:15,sep="")   #Create names
names1 %in% names2
names2 %in% names1
which(names1 %in% names2)
which(names2 %in% names1)
```

Text manipulation:

R isn't very good with text; however there are some neat tricks:

```
cat()                     # combine multiple strings to a large one
cat("ind",1:10,sep="")
paste()                   # paste things together pair wise
paste("ind",1:10,sep="")
```

There are GREP options and string functions:
```
nchar, substr, strsplit
```

Plotting options:

Plotting is powerful the basic call to setup a window for use:

```
plot(x=c(-1,1),y=c(-10,25), xlab="D", ylab="R", main="F", type="n")

image()  #for matrices (to make heatmaplike plots)
hclust()  #for matrices (clustering)
```

Now add lines/points y calling the functions
```
p <- NULL
p$x <- 25
p$y <- 120
points(p)
```

However most objects can be plotted directly, or have specialized plotting routines.

Creating your own functions:

```
Div2 <- function(x=NULL){
  if(!is.null(x)){
    x <- x/2
    x
  }else{
    stop("please supply an x")
  }
}

Div2(25)        #Call the function
Div2(1:100)     #Call the function
```

Parameters:

```
Half <- function(x, method=c("A","B","C")){
    supported <- c("A","B","C")
    method <- pmatch(method, supported)
}
```

Error Handling:
```
stop("Something went terrible wrong")
warning("Some minor thing")
```

Installing packages

Using the Rgui: Packages -> install Packages
Using the commandline: R CMD INSTALL package_name.zip
Using the commandline: R CMD INSTALL package_name.tar.gz

Creating a basic R-script

Create a new file called script.R

```
numbers <- runif(100)                         # Generate 100 random numbers
jpeg(file="graphoutput.jpg")                  # Plot output to he JPEG file
hist(numbers)                                 # Make a histogram
dev.off()                                     # Close the JPEG
q("no")                                       # Quit without saving anything
```

Execute the script by running:

```
    R CMD BATCH script.R
```

A file script.Rout will be created this holds the 'verbose' output of the script (stdout)
Normally the last line in a script file will be something like:

```
    write.table(results,file="out.txt",sep="\t")
```

Basic statistics:

Doing a t-test between two conditions

```
    cond1 <- rnorm(100)
    cond2 <- runif(100)
    t.test(cond1,cond2)
```

Basic linair model:

```
    m1 <- as.factor(round(runif(100)))
    m2 <- as.factor(round(runif(100)))
    modelm2onm1 <- lm(cond1~m1+m2)
    modelm1onm2 <- lm(cond1~m2+m1)
    anova(modelm2onm1)
    anova(modelm1onm2)
```

And R supports a large number of statistical tests, machine learning algorithms, etc

Data to and from Molgenis

Basic exploration of data: find.data(), add.data(), remove.data()
Retrieving data: find.datamatrix(), add.datamatrix(), remove.datamatrix()

But also for all object types specialized functions are created to map to those objects, like
in the case of genetic markers we can use:
find.markers(), add.markers(), remove.markers()