R Cheat Sheet: Basic List of Useful Functions

Built-in constants

```
LETTERS; letters; month.abb; month.name; pi
```

Object creation

```
sz <- 20; x <- 4; t <- 'c' #length 1 vector
a <- letters[ceiling(runif(sz,0.00001,26))]
names(a) <- LETTERS[1:sz] # a named vector
i <- 1:sz; j <- i + rnorm(sz, 0, 2)
z <- exp((0+1i)*pi) + 1+0i #complex numbers

d <- as.Date('2012-01-01') + seq(1, sz)
f <-factor(rep(1:x, sz/x), levels=x:1)
df <- data.frame(a=a, d=d, f=f, i=i, j=j)
m <- matrix(rnorm(x^2), nrow=x, ncol=x)
l <- list(df, m, a, z)
```

Object inspection

```
mode(i); class(df); typeof(j); dput(a)
str(l); summary(df); head(df); tail(df)
attributes(l); is.null(i)

names(a); dimnames(df); colnames(df)
rownames(df); dim(df); nrow(m); ncol(df)
is.list(l); is.factor(f); is.complex(z)
is.character(a); is.matrix(m); is.real(z)
is.numeric(z); is.integer(i); is.vector(i)
is.data.frame(df); is.ordered(f)

isTRUE(all.equal(1:10, as.numeric(1:10)))
identical(1:10, as.numeric(1:10)) # FALSE
```

Utility functions

```
assign('variable.name', 5) # like: <-, ->
c(i, j); rep(NA, 20) # concatenate; repeat
append(l, list(c(1, 2, 3))); seq_along(a)
seq(from=5,to=100, by=5); seq_len(nrow(df))

sort(a); order(a); rank(a); rev(a)
any(i %in% c(1,3,5)); all(i %in% c(1,3,5))
which(a %in% letters[21:26]); match('c', a)
max <- ifelse(i > j, i, j) # vectorised if
switch(txt, a={print('a')}, b={print('b')},
      {print('default')} ) #not vectorised

df <- transform(df, k=j+i)
df <- within(df, s <- i/j) #merge(df1, df2)
x <- with(df, j+5); df <- cbind(df, z)
row.df <- head(df, 1); rbind(df, row.df)
df$f <- reorder(df$f, df$j, mean)

# many similar style conversion functions:
as.integer(f); as.data.frame(m) # etc.
```

Maths

```
is.na(i); is.nan(j); is.null(d)
is.finite(j); is.infinite(j)
Re(z); Im(z) # real and im coefficients

abs(j); sqrt(i); log(i); log10(j); exp(j)
ceiling(j); floor(j); round(j, digits=2)
trunc(j); sin(j); cos(j); tan(j); asin(0.5)
acos(0.5); atan(1) # atan2(y, x);
sum(j) prod(j); cumsum(j); cumprod(j)
```

Stats

```
length(j); sum(j); min(j); max(j); range(j)
cut(i, 5); mean(j); median(j); sd(j)
var(i); cov(i, j); cor(i, j)
diff(j, lag=1, diff=1) # difference data
rnorm(n=10, mean=0, sd=1)   # normal dist
runif(n=10, min=1, max=100) # uniform
# Also poisson and binomial random numbers

r <- lm(j ~ i, data=df); summary(r)
anova(r); residuals(r); coef(r);
plot(r); plot(r$fitted); plot(r$resid)
# Also glm() gam() lme() lmer() nls() etc.
```

Character strings

```
as.character(j); toString(l)
nchar(a); B <- toupper(a); b <- tolower(B)
s <- "the cow jumped over the moon."
sub('the', 'a', s) #  -> a cow … the moon
gsub('the', 'a', s) # -> a cow … a moon
substr(s, 5, 7); substr(s, 5, 7) <- "dog"
substr(s, 5, 7) <- "monkey" # FAIL!
paste('a', 'b', 'c', sep='; ') # 'a; b; c'
strsplit(s, ' ') # -> list; regex pattern
grep('the', s) # see also: grepl and agrep
make.unique(a) # -> change dups in vector
format(j, digits=2); sprintf("%d: %s", i,a)
format(d, format="%A %Y-%b-%d")
```

Dates

```
# Date: a double; days since 1970-01-01
x<- as.Date('03-06-1930',format='%d-%m-%Y')
Sys.Date(); # today: date only
days.apart <- d-x; weekdays(d); months(d)
dp <- as.POSIXlt(d); dp$year <- dp$year - 1
d <- as.Date(dp); names(unclass(dp))
# Time: fraction of a date in a double
Sys.time(); date() # today's date and time
# Really useful: zoo and lubridate packages
```

I/O and the file system

```
cat(i); print(j) # cat: tighter, no newline
getwd(); setwd('~/Desktop'); list.files()
list.dirs(); dir(); Sys.glob() # wild card
save(z, file='z.bin'); load('z.bin')
unlink('z.bin'); con <- file('f.txt', 'rt')
y <- readLines(con, 1) # read lines of text
# writeLines(text, con=c, sep="\n") # etc.
write.csv(df, file='fileName.csv')
```

Script and package management

```
source('program.R') # incorporate R code
install.packages('ggplot2') # install pack
library('ggplot2'); require('ggplot2')
```

In the workspace

```
ls(); rm(z); help('help') # q() to quit()
help.search('help') # look for help
```

Useful debugging functions

```
# browser(); debug(); trace()
stopifnot(i[1] == 1) # assert!
warning('message'); stop('message')
```

