

R help functions		Managing R workspace		Managing R workspace (cont)		Packages	
help.s- tart()	General help	getwd()	Get working directory	save(o- bjectlist, file = "- myfile")	Save specific objects to a file	.libPaths()	Shows where library is located
help(plot) or ?plot	Help on function <code>plot</code>	setwd("- mydire- ctory")	Set working directory to <i>mydire- ctory</i>	load("m- yfile")	Load a workspace in the current session (default = .RData)	library()	Shows the packages saved in the library
help.s- earch("- geo")	Search help system for instances of string <i>geo</i>	ls()	List the objects in workspace	q()	Quit R. Will be prompted to save the workspace.	search()	Shows which packages have been loaded and ready to use
exampl- e(plot)	Examples of function <code>plot</code>	rm(obj- ectlist)	Remove one or more objects	<pre>setwd(" C:/ myp roj ect s/p roj ect 1") options() option s(d igi ts=3) x <- runif(20) summary(x) hist(x) savehi story() save.i mage() q()</pre>		install.p- ackage- s("ggpl- ot")	Install a package for the first time
RSiteS- earch("- geo")	Search for string <i>geo</i> in online help manuals and archived mailing lists	help(o- ptions)	Learn about available options			update.pa ckages- ("ggplot")	Update packages that have been installed
apropo- s("geo", mode="f- unction")	List all available functions with <i>geo</i> in their name	options()	View or set current options			instal- led.pa- ckages()	Lists packages with version numbers, dependencies, and other info
data()	List all example datasets in loaded packages	history(#)	Display the last # of commands (defau- lt=25)			library(g- gplot)	Load package
vignette()	List all vignettes for installed packages	savehi- story("- myfile")	Save the commands history to <i>myfile</i> (default = .Rhistory)	Saving graphic output		help(p- ackage- ="ggplot")	Brief description of package and index of functions and datasets
vignet- te("- geo")	Display specific vignettes for topic <i>geo</i>	loadhi- story("- myfile")	Reload a commands history (default = .Rhistory)	pdf("filename.pd- f")	PDF file	<pre>help.s tart() instal l.p ack age s("v - cd") help(p ack age ="vc d") librar y(vcd) help(A rth ritis) Arthritis exampl e(A rth ritis) q()</pre>	
		save.i- mag- e("myfi- le")	Save the workspace to myfile (default = .RData)	win.metafile("filen- ame.wmf")	Windows metafile		
				png("filename.pn- g")	PBG file		
				jpeg("filename.j- pg")	JPEG file		
				bmp("filen- ame.bmp")	BMP file		
				postscript("file- name.ps")	PostScript file		
				<pre>jpeg("r plo t.j peg ") plot(x,y) dev.off()</pre>			



Data input	Accessing DBMSs - RODBC package (cont)	Data structures (cont)	Example code (cont)
Keyboard input	<pre>mydata <- data.frame(age=numeric(0), gender=character(0)) sqlQuery(channel, query)</pre>	<pre>Arrays = character(0), vector = numeric(0), logical(0), complex(0)</pre> Similar to matrices but can have more than 2 dimensions	<pre>> patientID <- c(1,2,3,4) age <- c(25,34,28,52) diabetes <- c("Type1","Type2","Type2","Type1") dimnames <- list(c("PatientID", "age", "diabetes"), c("PatientID", "age", "diabetes")) status <- c("Poor", "Improved", "Excellent", "Poor")</pre>
Delimited text file	<pre>mydata frame <- read.table("file", header=TRUE, as.is=TRUE, sep=" ", act ors =logical_value)</pre>	Data frames Different columns can contain different modes of data	<pre>patientdata <- data.frame(patientID=patientID, age=age, diabetes=diabetes, status=status) # cross tabulate diabetes by status table(patientdata\$diabetes, patientdata\$status)</pre>
Excel file	<pre>library(R ODBC) channel <- odbcConnection("mychannel") mydata frame <- sqlFetch(channel, "mytable") odbcClose(channel) or library(xlsx) workbook <- "c:/myworkbook.xlsx" mydata frame <- read.xlsx(workbook,1)</pre>	<pre>mydata <- data.frame(coll1=1:10, coll2=1:10, coll3=1:10) update(patientdata, coll1="diabetes", status="Poor")</pre>	<pre>patientdata <- data.frame(patientID=patientID, age=age, diabetes=diabetes, status=status) # cross tabulate diabetes by status table(patientdata\$diabetes, patientdata\$status)</pre>
Others	XML, Webscrapping, SPSS, SAS, Stata, netCDF, HDF5, DBMSs	Factors Nominal, ordinal or continuous	<pre>status <- factor(status, levels=c("Poor", "Improved", "Excellent")) patientdata\$diabetes <- factor(patientdata\$diabetes, levels=c("Type1", "Type2")) # Remove a table from the ODBC database j <- matrix(1:10, nrow=5) k <- c("one", "two", "three") mylist <- list(title=g, ages=h, j, k) mylist[[2]]</pre>
Accessing DBMSs - RODBC package	<pre>sqlDrop(channel, sqtable) odbcConnection(dsn, uid=" ", pwd=" ") close(channel) library(R ODBC) myconn <- odbcConnection("mydsn", uid="Rob", pwd="aa") crimedat <- sqlFetch(myconn, "crimdat") pundat <- sqlQuery(myconn, "select * from Punishmen t") close(myconn)</pre>	Lists Ordered collection of objects	<pre>mylist <- list(n ame1=object1, n ame2= object2, ..) cells <- c(1,26,24,68) rnames <- c("R 1", "R - 2") cnames <- c("C 1", "C - 2") mymatrix <- matrix (cells, nrow=2, ncol=2, byrow= TRUE, dimnames= list(r names, cnames)) dim1 <- c("A 1", "A 2") dim2 <- c("B 1", "B 2", "B 3") dim3 <- c("C 1", "C 2", "C 3", "C 4") z <- array(1:24, dim1, dim2, dim3) mymatrix <- matrix (vector, nrow=number_of_rows, ncol=number_of_col umbers, dimnames= list(dim1, dim2, dim3))</pre>
	Data structures Vectors Hold numeric, character, or logical data Matrices 2D array where each element has the same mode		

Useful tips	Useful functions (cont)	Arithmetic operators
Attach, Detach and With <code>attach (mtcars)</code> <code>summary(mpg)</code> <code>plot(mpg, wt)</code> <code>detach (mtcars)</code> <code>with(mtcars, {</code> <code>summary(mpg, disp, wt)</code> <code>plot(mpg, disp)</code> <code>plot(mpg, wt)</code> <code>})</code> <code>with(mtcars, {</code> <code>nokeep stats <- summary(mpg)</code> <code>keepstats <-<- summary(mpg)</code> <code>})</code> <code>nokeep stats # object not found</code> <code>keepstats # object exists in global env with superassignment</code> <code>-'</code>	<code>names(object)</code> <code>cbind(object, object, ...)</code> <code>rbind(object, object, ...)</code> <code>head(object)</code> <code>tail(object)</code> <code>rm(object, object, ...)</code> <code>newobject <- edit(object)</code> <code>fix(object)</code> <code>set.seed()</code>	Arithmetic operators <code>+</code> Names of Addition <code>-</code> components in Subtraction <code>*</code> an object Multiplication <code>/</code> Combines Division <code>/</code> objects as <code>^</code> columns Exponentiation <code>%%</code> Combines Modulus (x mod y) <code>x%/%y</code> Integer division Logical operators <code><</code> Lists the first part <code><=</code> Lists the last part <code><=</code> of the object Less than or <code>></code> Deletes one or more objects. Greater than <code>></code> more objects. Greater than <code>(list=ls())</code> Greater than or <code>>=</code> will remove most equal to <code>objects.</code> Exactly equal to <code>==</code> Edits object and <code>!=</code> Saves as Not equal to <code>!=newobject</code> Not x <code>x&y</code> Edits in place x or y <code>x&y</code> To have reproducible results <code>isTRUE(x)</code> Test if x is TRUE
Useful functions		
<code>length(object)</code> Number of elements/components <code>dim(object)</code> Dimensions of an object <code>str(object)</code> Structure of an object <code>class(object)</code> Class or type of an object <code>mode(object)</code> How an object is stored		

