

Data Structure

Vectors	Entries all types
Arrays	Multidimensional, all of the same type. A 2D array is a matrix.
Data frames	A list of vectors of the same length. These can be of different types. Each has a name.
Lists	Entries are completely general. Good for returning output of a function. <code>list(vec, num, char)</code>

Data Types

Numeric	<code>is.numeric(x)</code> to check if x is numeric
Character	<code>character(x)</code> to check if x is character
Logical	<code>is.logical(x)</code> to check if x is logical
Factor	<code>is.factor(x)</code> to check if x is a factor. Factors are numeric. <code>factor(x)</code> coerce number x into factor.

Creating Vectors

```
c(1, 2, 3)
1:7
seq(from=1, to=10, by=.5)
rep(1:5, each=3, time=2)
scan("filename")
```

Extracting Elements from Vectors

```
x[c(2, 17, 4)]           By index
x[-c(2, 17, 4)]         By excluding some indices
x[x<3] or x[y=="female"] By logical statement
```

Vector Indices

```
which.max(x), which.min(x) - Extract index/indices of max, min, <
in(x), which(x<3)          3 values in vector x
order(x)                   Sort vector x
```

Read File

Function

<code>sqr <- function(x) { return (x*x) }</code>	<code>sqr()</code> to call function
<code>if(x>3){return(x)}</code>	if function
<code>invisible()</code>	Does the same as <code>return()</code> but does not print output to screen
<code>cat()</code>	Does the same as <code>print()</code> but is valid only for atomic types (logical, integer, real, complex, character) and names
<code>system.time()</code>	Output time taken to run a function. Output user, system, elapsed time.

List

```
list$sdev           Extract element by name
list["sdev"]        Extract element by name
list[[1]]           Extract element by index
```

Matrix

```
scan(file="n.txt", what="character", quote=" ")
```

```
read.csv(file="name.csv")
```

```
readLines(file="name.txt")
```

```
matrix(1:8, nrow=4)  
name,  
what  
= the
```

```
cbind(1:4, 5:8)  
of  
data
```

```
rownames(x) <- letters[1:4]  
to be  
read,
```

```
colnames(x) <- letters[1:4]  
read  
csv
```

```
file *  
read *%  
txt file
```

```
solve(x)  
line
```

```
as.matrix(dataframe)  
by
```

```
line  
apply(x, 2, mean)
```

```
x[1,2]  
x[,2]  
x[,-2]
```

```
x[1,2] Extract element on row 1, col 2 of matrix x
```

```
x[,2] Extract elements on col 2
```

```
x[,-2] Extract elements not on col 2
```

Regular Expression

```
grep("regexpr", vector)  
or)  
Return the indices of a vector that match a set of characters (or a pattern)
```

C

By **felyne223**

cheatography.com/felyne223/

Not published yet.

Last updated 13th April, 2022.

Page 1 of 5.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Regular Expression (cont)	Regular Expression (cont)	Regular Expression (cont)
<code>grepl(" reg exp r", vector)</code>	Return TRUE or FALSE for each element of a vector on the basis of whether it matches a set of characters	Matches at most optional string
<code>regexpr("r ege xpr ", vector)</code>	Tells you which elements match, where they match, and how long each match is. Matches the first occurrence of pattern in an element.	Matches at least
<code>gregexpr(" reg exp r", vector)</code>	Same as regexpr. Matches every occurrence of pattern in an element.	Matches at least
<code>gsub("r ege xpr ", vector)</code>	String subs	match from a to
<code>Cur.n</code>	Single wild card character e.g. <code>Cur.n</code> matches "Curran", "Curren" and "Currin"	ences of the pre
<code>Cur(a e i)n</code>	Alternation. Matches "Curran", "Curren" and "Currin"	match a or more
metacharacter	If a character is a regex metacharacter then it has a special meaning to the RegEx interpreter. Metacharacter [], [], \, ?, *, +, {, }, , \$, \<, \>, and (). Escape done by preceding it with a double backslash \.	of the previous
<code>[a-9]</code>	Will match any digit from 0 to 9	Looks for a patt
<code>[a-z]</code>	Will match any lower case letter from a to z	matches C or K
<code>[A-Z0-9]</code>	Will match uppercase letter from A to Z or any digit from 0 to 9	a, r appears 1 to
<code>[:alpha:]</code>	Alphabetic (only letters)	matches i or e f
<code>[:lower:]</code>	Lowercase letters	more occuranc
<code>[:upper:]</code>	Uppercase letters	If a character is
<code>[:digit:]</code>	Digits	metacharacter t
<code>[:alnum:]</code>	Alphanumeric (letters and digits)	special meaning
<code>[:space:]</code>	White space	RegEx interpre
<code>[:punct:]</code>	Punctuation	?, *, +, {, }, ^, \$,
		Escape done by
		with a double b
		Use round brac
		to capture the n
		interest. Use \1
		backreference c
		retrieve the info
		matched.
		Example use of
		brackets in rege
		extracts inform
		round bracket, \
		information in s
		bracket.
		Extract substrin
	<code>substr(string, start, stop)</code>	'a bcdef', \
		bcd.
		paste element
	<code>paste(x, y, sep = ' ', collapse = ' ')</code>	(more are allow
		separator betwe
		ponding sub-ele
		and y. Collapse
		between x and :



Regular Expression (cont)

`strsplit(vector of strings, sep=' ')` Separate strings in vector based on separator set in `sep`

Regular expression provide a way of matching patterns in text.

R plot

`par(mfrow=c(3,3))` Set the plotting area to 3 * 3 array

`apply(matrix, 2, hist, xlim=c(-4, 4))` for each column in matrix, plot histogram, x axis limit is -4 to 4

`rnorm(n, mean=1, sd=1)` random number generation following normal distribution

`lm(y~x, data=data)` linear regression

`abline(lm(y~x))` plot linear regression

`plot(x, y)` plot points

`main, xlim,` variables to be included in graphical functions. Title, x-axis range,

R graphics (cont)

Base R vs ggplot

Base R - environment set up

Base R - type of plot

Base R - graph bits

Base R - graph parameters

`library(ggplot2)`

`p <- ggplot(df, aes(x=xvar, y=yvar)) + geom_line()`

ggplot - Scales

`ggplot facet_wrap(~var)`

`ggplot facet_grid(var1 ~var2)`

`ggplot library(plotly)`

`ggplot theme_bw()`

R graphics

Bitmap

Graphic format, pixelwise representation of your screen. If >1000 points/lines, use Bitmap format instead of Vector. Bitmap formats are bmp, png, jpg.

Vector

Graphic format, uses a set of basic plotting tools (point, line, etc) to describe a plot. Looks better, especially when you change devices/resolution. Vector formats are pdf, eps, wmf.

```
pdf(filename="my plot.pdf", width=5, height=5)
```

Saving to pdf format. Many different commands (jpeg, png, postscript) depending on the output type you want.

