# R Cheat Sheet

by felyne223 via cheatography.com/146361/cs/31646/

## 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. `list(vec, num, char)` |

## Data Types

| | |
|---|---|
| Numeric | `is.numeric(x)` to check if x is numeric |
| Character | `character(x)` to check if x is character |
| Logical | `is.logical(x)` to check if x is logical |
| Factor | `is.factor(x)` to check if x is a factor. Factors are numeric. `factor(x)` 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)`, `which( x<3)` | Extract index/indices of max, min, < 3 values in vector x |
| `order(x)` | Sort vector x |

## Read File

## Function

| | |
|---|---|
| `sqr <- function(x) { return (x*x) }` | `sqr()` to call function |
| `if(x>3 ){return(x)}` | if function |
| `invisible()` | Does the same as `return()` but does not print output to screen |
| `cat()` | Does the same as `print()` but is valid only for atomic types (logical, integer, real, complex, character) and names |
| `system.time()` | 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= " ")` | file name, what = the type of data to be read, |
| `read.csv(file="name.csv")` | read csv file |
| `readLines(file= "name.txt")` | read txt file line by line |

| | |
|---|---|
| `matrix (1:8, nrow=4)` | Creates a matrix with 4 rows and 2 columns. 1:4 in first column, 5:8 in second column. |
| `cbind(1:4, 5:8)` | Creates a same matrix, as above. |
| `rownames(x) <- letters[1:4]` | Give row names |
| `colnames(x) <- letters[1:4]` | Give column names |
| `*` | Element-wise multiplication |
| `%*%` | Matrix multiplication |
| `solve(x)` | Inverse of a matrix x |
| `as.matrix (dataframe)` | Treats a all numeric data frame as a matrix |
| `apply(x, 2, mean)` | Performs an operation for all rows or columns. Margin = 2 performs operation on column, 1 on row. |
| `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)` | Return the indices of a vector that match a set of characters (or a pattern) |

## Regular Expression (cont)

| | |
|---|---|
| `grepl( " reg exp r", vector )` | Return TRUE or FASE for each element of a vector on the basis of whether it matches a set of characters |
| `regexp r("r ege xpr ", vect or)` | Tells you which elements match, where they match, and how long each match is. Matches the first occurrence of pattern in an element. |
| `gregex pr( " reg exp r", ve ctor)` | Same as regexpr. Matches every occurrence of pattern in an element. |
| `gsub("r ege xpr ",   vector )` | String subs |
| `Curr.n` | Single wild card character e.g. `Curr.n` matches "Curran", "Curren" and "Currin" |
| `Curr(a \|e\|i)n` | Alternation. Matches "Curran", "Curren" and "Currin" |
| metacaracter | If a character is a regex metacharacter then it has a special meaning to the RegExp interpreter. `[ ], [], \, ?, *. +, {,}, , $, \<, \>, \| and ().` Escape done by preceding it with a double back slash `\`. |
| `[a-9]` | Will match any digit from 0 to 9 |
| `[a-z]` | Will match any lower case letter from a to z |
| `[A-Z0-9]` | Will match uppercase letter from A to Z or any digit from 0 to 9 |
| `[:alpha:]` | Alphabetic (only letters) |
| `[:lower:]` | Lowercase letters |
| `[:upper:]` | Uppercase letters |
| `[:digit:]` | Digits |
| `[:alnum:]` | Alphanumeric (letters and digits) |
| `[:space:]` | White space |
| `[:punct:]` | Punctuation |

## Regular Expression (cont)

| | |
|---|---|
| `?` | Matches at mos optional string |
| `*` | Matches at leas |
| `+` | Matches at leas |
| `{a,b}` | match from a to ences of the pre |
| `{a,}` | match a or more of the previous |
| `[CK] (u\|a)r {1, 2}( i\|e)*n` | Looks for a patt matches C or K a, r appears 1 t matches i or e f more occurrenc |
| Metacaracter | If a character is metacharacter t special meaning RegExp interpre ?, *. +, {,}, ^, $, Escape done by with a double ba |
| Back Substitution | Use round brac to capture the n interest. Use \\1 backreference c retrieve the info matched. |
| `(^[0-9 ][.])[ ]+([A- Za- z]+$)` | Example use of brackets in rege extracts informa round bracket, \ information in s bracket. |
| `substr (st ring, start, stop)` | Extract substrin `'a bcdef', 2` `bcd`. |
| `paste(x, y, sep = ' ', collapse =' ' )` | `paste` element (more are allow separator betwe ponding sub-ele and y. Collapse between x and y |

### 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

### R graphics (cont)

| | |
|---|---|
| Base R vs ggplot | Ba gr gg tou un |
| Base R - environment set up | `pa 5)` |
| Base R - type of plot | sc bo |
| Base R - graph bits | po ab |
| Base R - graph parameters | xli lty |
| `library(ggplot2)` | im |
| `p <- ggplot(df, aes(x= xvar, y=yvar ))+ geom_line()` | Ae plo to |
| ggplot - Scales | Us ax lim ma `ol` |
| ggplot `facet_wrap(~var)` | pu dif `(~` |
| ggplot `facet_grid(var1 ~var2)` | go fac |
| ggplot `library(patchwork)` | Co Or p2 |
| ggplot `theme_bw()` | Mo plo ba |

| | |
|---|---|
| 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 foramts are pdf, eps, wmf. |
| `pdf(filename="myplot.pdf", width=5, height=5)` | Saving to pdf format. Many different commands (jpeg, png, postscript) depending on the output type you want. |