R Cheat Sheet: tRips and tRaps for new players

General Vectors and vector recycling

Trap: R error messages are not helpful Trap: most objects in R are vectors. R does
Tip: use traceback() to understand errors not have scalars (just length=1 vectors).
Many Fns work on entire vectors at once.
Object coercion Tip: In R, for-loops are often the
Trap: R objects are often silently coerced inefficient and inelegant solution. Take
to another class/type as/when needed. the time to learn the various “apply”
Examples: c(1, TRUE) # -> 1 1 family of functions. Hadley Wickham's plyr
c(1, TRUE, 'cat') # -> "1" "TRUE" "cat" package is also worth learning and using.
30 < '8' # yields TRUE; 30 became "30" Trap: Math with different length vectors
Tip: inspect objects with str(x) mode(x) will work with the shorter vector recycled

```
class(x) typeof(x) dput(x) or attributes(x) Eg: c(1, 2, 3) + c(10, 20) # -> 11, 22, 13
Trap: is.vector(list(1, 2, 3)) # -> TRUE
```

Factors (special case of coercion)

Trap: Factors cause more bug-hunting grief Vectors need the c() operator
than just about anything else in R Wrong: mean(1, 2, 3, 4, 5, 6) # -> 1
(especially when string and integer vectors Correct: mean(c(1, 2, 3, 4, 5, 6)) # -> 3.5
and data.frame cols are coerced to factors)
Tip: Learn about factors and using them. Use the correct Boolean operator
Tip: explicitly test with is.factor(df$col) Tip: | and & are vectorised - use ifelse()
Tip: use stringsAsFactors=FALSE argument (| and & also used with indexes to subset)
when you create a data frame from file Tip: || and && are not vectorised - use if
Trap: maths doesn’t work on numeric factors Trap: || && lazy evaluation; | & full eval
and they are tricky to convert back. Trap: == (Boolean equality) = (assignment)

```
Tip: try as.numeric(as.character(factor))
```

Trap: appending rows to a data frame with Equality testing with numbers
factor columns is tricky. Tip: make sure Trap: == and != test for near in/equality
the row to be appended is a presented to Eg:

```
as.double(8) == as.integer(8) is TRUE
```

rbind() as a data.frame, and not as a isTRUE(all.equal(x, y)) tests near equality
vector or a list (which works sometimes)) Tip:
identical(x, y) is more fussy
Trap: the combine function c() will let you
combine different factors into a vector of Think hard about NA, NaN and NULL
integer codes (probably garbage). Trap:
NA and NaN are valid values.
Tip: convert factors to strings or integers Eg

```
: c(1, 2) == c(1, NA) # -> TRUE, NA
```

(as appropriate) before combining. Trap: many Fns fail by default on
NA input
Tip: many functions take: na.rm=TRUE
Garbage in the workspace Tip: vector test for NA:

```
any(is.na(y))
```

Trap: R saves your workspace at the end of
Trap: x == NA is not the same as is.na(x)
each session and reloads the saved
Trap: x == NULL not the same as is.null(x)
workspace at the start of the next session.

```
Trap: is.numeric(NaN) returns TRUE
```

Before you know it, you can have heaps of
variables lurking in your workspace that

Indexing ([], [[]], $)

are impacting on your calculations.
Tip: Objects are indexed from 1 to N.
Tip: use ls() to check on lurking variables
Trap: many subtle differences in indexing

```
Tip: clean up with rm(list = ls(all=TRUE))
```

for vectors, lists, matrices, arrays and
Tip: library() to check on loaded packages
data.frames. Return types vary depending on
Tip: avoid saving workspaces, start R with
object being indexed and indexation method.

```
the --no-save --no-restore arguments
```

Tip: take the time to learn the differences
Trap: the zero-index fails silently
The 1:0 sequence in for-loops Eg:

```
c(1, 2, 3)[c(0,1,2,0,2,3)] # -> 1,2,2,3
Trap: for(x in 1:length(y)) fails on the
```

Trap: negative indexes return all but those
zero length vector. It will loop twice: Eg:

```
c(1, 2, 3, 4)[-c(1, 3)]] # -> 2, 4
```

first setting x to 1, then to 0.
Trap: NA is a valid Boolean index

```
Tip: use for(x in seq_len(y))
Eg: c(1, 2)[c(TRUE, NA)] # -> 1, NA
not for(x in 1:length(y))
```

Trap: mismatched Boolean indexes work

```
Tip: for(x in seq_along(y)) not for(x in y)
Eg: c(1, 2, 3)[c(T,F,T,F,T)] # -> 1, 3, NA
```