Regular Expressions in Stata
Cheat Sheet

Works with regular expressions (i.e. using operators like ^, $, * etc.)?

Detect pattern

✓ `gen new_var = regexm(variable, "pattern")`

Returns a number depending on whether the pattern was matched (1) or not (0).

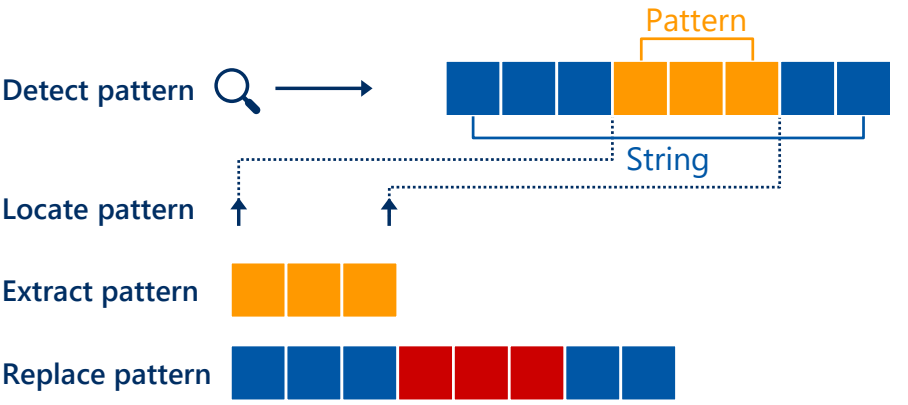
Locate pattern

✗ `gen new_var = strpos(variable, "pattern")`

Returns the starting position (4 in the e.g.) of the pattern inside the string.

✗ `gen new_var = strpos(variable, "pattern") + length("pattern")`

Returns the final position (6 in the e.g.) of the pattern inside the string.



Extract pattern

✓
```
gen new_var = ""
replace new_var = regexs(0) if regexm(variable, "pattern") == 1
```
Extracts first pattern matched by regexm.

✓
```
gen new_var = ""
replace new_var = rexegs(1) + "-" + regexs(2) if regexm(variable, "(pat)(tern)") == 1
```
Extracts first pattern match and splits it into the sub-patterns created by the brackets in regexm.

✗ `gen new_var = substr(variable, 4, 3)`

Extracts a pattern of length 3, starting from the 4th character (requires all observations to be in the same format and for the desired string to be the same length and in the same position for all observations).

Replace pattern

✓ `gen new_var = regexr(variable, "pattern", "new_pattern")`

Replaces first pattern match. Can replace all pattern matches using `ustrregexra()`.

✗ `gen new_var = subinstr(variable, "pattern", "new_pattern", .)`

Replaces all pattern matches.`

Note: these string functions only work for ASCII characters. For Unicode characters (e.g. to search for foreign accents), `ustrregexm()`, `ustrpos()`, `ustrregxs()`, `usubstr()`, `ustrregexrf()` (or `ustrregexra()` to replace all matches) and `usubinstr()` need to be used instead.

Regular Expressions in Stata
Cheat Sheet

Quantifiers		e.g.
*	Matches zero or more of the previous expression *- e.g. matches zero or more "a"s in a row*	"a*"
+	Matches one or more of the previous expression *- e.g. matches one or more "a"s in a row*	"a+"
?	Matches zero or one of the previous expression *- e.g. matches zero or one "a".*	"a?"
{}	Matches exactly this number of the previous expression *- e.g. matches two "a"s in a row, matches between two and three "a"s in a row*	"a{2}" "a{2,3}" Only works with unicode functions

Groups		e.g.
|	Matches this or that character(s) / expression *- e.g. matches "a" or "b" (can use longer strings on either side of the operator, making it different to the next line)*	"a|b"
[]	Matches the specified set of characters / expressions *- e.g. matches "a" or "b"*	"[ab]"
[^]	Matches anything not in the specified set of characters / expressions *- e.g. matches anything that isn't "a" or "b"*	"[^ab]"
()	A sub-expression (for extraction using `regexs()`) *- e.g. matches "ab"*	"(ab)"

Anchors		e.g.
^	Starts with subsequent expression *- e.g. starts with "a"*	"^a"
$	Ends with previous expression *- e.g. ends with "a"*	"a$"
\b	Matches subsequent / previous expression at the start / end of a word *- e.g. matches "a" if it is at the start of a word, matches "a" if it is at the end of a word* Only works with unicode functions	"\ba" "a\b"

Characters		e.g.
-	Matches a particular range of characters or numbers *- e.g. matches any character between "a" and "z" (can specify a different range like e-j), matches any digit between 0 and 9, matches any alphanumeric character*	"[a-z]" "[0-9]" "[a-zA-Z0-9]"
.	Matches any character	"."
\	Matches subsequent character that would otherwise be recognised as a regular expression operator *- e.g. matches a period*	"\."
[:punct:]	Matches one of the following characters: ! " # % & ' ( ) * , - . / : ; ? @ [\ ] ^ _ { } *- e.g. matches any punctuation mark (this does not match other special characters, which can be matched using (add to the set as needed): "[[:punct:]£$+<=>`|~]")*	"[:punct:]" Only works with unicode functions