

Make Data Analyzable Before Data Analysis:
From Messy Strings to Use-Ready Variables

Wen Rixin PhD Claremont Graduate University

Stata Conference Boston
Oct 1 2026

When copied and pasted from PDF and other text sources, Word or Twitter, into Stata, data becomes variable in string format. It is not meaningless or useless, but not ready for immediate analysis. Careful looks at these messy strings often can reveal their hidden structures, enabling us to convert them into use-ready variables.

All examples here are based on Stata Version 14.

Simpler extraction

Extracting the numerical sector: `gen var2=regexr(var1,"[\.\\}\)*a-zA-Z]+", "")`. The shortcoming with this command is it may appear intimidating to the beginner; thus, I try a simpler one: `gen var2=regexr(var1, "^[0-9]+", "")`.

var1	var2
NYSE.12	12
AFN.345	345
31231)	31231
23146}	23146
234212*	234212
34234N	34234
35457S	35457
3567754G	3567754
2312A	2312
2144F	2144
123A33	12333

For the variable below, to extract the string from the mixture: `gen var2=regexr(var1,"[.0-9]+", "")`. It works. So does “regexs”: `gen var=regexs(0) if regexm(var1, "[a-zA-Z]+")`. Yet, neither is better than this: `split var1, parse(.)`!

var1	var2
ACGETYF.1235	ACGETYF
CDFGEEGY.596544	CDFGEEGY
ACDET.1234564	ACDET
ADGT.2345	ADGT
AFM.123	AFM

With common string functions

Extracting zip, `gen zip = regexs(0) if(regexm(address, "[0-9][0-9][0-9][0-9][0-9]$"))`. Since zip code is the last word of string variable, `gen zip=word(address,-1)` to get the same result in an easy way.

address	zip
4905 Lakeway Drive, College Station, Texas 77845	77845
673 Jasmine Street, Los Angeles, CA 90024	90024
2376 First street, San Diego, CA 90126	90126
66666 West Central St, Tempe AZ 80068	80068
12345 Main St. Cambridge, MA 01238	01238

Previous solution does not work for the data-set below since the last word can be “names” or digits, `gen zip = regexs(1) if(regexm(address, "([0-9][0-9][0-9][0-9][0-9])[-]*[0-9]*[ a-zA-Z]*$"))` is surely a viable solution; however, the command looks complicated.

This case has a simpler solution: `gen zip=cond(word(address,-1)=="USA", word(address,-2), word(address,-1))`; `replace zip=substr(zip, 1,5)`. If country name includes variant like United States, United States of America, US or USA, `gen zip=regexr(address, "[a-zA-Z ]+$", "")`, then `replace zip=substr(word(zip,-1),1,5)`.

address	zip
4905 Lakeway Drive, College Station, Texas 77845 USA	77845
673 Jasmine Street, Los Angeles, CA 90024 USA	90024
2376 First street, San Diego, CA 90126	90126
66666 West Central St, Tempe AZ 80068	80068
12345 Main St. Cambridge, MA 01238-1234	01238
12345 Main St Sommerville MA 01239-2345	01239
12345 Main St Watertwon MA 01239 USA	01239

Extracting the first and last name and arranging them in reverse order : `gen new_name = regexs(2)+", "+regexs(1) if(regexm(fullname, "(\\[a-zA-Z]+)\\[ ]*(\\[a-zA-Z]+)"))`. Solution can be more straightforward and simpler: `gen name=word(fullname,-1)+", "+word(fullname,1)`. If some full names consist of three sectors, we need to set a condition, `gen name=cond(word(fullname,2)==word(fullname,-1), word(fullname,-1)+", "+word(fullname,1), word(fullname,-1)+ " "+word(fullname,1)+ " "+word(fullname,2))`

fullname	name
John Adams	Adams,John
Adam Smiths	Smiths,Adam
Mary Smiths	Smiths,Mary
Charlie Wade	Wade,Charlie

“Strlen” contains more information

As a mess of careless or mistaken concatenated duplication, keep half-length to get (rid of) what we (don’t) want: replace `var=substr(var, 1, strlen(var)/2)`

var1	var2
USAUSA	USA
9002490024	90024
9012690126	90126
8006880068	80068
01238-123401238-1234	01238-1234
01239-234501239-2345	01239-2345
ABe23ABe23	ABe23

To add leading zero to string variable x, gen `y=cond(strlen(x)==2, "0"+x,x)`. Strlen not only tells length but also sets condition.

x	y
55	055
23	023
81	081
144	144
174	174
92	092
84	084
178	178
11	011
135	135

Regular expression for string time variable

To get the date, `gen day=regexs(0) if regexm(date, "[0-9]+")`; to extract month, `gen month=regexs(0) if regexm(date, "[a-zA-Z]+")` or `gen month=regexs(0) if regexm(date, "[^0-9]+")`; to get year, `gen year=regexs(0) if regexm(date, "[0-9]+$")` or `gen year=regexpr(date, day+month, "", .)`. In this case, we can replace both day and month simultaneously; but pay attention to the conditionality: right order and recognizable combination. To replace month+day or day+" "+month gives incorrect result!

“Date” settles this problem “gracefully”!

date	day	month	year
20jan2007	20	jan	2007
16June06	16	June	06
06sept1985	06	sept	1985
21june04	21	june	04
4july90	4	july	90
9jan1999	9	jan	1999
6aug99	6	aug	99
19august2003	19	august	2003

Useful “strpos” and “strrpos”

Another useful tool for effective and efficient extraction is "strpos", sometimes "strrpos". For the infant mortality rate (IMR) for 50 states 2022 from CDC website:

```
United States ..... 5.61 20,577 3,667,758
Alabama ..... 6.69 389 58,149
Alaska ..... 6.62 62 9,359
Arizona ..... 6.17 485 78,547
Arkansas. .... 7.67 272 35,471
California ..... 4.11 1,724 419,104
Colorado ..... 4.54 283 62,38
```

To get state, `gen state=substr(var1,1, strpos(var1,".")-1)`; to get the imr, `gen imr=regexs(0) if regexm(var1, "[0-9]+")`, then `replace imr=substr(var1, strpos(var1, imr), .)`.

Sometimes, "strrpos" can help with data cleaning. For this infant birthweight data:

birthweight (grams)

```
Less than 2,500 ..... 316,381 2,270 20,619 75,686 864 131,368 73,996 15,120 1,998 2,994 39,129 6,881

Less than 1,500 ..... 50,760 389 2,545 15,061 151 18,850 11,818 2,272 370 545 6,195 1,180

1,500–2,499 ..... 265,621 1,881 18,074 60,625 713 112,518 62,178 12,848 1,628 2,449 32,934 5,701

2,500 or more ..... 3,350,350 23,447 198,357 435,642 9,257 1,709,108 863,368 192,707 24,388 30,534 471,042
62,378

2,500–4,499 ..... 3,317,042 23,047 197,470 432,932 9,116 1,689,020 855,451 191,128 24,137 30,297 466,357 61,872

4,500 or more ..... 33,308 400 887 2,710 141 20,088 7,917 1,579 251 237 4,685 506

Not stated ..... 1,027 4 18 111 1 263 57 16 4 3 23 7
```

To get the value part, we can use "strrpos" to achieve our goal:

gen weight=substr(birthweight, strrpos(birthweight,"")+1,.). But keep in mind this is a particular scenario because there is no "point" in the numeric part! Strpos usually works well when the pattern marks the beginning of a separator; by contrast, strrpos may not be reliable when the separator itself is repetitive and generates multiple overlapping matches.

Suppose we have such dataset:

var1

```
Less than 2.500 ..... 316.381 2.270 20.619 75.686 864 131.368 73.996 15.120 1.998 2.994 39.129 6.881

Less than 1.500 ..... 50.760 389 2.545 15.061 151 18.850 11.818 2.272 370 545 6.195 1.180

1.500–2.499 ..... 265.621 1.881 18.074 60.625 713 112.518 62.178 12.848 1.628 2.449 32.934 5.701

2.500 or more ..... 3.350.350 23.447 198.357 435.642 9.257 1.709.108 863.368 192.707 24.388 30.534 471.042
62.378

2.500–4.499 ..... 3.317.042 23.047 197.470 432.932 9.116 1.689.020 855.451 191.128 24.137 30.297 466.357 61.872

4.500 or more ..... 33.308 400 887 2.710 141 20.088 7.917 1.579 251 237 4.685 506

Not stated ..... 1.027 4 18 111 1 263 57 16 4 3 23 7
```

When separator differentiating category from values is long-running dots of varying length, we can use a string of dots to subtract the first parts: gen id=substr(var1, 1, strrpos(var1, ".")-1); to obtain the numeric values, gen value=regexr(var1, id, ""), then replace values=regexr(value, "[.]+ ", "").

id	value
Less than 2.500	316.381 2.270 20.619 75.686 864 131.368 73.996 15.120 1.998 2.994 39.129 6.881
Less than 1.500	50.760 389 2.545 15.061 151 18.850 11.818 2.272 370 545 6.195 1.180
1.500–2.499	265.621 1.881 18.074 60.625 713 112.518 62.178 12.848 1.628 2.449 32.934 5.701
2.500 or more	3.350.350 23.447 198.357 435.642 9.257 1.709.108 863.368 192.707 24.388 30.534 471.042 62.378

2.500–4.499	3.317.042 23.047 197.470 432.932 9.116 1.689.020 855.451 191.128 24.137 30.297 466.357 61.872
4.500 or more	33.308 400 887 2.710 141 20.088 7.917 1.579 251 237 4.685 506
Not stated	1.027 4 18 111 1 263 57 16 4 3 23 7

Used together, “substr’ with “strpos” are useful with data cleaning, nevertheless it can hide subtle error out of miscomprehension or confusion: please keep clear in mind that subtraction begins with position while ends with length.

More than one solution

This data is about annual GDP by state 2023-2025¹

GeoFIPS GeoName 2023 2024 2025

00000 United States	2.9 2.8 2.1
01000 Alabama	3.5 2.9 2.0
02000 Alaska	7.3 3.3 2.8
04000 Arizona	3.3 4.5 2.0
05000 Arkansas	2.4 3.8 2.2
06000 California	1.8 3.2 2.5
08000 Colorado	4.2 2.0 2.1
09000 Connecticut	1.4 1.0 2.4
10000 Delaware	1.2 4.0 2.3
11000 District of Columbia	0.1 1.3 0.4
12000 Florida	5.1 3.3 3.1
13000 Georgia	1.4 2.6 1.9
15000 Hawaii	2.5 2.5 2.5
16000 Idaho	2.7 4.5 2.0
17000 Illinois	1.4 1.6 1.6
18000 Indiana	1.8 2.9 2.5
19000 Iowa	-0.6 2.5 1.2
20000 Kansas	1.4 0.7 2.0
21000 Kentucky	2.9 1.8 1.0
22000 Louisiana	3.7 3.3 1.1
23000 Maine	3.6 3.0 0.6

¹ As of Oct 3, 2026, the original information has been superseded.

24000 Maryland 1.9 3.0 0.7

25000 Massachusetts 1.2 2.3 2.4

26000 Michigan 2.7 1.4 1.2

27000 Minnesota 1.0 2.4 1.6

28000 Mississippi 3.2 3.3 1.6

29000 Missouri 2.9 1.8 1.3

30000 Montana 5.0 3.4 2.2

31000 Nebraska 5.2 0.6 1.4

To get the code, `gen code=substr(var1,1,5)`; `gen state = regexs(0) if regexm(var1, "[a-zA-Z]+")`; to get GDP, `gen GDP=regexr(var1, code, "")`, `replace GDP=regexr(GDP, state, "")`. Though sometimes “subinstr” or “subinword” can serve as an alternative, “regexr” is more helpful because it functions on criteria as well as specific content; moreover, removal is much safer step by step.

There is another way to achieve the same goal: code is the word No1 in var1, `gen code=word(var1,1)`; the last three words are GDP, `gen GDP=word(var1,-3)+" "+word(var1,-2)+" "+word(var1,-1)`; remove both code and GDP, we have state.

Data cleaning rarely has only one solution!

Area code extraction

This is an example of area code extraction from phone number at Stata webpage with my alterations and additions:

var1

(979) 123-1234

979-121-1231

213/7466612

{626}7979630

<858>900-2175

First, we use regular expression to remove all the non-numeric part at the beginning of the variable, `gen code=regexr(var1, "[^0-9]+","")`. Since area code has three digits, `replace code=regexs(0) if regexm(code, "[0-9][0-9][0-9]")`. We use `[0-9][0-9][0-9]` instead of `[0-9]+` for precision.

Final Thought

- i. Decomposition before construction: take a careful look at the structure of a string variable and try to uncover its hidden (regularized) patterns.
- ii. turn irregularity into regularity: use regular expressions when they help make the data less messy and then let more familiar string functions take over whenever they can.
- iii. select vs deselect: usually, there is more than one solution. When "how to keep" does not work, try "how to drop". Many data-cleaning problems can be approached from either direction though they are not always equivalent.
- iv. From underestimation to appreciation: Stata's data-management capabilities are truly strong, and its string functions may be more useful-and more powerful-than we initially realize or anticipate.

Many string variables can be cleanly separated with “split” command. So, if simplicity works, avoid complexity!

References

Danielle Ely & Anne Driscoll, “Infant Mortality in the United States, 2022: Data From the Period Linked Birth/Infant Death File,” *National Vital Statistics Reports* Vol 73, no.5(2024): 16, <https://www.cdc.gov/nchs/data/nvsr/nvsr73/nvsr73-05.pdf>

Gary Petersen, “Is there a way to put leading zeros in output?” Home/Resources&support/FAQs, Stata Corp, <https://www.stata.com/support/faqs/data-management/putting-leading-zeros-in-output/>.

Kevin Tuner, “What are regular expressions and how can I use them in Stata?” Home/Resources&support/FAQs, Stata Corp, <https://www.stata.com/support/faqs/data-management/regular-expressions/>.

Oscar Torres-Reyna, “Getting Started in Data Analysis using Stata, Princeton, Dec 2007, <https://www.princeton.edu/~otorres/StataTutorial.pdf>

“How can I extract a portion of a string variable using Regular Expression?” Stata FAQs, UCLA: Statistical Consulting Group, last accessed Oct 3, 2026, <https://stats.oarc.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-cite-web-pages-and-programs-from-the-ucla-statistical-consulting-group/>.