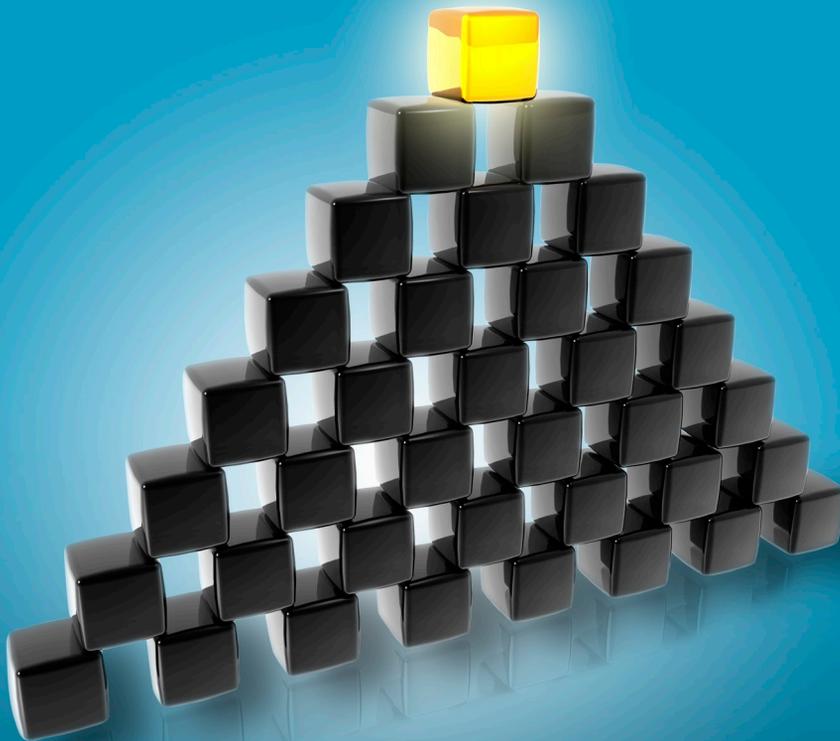
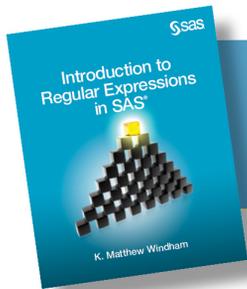


# Introduction to Regular Expressions in SAS<sup>®</sup>



K. Matthew Windham



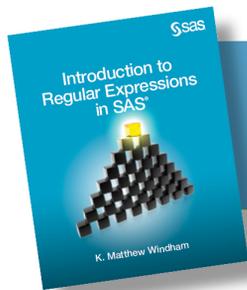
From *Introduction to Regular Expressions in SAS*<sup>®</sup>.  
Full book available for purchase [here](#).

## Contents

<b>About This Book .....</b>	<b>vii</b>
<b>About The Author .....</b>	<b>xi</b>
<b>Acknowledgments .....</b>	<b>xiii</b>
<b>Chapter 1: Introduction .....</b>	<b>1</b>
1.1 Purpose of This Book .....	1
1.2 Layout of This Book .....	1
1.3 Defining Regular Expressions.....	2
1.4 Motivational Examples .....	3
1.4.1 Extract, Transform, and Load (ETL).....	3
1.4.2 Data Manipulation .....	4
1.4.3 Data Enrichment .....	5
<b>Chapter 2: Getting Started with Regular Expressions .....</b>	<b>9</b>
2.1 Introduction .....	10
2.1.1 RegEx Test Code .....	11
2.2 Special Characters.....	13
2.3 Basic Metacharacters.....	15
2.3.1 Wildcard.....	15
2.3.2 Word.....	15
2.3.3 Non-word.....	16
2.3.4 Tab.....	16
2.3.5 Whitespace.....	17
2.3.6 Non-whitespace.....	17
2.3.7 Digit.....	17
2.3.8 Non-digit.....	18
2.3.9 Newline .....	18
2.3.10 Bell .....	19

2.3.11 Control Character .....	20
2.3.12 Octal .....	20
2.3.13 Hexadecimal .....	21
2.4 Character Classes .....	21
2.4.1 List .....	21
2.4.2 Not List .....	22
2.4.3 Range .....	22
2.5 Modifiers .....	23
2.5.1 Case Modifiers .....	23
2.5.2 Repetition Modifiers .....	25
2.6 Options .....	32
2.6.1 Ignore Case .....	32
2.6.2 Single Line .....	32
2.6.3 Multiline .....	33
2.6.4 Compile Once .....	33
2.6.5 Substitution Operator .....	34
2.7 Zero-width Metacharacters .....	34
2.7.1 Start of Line .....	35
2.7.2 End of Line .....	35
2.7.3 Word Boundary .....	35
2.7.4 Non-word Boundary .....	36
2.7.5 String Start .....	36
2.8 Summary .....	37
<b>Chapter 3: Using Regular Expressions in SAS .....</b>	<b>39</b>
3.1 Introduction .....	39
3.1.1 Capture Buffer .....	39
3.2 Built-in SAS Functions .....	40
3.2.1 PRXPARSE .....	40
3.2.2 PRXMATCH .....	42
3.2.3 PRXCHANGE .....	43
3.2.4 PRXPOSN .....	46
3.2.5 PRXPAREN .....	47

<b>3.3 Built-in SAS Call Routines</b> .....	<b>49</b>
3.3.1 CALL PRXCHANGE .....	50
3.3.2 CALL PRXPOSN .....	54
3.3.3 CALL PRXSUBSTR .....	56
3.3.4 CALL PRXNEXT .....	57
3.3.5 CALL PRXDEBUG .....	59
3.3.6 CALL PRXFREE.....	62
<b>3.4 Summary</b> .....	<b>63</b>
<b>Chapter 4: Applications of Regular Expressions in SAS</b> .....	<b>65</b>
4.1 Introduction .....	65
4.1.1 Random PII Generator .....	66
4.2 Data Cleansing and Standardization.....	72
4.3 Information Extraction .....	77
4.4 Search and Replacement .....	80
4.5 Summary .....	83
4.5.1 Start Small .....	83
4.5.2 Think Big.....	83
<b>Appendix A: Perl Version Notes</b> .....	<b>85</b>
<b>Appendix B: ASCII Code Lookup Tables</b> .....	<b>87</b>
Non-Printing Characters .....	87
Printing Characters .....	89
<b>Appendix C: POSIX Metacharacters</b> .....	<b>97</b>
<b>Index</b> .....	<b>101</b>



From *Introduction to Regular Expressions in SAS*®.  
Full book available for purchase [here](#).

## Chapter 1: Introduction

<b>1.1 Purpose of This Book</b> .....	<b>1</b>
<b>1.2 Layout of This Book</b> .....	<b>1</b>
<b>1.3 Defining Regular Expressions</b> .....	<b>2</b>
<b>1.4 Motivational Examples</b> .....	<b>3</b>
1.4.1 Extract, Transform, and Load (ETL) .....	<b>3</b>
1.4.2 Data Manipulation .....	<b>4</b>
1.4.3 Data Enrichment .....	<b>5</b>

---

### 1.1 Purpose of This Book

This book is meant for SAS programmers of virtually all skill levels. However, it is expected that you have at least a basic knowledge of the SAS language, including the DATA step, and how to use SAS PROCs.

This book provides all the tools you need to learn how to harness the power of regular expressions within the SAS programming language. The information provided lays the foundation for fairly advanced applications, which are discussed briefly as motivating examples later in this chapter. They are not presented to intimidate or overwhelm, but instead to encourage you to work through the coming pages with the anticipation of being able to rapidly implement what you are learning.

---

### 1.2 Layout of This Book

It is my goal in this book to provide immediately applicable information. Thus, each chapter is structured to walk through every step from theory to application with the following flow: Syntax ► Example. In addition to the information discussed in the coming chapters, a regular expression reference guide is included in the appendix to help with more advanced applications outside the scope of this text.

#### Chapter 1

In addition to providing a roadmap for the remainder of the book, this chapter provides motivational examples of how you can use this information in the real world.

#### Chapter 2

This chapter introduces the basic syntax and concepts for regular expressions. There is even some basic SAS code for running the examples associated with each new concept.

### Chapter 3

This chapter is designed to walk through the details of implementing regular expressions within the SAS language.

### Chapter 4

In this final chapter, we work through a series of in-depth examples—case studies if you will—in order to ‘put it all together.’ They don’t represent the limitations of what you can do by the end of this book, but instead provide some baseline thinking for what is possible.

### Appendixes

While not comprehensive, these serve as valuable, substantial references for regular expressions, SAS documentation, and reference tables. I hope everyone can leverage the additional information to enrich current and future regular expressions capabilities.

---

## 1.3 Defining Regular Expressions

Before going any further, we need to define *regular expressions*.

Taking the very formal definition might not provide the desired level of clarity:

Definition 1 (formal)

*regular expressions*: “Regular expressions consist of constants and operator symbols that denote sets of strings and operations over these sets, respectively.”<sup>1</sup>

In the pursuit of clarity, we will operate with a slightly looser definition for regular expressions. Since practical application is our primary aim, it doesn’t make sense to adhere to an overly esoteric definition. So, for our purposes we will use the following:

Definition 2 (easier to understand—our definition)

*regular expressions*: character patterns used for automated searching and matching.

When programming in SAS, regular expressions are seen as strings of letters and special characters that are recognized by certain built-in SAS functions for the purpose of searching and matching. Combined with other built-in SAS functions and procedures, you can realize tremendous capabilities, some of which we explore in the next section.

**Note:** SAS uses the same syntax for regular expressions as the Perl programming language<sup>2</sup>. Thus, throughout SAS documentation, you find regular expressions repeatedly referred to as “Perl regular expressions.” In this book, I choose the conventions present in the SAS documentation, unless the Perl conventions are the most common to programmers. To learn more about how SAS views Perl, visit this website: [http://support.sas.com/documentation/cdl/en/lefunctionsref/67239/HTML/default/viewer.htm#p0s9ila\\_gexmj18n1u7e1t1jfnzlk.htm](http://support.sas.com/documentation/cdl/en/lefunctionsref/67239/HTML/default/viewer.htm#p0s9ila_gexmj18n1u7e1t1jfnzlk.htm). To learn more about Perl programming, visit <http://perldoc.perl.org/perlre.html>. In this book, however, I primarily dispense with the references to Perl, as they can be confusing.

---

## 1.4 Motivational Examples

The information in this book is very useful for a wide array of applications. However, that will not become obvious until after you read it. So, in order to visualize how you can use this information in your work, I present some realistic examples.

As you are all probably familiar with, data is rarely provided to analysts in a form that is immediately useful. It is frequently necessary to clean, transform, and enhance source data before it can be used—especially textual data. The following examples are devoid of the coding details that are discussed later in the book, but they do demonstrate these concepts at varying levels of sophistication. The primary goal here is to simply help you to see the utility for this information, and to begin thinking about ways to leverage it.

---

### 1.4.1 Extract, Transform, and Load (ETL)

ETL is a general set of processes for extracting data from its source, modifying it to fit your end needs, and loading it into a target location that enables you to best use it (e.g., database, data store, data warehouse). We’re going to begin with a fairly basic example to get us started. Suppose we already have a SAS data set of customer addresses that contains some data quality issues. The method of recording the data is unknown to us, but visual inspection has revealed numerous occurrences of duplicative records, as in the table below. In this example, it is clearly the same individual with slightly different representations of the address and encoding for gender. But how do we fix such problems automatically for all of the records?

First Name	Last Name	DOB	Gender	Street	City	State	Zip
Robert	Smith	2/5/1967	M	123 Fourth Street	Fairfax,	VA	22030
Robert	Smith	2/5/1967	Male	123 Fourth St.	Fairfax	va	22030

Using regular expressions, we can algorithmically standardize abbreviations, remove punctuation, and do much more to ensure that each record is directly comparable. In this case, regular expressions enable us to perform more effective record keeping, which ultimately impacts downstream analysis and reporting.

We can easily leverage regular expressions to ensure that each record adheres to institutional standards. We can make each occurrence of Gender either “M/F” or “Male/Female,” make every instance of the Street variable use “Street” or “St.” in the address line, make each City variable include or exclude the comma, and abbreviate State as either all caps or all lowercase.

This example is quite simple, but it reveals the power of applying some basic data standardization techniques to data sets. By enforcing these standards across the entire data set, we are then able to properly identify duplicative references within the data set. In addition to making our analysis and reporting less error-prone, we can reduce data storage space and duplicative business activities associated with each record (for example, fewer customer catalogs will be mailed out, thus saving

money!). For a detailed example involving ETL and how to solve this common problem of data standardization, see Section 4.2 in Chapter 4.

---

## 1.4.2 Data Manipulation

Suppose you have been given the task of creating a report on all Securities and Exchange Commission (SEC) administrative proceedings for the past ten years. However, the source data is just a bunch of .xml (XML) files, like that in Figure 1.1<sup>3</sup>. To the untrained eye, this looks like a lot of gibberish; to the trained eye, it looks like a lot of work.

**Figure 1.1: Sample of 2009 SEC Administrative Proceedings XML File**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <root>
  - <administrative_proceeding>
    <url>http://www.sec.gov/litigation/admin/2009/34-61262.pdf</url>
    <release_number>34-61262</release_number>
    <release_date>Dec. 30, 2009</release_date>
    <respondents>Stephen C. Gingrich</respondents>
  </administrative_proceeding>
  - <administrative_proceeding>
    <url>http://www.sec.gov/litigation/admin/2009/34-61256.pdf</url>
    <release_number>34-61256</release_number>
    <release_date>Dec. 30, 2009</release_date>
    <respondents>Gabelli Funds LLC</respondents>
  </administrative_proceeding>
  - <administrative_proceeding>
    <url>http://www.sec.gov/litigation/admin/2009/34-61255.pdf</url>
    <release_number>34-61255</release_number>
    <release_date>Dec. 30, 2009</release_date>
    <respondents>Gabelli Funds LLC</respondents>
  </administrative_proceeding>
```

However, with the proper use of regular expressions, creating this report becomes a fairly straightforward task. Regular expressions provide a method for us to algorithmically recognize patterns in the XML file, parse the data inside each tag, and generate a data set with the correct data columns. The resulting data set would contain a row for every record, structured similarly to this data set (for files with this transactional structure):

### Example Data Set Structure

---

Release Number	Release Date	Respondents	URL
34-61262	Dec 30, 2009	Stephen C. Gingrich	<a href="http://www.sec.gov/litigation/admin/2009/34-61262.pdf">http://www.sec.gov/litigation/admin/2009/34-61262.pdf</a>
...	...	...	...

---

**Note:** Regular expressions cannot be used in isolation for this task due to the potential complexity of XML files. Sound logic and other Base SAS functions are required in order to process XML files in general. However, the point here is that regular expressions help us overcome some otherwise

significant challenges to processing the data. If you are unfamiliar with XML or other tag-based languages (e.g., HTML), further reading on the topic is recommended. Though you don't need to know them at a deep level in order to process them effectively, it will save a lot of heartache to have an appreciation for how they are structured. I use some tag-based languages as part of the advanced examples in this book because they are so prevalent in practice.

---

### 1.4.3 Data Enrichment

Data enrichment is the process of using the data that we have to collect additional details or information from other sources about our subject matter, thus enriching the value of that data. In addition to parsing and structuring text, we can leverage the power of regular expressions in SAS to enrich data.

So, suppose we are going to do some economic impact analysis of the main SAS campus—located in Cary, NC—on the surrounding communities. In order to do this properly, we need to perform statistical analysis using geospatial information.

The address information is easily acquired from [www.sas.com](http://www.sas.com). However, it is useful, if not necessary, to include additional geo-location information such as latitude and longitude for effective analysis and reporting of geospatial statistics. The process of automating this is non-trivial, containing advanced programming steps that are beyond the scope of this book. However, it is important for you to understand that the techniques described in this book lead to just such sophisticated capabilities in the future. To make these techniques more tangible, we will walk through the steps and their results.

1. Start by extracting the address information embedded in Figure 1.2, just as in the data manipulation example, with regular expressions.

**Figure 1.2: HTML Address Information**

```
<p>World Headquarters<br>
SAS Institute Inc.<br>
100 SAS Campus Drive<br>
Cary, NC 27513-2414, USA<br>
Phone:919-677-8000<br>
Fax:919-677-4444<br>
</p>
```

**Example Data Set Structure**

Location	Address Line 1	Address Line 2	City	State	Zip	Phone	Fax
World Headquarters	SAS Institute Inc.	100 SAS Campus Drive	Cary	NC	27513-2414	919-677-8000	919-677-4444

## 6 Introduction to Regular Expressions in SAS

2. Submit the address for geocoding via a web service like Google or Yahoo for free processing of the address into latitude and longitude. Type the following string into your browser to obtain the XML output, which is also sampled in Figure 1.3.

<http://maps.googleapis.com/maps/api/geocode/xml?address=100+SAS+Campus+Drive,+Cary,+NC&sensor=false>

**Figure 1.3: XML Geocoding Results**

```
- <geometry>
  - <location>
    <lat>35.8301733</lat>
    <lng>-78.7664916</lng>
  </location>
  <location_type>ROOFTOP</location_type>
  - <viewport>
    - <southwest>
      <lat>35.8288243</lat>
      <lng>-78.7678406</lng>
    </southwest>
    - <northeast>
      <lat>35.8315223</lat>
      <lng>-78.7651426</lng>
    </northeast>
  </viewport>
</geometry>
```

3. Use regular expressions to parse the returned XML files for the desired information—latitude and longitude in our case—and add them to the data set.

**Note:** We are skipping some of the details as to how our particular set of latitude and longitude points are parsed. The tools needed to perform such work are covered later in the book. This example is provided here primarily to spark your imagination about what is possible with regular expressions.

### Example Data Set Structure

Location	...	Latitude	Longitude
World Headquarters	...	35.8301733	-78.7664916

4. Verify your results by performing a reverse lookup of the latitude/longitude pair that we parsed out of the results file using <https://maps.google.com/>. As you can see in Figure 1.4, the expected result was achieved (SAS Campus Main Entrance in Cary, NC).

Figure 1.4: SAS Campus Using Google Maps



Now that we have an enriched data set that includes latitude and longitude, we can take the next steps for carrying out the economic impact analysis.

Hopefully, the preceding examples have proven motivating, and you are now ready to discover the power of regular expressions with SAS. And remember, the last example was quite advanced—some sophisticated SAS programming capabilities were needed to achieve the result end-to-end. However, the majority of the work leveraged regular expressions.

## 8 Introduction to Regular Expressions in SAS

---

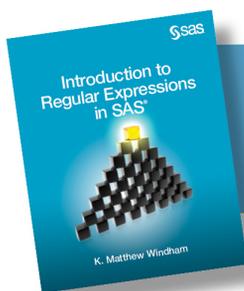
<sup>1</sup> Wikipedia, [http://en.wikipedia.org/wiki/Regular\\_expression#Formal\\_definition](http://en.wikipedia.org/wiki/Regular_expression#Formal_definition)

<sup>2</sup> For more information on the version of Perl being used, refer to the artistic license statement on the SAS support site here: [http://support.sas.com/rnd/base/datastep/perl\\_regex/regexp.compliance.html](http://support.sas.com/rnd/base/datastep/perl_regex/regexp.compliance.html)

<sup>3</sup> This example file was obtained from data.gov here:

[http://www.sec.gov/open/datasets/administrative\\_proceedings\\_2009.xml](http://www.sec.gov/open/datasets/administrative_proceedings_2009.xml)

From *Introduction to Regular Expressions in SAS®* by K. Matthew Windham. Copyright © 2014, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.



From *Introduction to Regular Expressions in SAS®*.  
Full book available for purchase [here](#).

# Index

## A

### ASCII

- about 19
- code lookup tables 87–95

## B

- backslash (\) 13
- backtracking 25–26
- bell (\a) metacharacters 19
- built-in call routines 49–63
- built-in functions 40–49

## C

- CALL PRXCHANGE 50–54, 74–75, 81
- CALL PRXDEBUG 59–62
- CALL PRXFREE 62–63
- CALL PRXNET 57–59
- CALL PRXNEXT 79
- CALL PRXPOSN 54–55
- CALL PRXSUBSTR 56–57
- CALL routine 12
- capture buffers
  - about 39–40
  - extracting data with 46–47
  - identifying 48–49
  - using 45
- case modifiers 23–25
- case sensitivity, of metacharacters 15
- character classes 21–22
- cleansing data 72–76
- CLOSE statement 61
- compile once (//o) option 33–34
- COMPRESS function 71
- context-specific algorithm development 55
- control (cA-cZ) metacharacters 20

## D

### data

- cleansing 72–76
- extracting with capture buffers 46–47
- redacting sensitive 51–52
- standardizing 44–45, 72–76
- transforming 51
- data enrichment 5–7
- data manipulation 4–5
- DATALINES statement 12
- debugging
  - information printed to log 60–62
  - PRXPARSE function 60
- digit (d) metacharacters 17–18
- DO WHILE loop 58, 79
- dot character (.) 32

## E

- ELSE tag 11
- end of line (\$) metacharacter 35
- END tag 11, 61
- escape character 14
- examples
  - data enrichment 5–7
  - data manipulation 4–5
  - Extract, Transform, and Load (ETL) 3–4
- Extract, Transform, and Load (ETL) 3–4
- extracting
  - data with capture buffers 46–47
  - information 56–57, 77–80

## F

- FILE statement 80, 81
- forward slash (/) 13
- functions, 40–49
  - See also specific functions*
- fuzzy matching 97

**G**

GOTO tag 11  
 greedy 0 or 1 time (?) modifier 26–27  
 greedy 0 or more (\*) modifier 26  
 greedy 1 or more (+) modifier 26  
 greedy n or more ({n,}) modifier 27–28  
 greedy n times ({n}) modifier 27  
 greedy n to m times ({n,m}) modifier 28  
 greedy repetition modifiers 25–26

**H**

hexadecimal (\xdd) metacharacters 21  
 hexadecimal number system 38  
 HTML 77

**I**

IF statement 11, 42, 46–47, 49  
 ignore case (/i) option 32  
 INFILE statement 78  
 information  
   debugging 60–62  
   extracting 56–57, 77–80  
 INPUT statement 12, 78  
 INT function 71

**L**

lazy 0 or 1 times (??) modifier 30  
 lazy 0 or more (\*?) modifier 28–29  
 lazy 1 or more (+?) modifier 29  
 lazy n or more ({n,}?) modifier 31  
 lazy n times ({m}?) modifier 30  
 lazy n to m times ({n,m}?) modifier 31  
 lazy repetition modifiers 28–31  
 list ([...]) metacharacter 21–22  
 lowercase (\l) metacharacter 23  
 lowercase range (\L...\E) metacharacter 24

**M**

memory, releasing with CALL PRXFREE 62–63  
 metacharacters  
   about 10–11, 15  
   bell (\a) 19

  case sensitivity of 15  
   control (\cA-\cZ) 20  
   digit (\d) 17–18  
   end of line (\$) 35  
   hexadecimal (\xdd) 21  
   list ([...]) 21–22  
   lowercase (\l) 23  
   lowercase range (\L...\E) 24  
   newline (\n) 18–19  
   non-digit (\D) 18  
   non-whitespace (\S) 17  
   non-word (\W) 16  
   non-word boundary (\B) 36  
   not list ([^...]) 22  
   octal (\ddd) 20  
   POSIX 97–99  
   quote range (\Q...\E) 25  
   range ([...-...]) 22  
   start of line (^) 35  
   string start (\A) 36–37  
   tab (\t) 16–17  
   uppercase (\u) 24  
   uppercase range (\U...\E) 24–25  
   whitespace (\s) 10–11, 17, 26  
   word (\w) 15–16  
   word boundary (\b) 35–36  
   zero-width 34–37

**modifiers**

  case 23–25  
   greedy repetition 25–26  
   lazy repetition 28–31  
   repetition 25–31  
 multiline (/m) option 33

**N**

newline (\n) metacharacter 15, 18–19  
 non-digit (\D) metacharacter 18  
 non-printing characters, ASCII codes for 87–89  
 non-whitespace (\S) metacharacters 17  
 non-word boundary (\B) metacharacter 36  
 non-word (\W) metacharacters 16  
 not list ([^...]) metacharacter 22

**O**

octal (\ddd) metacharacters 20  
 octal number system 38n1  
 OPEN statement 61  
 options 32–34  
 OUTPUT statement 79

**P**

parentheses ( ) 13  
 pattern processing 11  
 patterns  
   defining with PRXPARSE function 41  
   matching multiple times per line 58–59  
 period (.) 15  
 Perl  
   about 2  
   escape character 14  
   version notes 85  
 Personally Identifiable Information (PII) 51, 65  
 POSIX metacharacters 97–99  
 PRINT procedure 47, 53, 54  
 printing characters, ASCII codes for 89–95  
 PRX (Perl-Regular-eXpressions) 40  
 PRXCHANGE function 39–40, 43–45  
 PRXMATCH function 42–43, 74–75  
 PRXPAREN function 39–40, 47–49  
 PRXPARSE function 40–41, 60, 74–75, 78  
 PRXPOSN function 39–40, 46–47, 74–75  
 PUT statement 80

**Q**

question mark (?) 28  
 quote range (\Q...\E) metacharacter 25

**R**

RAND function 71  
 random PII generator 66–72  
 range ([...-...]) metacharacter 22  
 RANPERK function 71  
 redacting sensitive data 51–52  
 regular expressions (RegEx)  
   about 2, 10–11  
   applications of 65–84

character classes 21–22  
 metacharacters 15–21  
 modifiers 23–25  
 options 32–34  
 special characters 13–14  
 test code 11–13  
 using in SAS 39–64  
 zero-width metacharacters 34–37

repetition modifiers 25–31  
 replacement and search 80–83  
 results, inserting 52–54  
 RETAIN statement 12, 33–34, 79

**S****SAS**

built-in call routines 49–63  
 built-in functions 40–49  
 CALL PRXCHANGE 50–54  
 CALL PRXDEBUG 59–62  
 CALL PRXFREE 62–63  
 CALL PRXNET 57–59  
 CALL PRXPOSN 54–55  
 CALL PRXSUBSTR 56–57  
 capture buffer 39–40  
 PRXCHANGE function 43–45  
 PRXMATCH function 42–43  
 PRXPAREN function 47–49  
 PRXPARSE function 40–41  
 PRXPOSN function 46–47  
 using regular expressions in 39–64  
 website 5  
 search and replacement 80–83  
 SET statement 80  
 single line (/s) option 32  
 slashes (/) 34  
 source text, finding strings in with PRXMATCH  
   function 42–43  
 special characters 11, 13–14  
 SQL procedure 97  
 square brackets ([]) 21  
 standardizing data 44–45, 72–76  
 start of line (^) metacharacter 35  
 START tag 11

string start (\A) metacharacter 36–37  
strings, finding in source text with PRXMATCH  
    function 42–43  
substitution (s/) operator 34

## **T**

tab (\t) metacharacters 16–17  
test code, for regular expressions (RegEx) 11–13  
THEN tag 11  
transforming data 51

## **U**

uppercase (\u) metacharacter 24  
uppercase range (\U...\E) metacharacter 24–25

## **V**

vertical bar (|) 13

## **W**

whitespace (\s) metacharacters 10–11, 17, 26  
wildcard metacharacter 15  
word boundary (\b) metacharacter 35–36  
word (\w) metacharacters 15–16

## **X**

XML 77

## **Z**

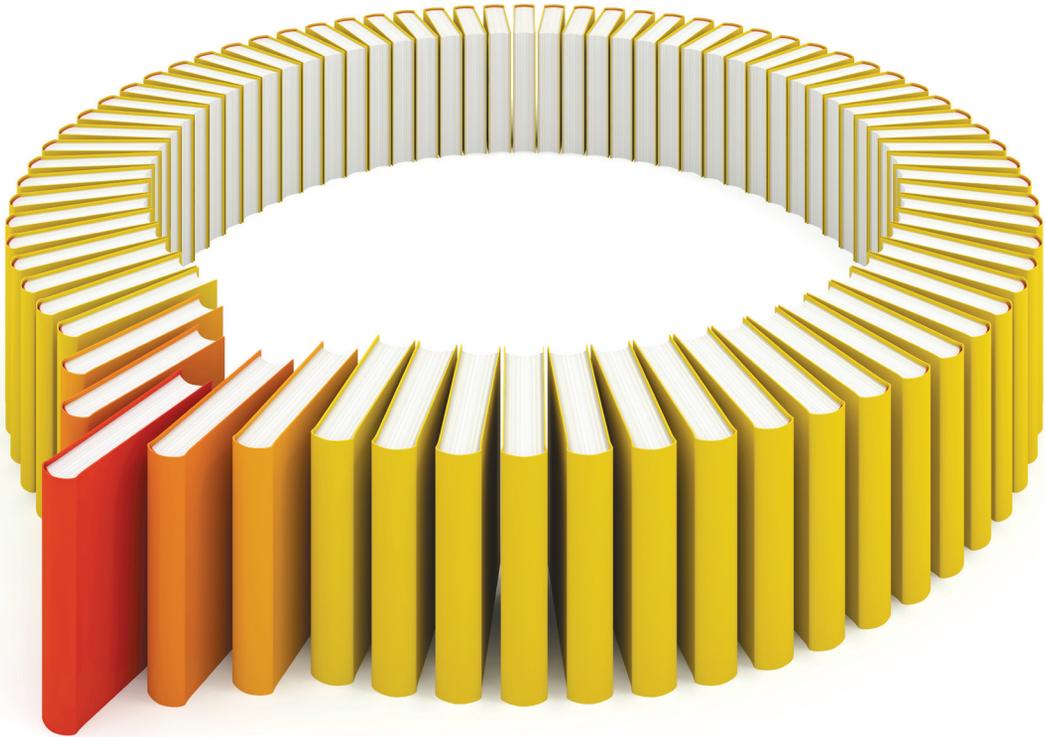
zero-width metacharacters 34–37

## About The Author



K. Matthew Windham, CAP, is the director of analytics at NTELX Inc., an analytics and technology solutions consulting firm located in the Washington, DC area. His focus is on helping clients improve their daily operations through the application of mathematical and statistical modeling, data and text mining, and optimization. A longtime SAS user, Matt enjoys leveraging the breadth of the SAS platform to create innovative, predictive analytics solutions. During his career, Matt has led consulting teams in mission-critical environments to provide rapid, high-impact results. He has also architected and delivered analytics solutions across the federal government, with a particular focus on the US Department of Defense and the US Department of the Treasury. Matt is a Certified Analytics Professional (CAP) who received his BS in Applied Mathematics from N.C. State University and his MS in Mathematics and Statistics from Georgetown University.

Learn more about this author by visiting his author page at <http://support.sas.com/publishing/authors/windham.html>. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.



# Gain Greater Insight into Your SAS<sup>®</sup> Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

 [support.sas.com/bookstore](http://support.sas.com/bookstore)  
for additional books and resources.

  
THE POWER TO KNOW.®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0413