Working with User Agent Strings in Stata: The parseuas Command

Roßmann, Joss; Gummer, Tobias; Kaczmirek, Lars

Journal of Statistical Software

February 2020, Volume 92, Code Snippet 1. *doi: 10.18637/jss.v092.c01*

Working with User Agent Strings in Stata: The parseuas Command

Joss Roßmann
GESIS – Leibniz Institute for the Social Sciences

Tobias Gummer
GESIS – Leibniz Institute for the Social Sciences

Lars Kaczmirek
University of Vienna

Abstract

With the rising popularity of web surveys and the increasing use of paradata by survey methodologists, assessing information stored in user agent strings becomes inevitable. These data contain meaningful information about the browser, operating system, and device that a survey respondent uses. This article provides an overview of user agent strings, their specific structure and history, how they can be obtained when conducting a web survey, as well as what kind of information can be extracted from the strings. Further, the user written command **parseuas** is introduced as an efficient means to gather detailed information from user agent strings. The application of **parseuas** is illustrated by an example that draws on a pooled data set consisting of 29 web surveys.

Keywords: user agent string, web surveys, device detection, browser, paradata, Stata.

1. Introduction

In recent years, web surveys have become an increasingly popular and important data collection mode, and today, they account for a great share of the studies in the social sciences. Web surveys have been used to complement traditional offline surveys as a less expensive form of pre-test or a mode of choice for respondents who are not willing to use "traditional" modes, such as face-to-face or telephone interviews.

Along with the rising popularity of web surveys, survey methodologists have taken an increasing interest in paradata, which are collected as a byproduct of the survey process (Couper 2000). Examples of paradata include the device used to complete a survey, response latencies, and response patterns. These data are used for several purposes, such as to address the strength and consistency of attitudes (Bassili 1993; Mayerl 2013; Meyer and Schoen 2014), to study data quality issues, for example, those associated with interview durations or item level

response times (Couper and Kreuter 2013; Gummer and Roßmann 2015), questions of visual layout (Stern 2008), and nonresponse adjustment (Biemer, Chen, and Wang 2013; Roßmann and Gummer 2016b; Sinibaldi, Trappmann, and Kreuter 2014).

Survey researchers can choose from many different products, referred to as web survey software (cf. Kaczmirek 2008), to conduct web surveys. Web survey software provides an interface to program the online questionnaire and assists in managing respondents, sending out emails, collecting the data, and many even generate ready-to-use reports. With respect to the concerns of this article, they can provide information about the hardware and software that was used by respondents to complete a survey. Broadly speaking, whenever a respondent uses a device to participate in a web survey, the device reveals information to the server about itself in addition to the response. This information is known as user agent strings, and servers with a standard configuration store it in logfiles. These data can be accessed by survey software. These short strings include comprehensive information in a compressed form. As Callegaro (2010, 2013) shows, user agent strings are a cost efficient and easily accessible means to collect useful paradata. These characteristics are especially important for survey research, since several studies have shown that response behavior can differ between users of different devices (e.g., Mavletova 2013; Peytchev and Hill 2010). Given the increasing popularity of these paradata there is a surprising lack of detailed discussion on user agent strings in the social sciences. To our knowledge only Callegaro (2010, 2013) provides brief discussions on the strings' contents.

Apart from web surveys, user agent strings are of interest for other researchers with access to user data of web services (e.g., data on website visitors, or users of web applications). The information included in user agent strings may help to gain a deeper understanding of user characteristics (e.g., Mac users versus Windows users) and their use patterns (e.g., mobile use versus home use with a personal computer). Thus, the need to extract information from user agent strings covers a wide range of fields, for instance, computer sciences, economics, and the social sciences.

Until now, user agent strings had to be coded manually by referring to freely available sources like `http://www.useragentstring.com/` as suggested by Callegaro (2013) or by employing external tools, for example, the application programming interface (API) of the aforementioned website. Both approaches impose a burden on researchers using Stata (StataCorp 2019). First, researchers need to separately code the strings manually for every data set by rifling through the different sources of user agent string information. This approach is highly inefficient, since user agent strings are widely available data, and extracting information from them is often needed. Second, researchers need to convert data sets and switch between different software, which is labor intensive and takes up time that could be better used for substantive research.

The present article introduces the user written Stata command **parseuas**, which automatically extracts information from user agent strings and thus remedies the aforementioned shortcomings. We give a brief introduction on the technical properties and the structure of user agent strings and explain how to extract information from them. Then we explain how to collect these paradata with frequently used web survey software and, more generally, by using JavaScript. In the next section, we introduce the syntax and options of **parseuas**. After introducing the command, we provide a comprehensive example using a pooled data set of 29 web surveys, and in the process, demonstrate the application of **parseuas**.

2. User agent strings

Since many different browsers and devices are used to access the Internet, the software developers of web pages need to be able to detect the capabilities of browsers. This is necessary because different browsers have different ways of rendering and displaying web pages and vary in their implementation of JavaScript. With the emergence of more browsers and smartphones, variety has increased substantially. With respect to survey research, the ability of respondents to use different devices to participate in web surveys – and even change their device during an interview – poses new challenges in terms of equivalent measurement. A lot of research has identified best practices and problems in visual design (for an overview see Couper 2008; Dillman, Smyth, and Christian 2014; Tourangeau, Conrad, and Couper 2013), and the impact of participants who use smartphones and other mobile devices to answer web surveys. While the discussions about the best visual design continue, researchers still need to decide whether to develop their survey along the principles of a mode-specific design or opt for a uni-mode design. Unfortunately, a single right answer does not exist, since much depends on the research goals. Meanwhile, the software industry has coined the term mobile-first design, which means that developers program survey software with the goal to improve the survey experience for mobile participants. As it turns out, respondents using desktop computers and notebooks also can benefit from such an approach. Several web survey software companies claim that they can detect the device type being used and are able to utilize this information to send device-tailored questionnaires to respondents. Advanced survey software offers this capability as part of a rich set of features that survey researchers can use to decide how different types of questions (e.g., grid questions) should behave on different device types or whether mobile devices actually should receive a different questionnaire layout.

In summary, since survey researchers need a way to tap into the information that tells them what device is being used by respondents, the user agent string offers this type of information. Researchers can either use web survey software that includes the user agent string as part of the standard data download, or they can collect the user agent string themselves. Once the user agent strings are in the data set, the Stata command **parseuas** extracts the most useful information for further analyses.

2.1. The structure of a user agent string

The user agent string was developed so that each device can inform a server about the particular product and product version being used. In an ideal world, each device would send specific information to enable servers (and for the purpose of this article survey software) to detect the device so that the server knows the capabilities of the browser and can send appropriate web pages. However, as more products and new versions are developed, the variety of user agent strings has increased to many thousands. For a comprehensive overview of the history and current use of user agent strings and their different formats, see Zakas (2015, pp. 276–307).

The modern user agent string includes information about the web browser, the browser version, the rendering engine of the browser, the device type, and the operating system running on the device. In addition, the user agent string can contain additional device-specific information such as supported encryption or other proprietary information. Different companies are using different formats. Since the strings are produced to be parsed by specialized programs, it will be difficult or even impossible for most researchers to understand every part of

a given user agent string. By comparison, Internet Explorer 8 follows a rather easy-to-read format (Zakas 2015, p. 279):

> 'Mozilla/4.0 (compatible; MSIE Version; Operating System; Trident/TridentVersion)'

The first term 'Mozilla/4.0' is fixed and a remnant of the early days of the Internet. The same holds for the word 'compatible'.

In the next example, the interesting information is 'MSIE 8.0', which indicates Internet Explorer version 8. 'Windows NT 5.1' actually means Windows XP. 'Trident/4.0' is a token used to indicate that this is Internet Explorer version 8. This enables us to detect the correct version even when Internet Explorer is running in compatible mode; in this case, the first part would be 'MSIE 7.0'. While older detection scripts would see Internet Explorer 7, newer scripts are able to look for 'Trident' and detect Internet Explorer 8. This also illustrates why a simple string match approach would deliver insufficient and potentially wrong data. The '.NET CLR' entries are part of the potentially long series of additional information, in this case indicating that the machine runs five different versions of the .NET framework:

> 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)'

A principal problem of parsing user agent strings is that there is only modest uniformity amongst the different parts of the user agent string, which complicates coding. Furthermore, due to fast technological development, the content of user agent strings changes over time. As a consequence, the coding scheme or dictionary used for coding has to be updated on a regular basis to facilitate correct coding. We used information from the World Wide Web Consortium[1], the Mozilla Developer Network[2], and specialized web pages[3] to generate the coding scheme for this Stata command.

The script is written in a way so that it also works with a 10-year older format of user agent strings, which is possible because the industry has not changed the basic user agent string format. Instead, user agent strings have been gradually complemented with additional content for newly developed browsers, devices, and operating systems.

2.2. Device type: mobile, tablet, or desktop

The identification of the device used by the client is a common application of user agent strings. In general, three types of devices can be detected: desktop computers (including notebooks, ultrabooks, and netbooks), tablet computers, and mobile phones. Depending on the brand and type of the device, it can be identified directly or by combining different pieces of information. For instance, the user agent string of Android smartphones contains the terms 'android' and 'mobi(le)' whereas the user agent string of Android tablets only contains 'android' or a combination of 'android' and 'tablet'. The user agent string of other brands

[1] http://www.w3.org/TR/2014/WD-UAAG20-Reference-20140925/

[2] https://developer.mozilla.org/en-US/docs/Browser_detection_using_the_user_agent, https://developer.mozilla.org/en-US/docs/Web/HTTP/Gecko_user_agent_string_reference

[3] http://www.useragentstring.com/ and http://user-agent-string.info/

of mobile phones include the terms 'iPhone', 'Windows Phone', 'Symbian', or 'BlackBerry'. Other tablet computers are associated with the terms 'iPad', 'Playbook', or 'Kindle (Fire)'. User agent strings that do not include any of these terms are most likely desktop computers. However, some minor exceptions exist, especially since some older devices have to be identified by other information included in the user agent string, for instance, the device number.

2.3. Operating system

The operating system is given in most user agent strings. In addition, it often is possible to identify the version of the operating system. For instance, the user agent string 'Windows NT 6.1' refers to Windows 7, 'Windows NT 6.3' is Windows 8.1, whereas the Windows NT token's value changed to 'Windows NT 10.0' with the new Windows 10.0 operating system. In some cases, the user agent string also contains information on the hardware of the device (e.g., if the software runs on a 32- or 64-bit system, or if a Macintosh computer is used).

2.4. Browser name, version, and rendering engine

With regard to the web browser, the user agent string includes several bits of information. First, the rendering engine can be Gecko, WebKit, AppleWebKit, Presto, Trident, EdgeHTML, or Blink. Second, the user agent string usually includes the name of the browser. The currently most popular browsers are Firefox, Internet Explorer, Chrome, Safari, and Opera. Third, in most instances, the user agent string includes the version of the web browser. Usually, the version directly follows the browser name (e.g., 'Firefox/*.*' or 'MSIE *.*'). Detection of the browser and the browser version are complicated due to some notable exceptions. First, the same user agent string can be used potentially by different browsers, for example, some Chrome versions use the Safari user agent string. Second, some specific issues occur with respect to gathering the version of the browser that needs to be considered (e.g., the version number statement changes between different versions of the Opera browser). Accordingly, rules for the detection of the browser name and version have to take into account some specific cases.

2.5. Examples indicating different devices and browsers

In the following, we discuss five instructive examples of common combinations of the browser, operating system, and device in user agent strings.

The first example contains the substring 'Firefox/31.0' which indicates that the browser is Firefox 31.0. The expression 'Gecko' reveals that the rendering engine of the browser is Gecko. 'Windows NT 6.1' is Windows 7, while 'WOW64' shows that a 32-bit application is running on a 64-bit processor. The computer is a desktop or notebook.

'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0'

The second example is a Chrome browser in the version 34.0.1847.114 as indicated by the substring 'Chrome/34.0.1847.114'. The rendering engine is AppleWebKit. Further, it can be inferred from the expression 'Android 4.1.2' that the operating system is Android version 4.1.2 running on a tablet computer because the user agent string does not include 'mobi(le)'. Accordingly, the device number 'GT-P5100' shows that the device is a Samsung Galaxy Tab 2 10.1.

> 'Mozilla/5.0 (Linux; Android 4.1.2; GT-P5100 Build/JZO54K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.114 Safari/537.36'

The third example is Chrome 34.0.1847.114, which uses an AppleWebKit rendering engine. It can be inferred from the combination of the expressions 'Android 4.4.2' and 'Mobile' that the device is a smartphone using Android 4.4.2 as the operating system. The term 'GT-I9505' tells us that the device is a Samsung Galaxy S4.

> 'Mozilla/5.0 (Linux; Android 4.4.2; GT-I9505 Build/KOT49H) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.114 Mobile Safari/537.36'

The substrings 'Safari' and 'Version/7.0' indicate that the browser in the fourth example is Safari 7.0. The combination of 'iPhone' and 'OS 7_1_1' further reveals that the device is an iPhone using iOS 7.1.1 as the operating system.

> 'Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_1 like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D201 Safari/9537.53'

The last example is a desktop or notebook computer. The substrings 'Safari' and 'Version/7.0.3' show that the browser is Safari 7.0.3. The operating system is Mac OS X 10.9.3 as displayed by the expression 'Mac OS X 10_9_3'.

> 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/537.75.14'

3. How to obtain the user agent string during a web survey

Basically, there are two ways to access user agent strings when conducting a web survey. The information is either part of the standard export of a survey software tool or the survey researcher needs to insert program code in the web survey to obtain and store the information.

Many survey software tools provide this information as a variable in the standard data export. These variables usually have names like "browser" or "agent". The information usually is obtained when the respondent begins to answer a survey. This means that the variable stores the user agent from the device that is being used when a respondent starts a survey. This information might be sufficient to get an overview of the devices being used; however, when survey researchers expect that respondents may pause a survey and switch their devices during participation, one measurement will not convey the full story of which devices have been used to complete the survey. In this case, researchers could include program code on every page of a web survey or at least on several pages to capture the user agent information. When comparing six different surveys in a Dutch probability-based panel (LISS), Lugtig and Toepoel (2016) found that between 1.5% and 4.7% of respondents switched from a personal computer to a mobile device in a following survey. However, the likelihood of switching from a tablet or smartphone to a personal computer was much higher. Among respondents who used a mobile device in a survey, between 16.4% and 46.0% switched to using a PC in a following survey.

When looking at the variables in a data set, survey researchers will find it easy to see whether the user agent string is provided by the survey software tool as a standard output variable because of its unique content and string type (see Section 2). Since hundreds of web survey software tools are available to conduct web surveys (e.g., http://www.websm.org/c/1283/Software/?preid=1283 lists about 340 web survey software tools) and updates often are released several times a year, a list of features would soon be outdated. The authors of the present article suggest as general rule of thumb: the more features and flexibility a web survey software tool offers, the more likely it is that the user agent string is already in the data set. Whenever a survey software tool offers the flexibility to insert HTML and JavaScript code to program survey questions, it is always possible to collect the user agent string. It seems that professional survey software tools that focus on the low-budget, fast, do-it-yourself market – with a large but fixed set of features – are less likely to offer user agent information.

We checked whether the user agent string is available for a couple of web survey software tools.[4] At the time of the writing of this article, customer support at Google Consumer Surveys and SurveyMonkey informed us that they do not provide access to the raw user agent string data in their standard data output, and it also is not possible to program your own survey questions from scratch with the standard tools they offer. However, if you are a professional programmer and plan to program your own tool or app and are able to access an available API, it is likely that you also will be able to access this information.

To give some examples, the following software tools provide the user agent string as a standard variable or allow the collection of this information as part of a web survey: Blaise, Confirmit, EFS, Illume, and Qualtrics. It should be noted that companies who support research by providing respondents, such as online access panel providers[5] or paid crowd-sourcing services (e.g., Amazon Mechanical Turk or WorkHub), are not part of this discussion whether the user agent string can be obtained or not because a researcher would still need a web survey software tool or at least an online form to collect the data.

Whenever it is possible to insert your own survey questions in a web survey software tool, the user agent string can be obtained by adapting and inserting the following code.[6] We provide two examples. The first includes an input field that will be filled with the user agent string.

```
<html>
<!-- This is the input field which will be filled with the user agent
 string. The input field is usually invisible and the contents is submitted
 to the server together with other information when a respondent clicks
 on a 'next'-button. -->
<input id="v1" value="" type="text" size="120">
<script type="text/javascript">
/* to store the user agent string in an input field we need to know the
 id of the input field */
var id = "v1";
```

[4]The authors of the present article do not endorse any particular products. All product names are only provided as examples. A survey researcher should always consider the full set of required features before making a purchase decision. Kaczmirek (2008) gives an overview of the features of web survey software tools that should be taken into consideration.

[5]Many professional companies are members of professional survey and market research organizations such as http://www.aapor.org/ or http://www.dgof.org/.

[6]The code can be downloaded from http://www.kaczmirek.de/stata/uas/getuseragent.html.

```
// to access the input field we get the reference to the input field
var useragent = document.getElementById(id);
/* we change the value of the input field to contain the user agent
 string */
useragent.value = navigator.userAgent;
</script>
```

In contrast, the second example features a hidden input field.

```
<!-- This is a shorter example with hidden input field -->
<input id="v2" value="" type="hidden" size="120">
<script type="text/javascript">
var useragent = document.getElementById("v2");
useragent.value = navigator.userAgent;
</script>
</html>
```

4. The parseuas command

4.1. Description

The **parseuas** command is built on an engine that acknowledges the variable structure of user agent strings. As discussed previously in this article, user agent strings incorporate a varying degree of information on browser, operating system, and device. This information is partially non-consistent over time, for instance, the string 'Windows NT 6.3' does not refer to a latter version of Windows NT but to Windows 8.1, while the Android operating system is expressed by a string 'Android *.*' where *.* resembles the Android's version. Accordingly, when programming **parseuas**, the parsing process was designed stepwise, to search the user agent string for information on the browser, then the operating system, and finally the device. In each step, we relied on information from a variety of sources (see Section 2) to draw as much information from the string as possible.

Each step builds on a sequence of queries about whether the user agent string contains an identifying piece of information. Most of these queries rely on regular expressions, especially the `regexm` function because the exact position of information in the user agent string is unknown. `regexm` verifies whether a string (in this case the user agent string) contains another string (i.e., the identifying information). For example, to detect an Android operating system, we would use an expression like:

```
(...) regexm(`useragentstring', "Android") (...)
```

The **parseuas** command relies on queries like this to parse user agent strings into useful information. Depending on what information the user requires, respective variables are created.

The code is written to minimize maintenance and consider expectations of future, yet unknown user agent strings. Thus, **parseuas** is able to automatically parse the version of the most common browsers and operating systems. For example, the code can detect "Android 5.5"

even if it was released later than the most recent version of **parseuas**. In this example it would detect 'Android' and then search the string for information on the version.

However, when extracting information from user agent strings, we sometimes fail to achieve optimal results. This failure may be a result of a less common user agent or technological development (e.g., new browsers or operating systems) not yet covered by **parseuas**.[7] Thus, we believe residual categories to be a crucial indicator of validity when applying the **parseuas** command. To prevent misinterpretation, the command is designed to extract as much information as possible. If detailed information is missing (e.g., the version of an operating system), **parseuas** provides more general information. For example, the user agent string is searched for information about whether the operating system was Android. If the information on the version of the operating system is missing, the user agent string will be coded as "Android (other)". Those user agent strings that contain non-interpretable information will be coded into broader residual categories, e.g., "Browser (other)". Note that these residual categories only apply when more detailed information cannot be extracted.

4.2. Syntax

The syntax for **parseuas** to extract information from user agent strings is:

```
parseuas string [if] [in] [, browser(newvar) browserversion(newvar)
os(newvar) device(newvar) smartphone(newvar) tablet(newvar) numeric noisily]
```

4.3. Options

The **parseuas** command optionally stores the information from the user agent strings in new variables.

- `browser(newvar)` generates a variable containing the information on the browser name.
- `browserversion(newvar)` generates a variable containing the information on the version of the browser.
- `os(newvar)` generates a variable containing the information on the operating system of the device.
- `device(newvar)` generates a variable containing the information on the device type.
- `smartphone(newvar)` generates a dummy variable, which indicates whether a smartphone was used.
- `tablet(newvar)` generates a dummy variable, which indicates whether a tablet computer was used.
- `numeric` causes **parseuas** to create numeric variables instead of strings.
- `noisily` provides output of frequency tables for browser name, browser version, operating system, and device type.

[7] Note that tools exist that can be used to mask or manipulate a user agent string and, hence, some strings may contain non-interpretable information.

4.4. Examples

To demonstrate the use of **parseuas**, we present two examples. The first illustrates the application of the **parseuas** command, while the second provides the reader with a basic understanding of how the content of user agent strings has changed over time and, hence, how technical development has progressed over several years. In the second example, we also show how to interpret the residual categories to face-validate the application of **parseuas**.

Application of ***parseuas***

To illustrate the application of **parseuas**, we pooled 29 web surveys, which were conducted between 2009 and 2015 as part of the German Longitudinal Election Study (Rattinger, Roßteutscher, Schmitt-Beck, Weßels, and Wolf 2009–2015). Each survey included about 1,100 respondents, thus giving us a total of 36,575 observations. The user agent strings were collected by the web survey software and included in the data set as a string variable.

We used **parseuas** with all options to extract the information from the user agent strings:

```
. parseuas useragent, bro(browser) browserv(browserversion)
> os(operatingsystem) device(device) smart(smartphone) tab(tablet) numeric
> noisily
```

parseuas performed well in our example and extracted the requested information from 36,575 strings in approximately 2 seconds. After `noisily` running **parseuas**, we could assess the data stored in the newly created variables, and the command reported basic findings on the data:

```
     Browser name |      Freq.     Percent        Cum.
------------------+-----------------------------------
   Android Webkit |         834        2.28        2.28
  Browser (other) |          64        0.17        2.46
           Chrome |       3,749       10.25       12.71
             Edge |          70        0.19       12.90
          Firefox |      17,603       48.13       61.03
        Iceweasel |          22        0.06       61.09
Internet Explorer |      11,188       30.59       91.67
             Iron |          21        0.06       91.73
          K-Meleon |           9        0.02       91.76
          Maxthon |          15        0.04       91.80
          Netscape |           5        0.01       91.81
             Opera |         635        1.74       93.55
            Safari |       2,262        6.18       99.73
         SeaMonkey |          67        0.18       99.92
              Silk |          31        0.08      100.00
------------------+-----------------------------------
             Total |      36,575      100.00

       Browser version |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
(...)
```

```
    Android Webkit 4.0 |        629        1.72        2.28
(...)
  Chrome 40.0.2214.115 |        268        0.73        9.84
(...)
         Edge 12.10240 |         70        0.19       12.90
(...)
          Firefox 40.0 |        674        1.84       58.05
(...)
 Internet Explorer 8.0 |      5,143       14.06       88.25
(...)
        K-Meleon 1.5.4 |          3        0.01       91.76
(...)
            Opera 9.64 |        111        0.30       93.54
(...)
            Safari 8.0 |        614        1.68       99.26
(...)
        SeaMonkey 2.30 |          4        0.01       99.89
(...)
-----------------------+-----------------------------------
                 Total |     36,575      100.00

 Operating system version |      Freq.     Percent        Cum.
--------------------------+-----------------------------------
          Android (other) |        128        0.35        0.35
(...)
            Android 4.1.2 |        226        0.62        1.59
(...)
     Linux Ubuntu (other) |        127        0.35        5.32
(...)
            Mac OS X 10.6 |        107        0.29        6.34
(...)
               Windows 7 |     11,034       30.17       40.17
             Windows 8.0 |        387        1.06       41.23
(...)
               iOS 8_1_3 |        158        0.43       99.16
(...)
--------------------------+-----------------------------------
                   Total |     36,575      100.00

           Device type |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
        Device (other) |     33,337       91.15       91.15
Mobile phone (Android) |      1,137        3.11       94.26
Mobile phone (Windows) |         44        0.12       94.38
 Mobile phone (iPhone) |        739        2.02       96.40
  Mobile phone (other) |         18        0.05       96.45
      Tablet (Android) |        450        1.23       97.68
```

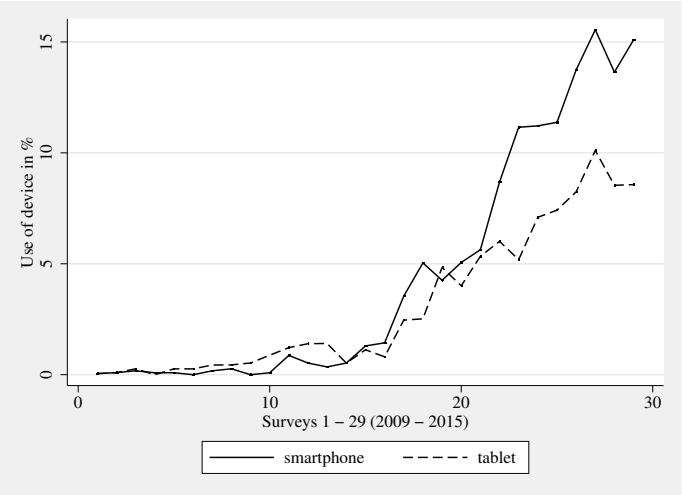

Figure 1: Use of smartphones and tablets in 29 web surveys from 2009 to 2015.

```
      Tablet (Windows) |        115        0.31       97.99
         Tablet (iPad) |        735        2.01      100.00
-----------------------+-----------------------------------
                 Total |     36,575      100.00
```

Further applications

The data set we collected enabled us to illustrate how user agent strings changed between 2009 and 2015. As mentioned previously, due to technological development, new user agent strings emerge. For example, when a new operating system is released, we can find this information in the user agent string.

In our example, we focus on the emergence of mobile devices and different versions of Windows. The former is an application that we would expect to be used in the context of a web survey, whereas the latter illustrates the use of **parseuas** when, for instance, analyzing data of website users.

We relied on information on the device type to analyze the use of mobile devices to complete the 29 web surveys over time (2009–2015). The results shown in Figure 1 reflect the technological development that has led to the increasing spread of mobile devices. The ability of **parseuas** to detect the increasing use of smartphones and tablets is due to user agent strings containing the necessary information to detect a mobile device.[8]

In Figure 2, we plotted the use of different versions of Windows over time. Note that Windows 7 was released in October 2009, Windows 8 in October 2012, and Windows 10 in July 2015. Similar to the patterns of use of mobile devices, newly developed versions of the operating system emerge and become increasingly used, while older versions such as Windows XP and Vista vanish. This development of new technology is reflected in changing user agent strings which, again, enabled us to detect them. Our data show the emergence of user agent strings with information referring to Windows 7 in survey 8, Windows 8 in survey 19, and Windows

[8]With respect to survey 17, the trend may be explained partially by a change in the online access panel provider.

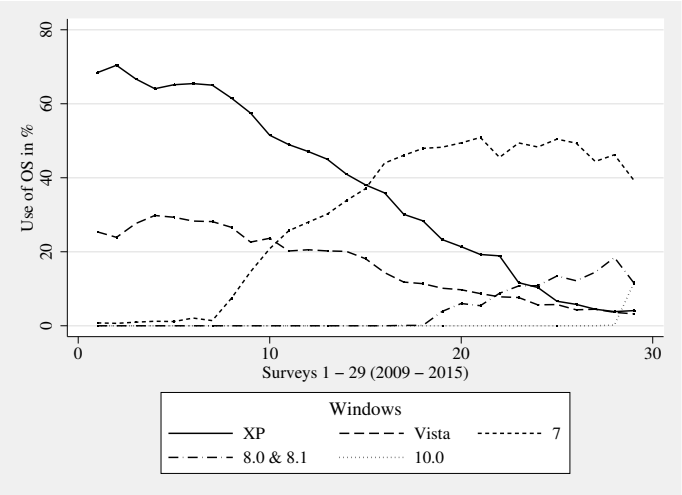

Figure 2: Use of Windows in 29 web surveys from 2009 to 2015.

Information	%
Browser (other)	0.17
OS (other)	0.09
Mobile phone (other)	0.05
Tablet (other)	0.00

Table 1: Relative frequencies of residual categories for parsed information.

10 in survey 29.[9] The appearance of the respective user agents fits the release dates: survey 8 was carried out in December 2009, survey 19 in January 2013, and survey 29 in September 2015.

User agent strings may lack information or include content that **parseuas** is not able to fully interpret. As outlined in Section 4.1, the command is set up to handle these cases by coding the available information into residual categories. Evidently, the residual categories are an important source when validating the results of the application of **parseuas**. "Browser (other)", "OS (other)", "Mobile phone (other)", or "Tablet (other)" indicate that **parseuas** did not identify the exact type of browser, operating system, or device. In our examples, we relied on a data set that was collected when we were developing **parseuas**. Hence, only a few user agent strings (<1% in each variable) were not optimally parsed, and these numbers are what we consider an ideal case of residuals when using the command. Table 1 illustrates the distribution of "others" for parsed information on the browser, operating system, and device.

5. Remarks

The Stata command will be updated on a regular basis to keep up with the development of new browsers, operating systems, and devices. Thus, we recommend using the latest version of **parseuas**. For this purpose, Stata provides the command `adoupdate`, which automatically

[9]Some respondents (<1%) had been using Windows 7, 8, and 10 before these surveys. Presumably, these operating systems were pre-release versions.

updates user-written ados (see **adoupdate**). In addition, to guarantee the reproducibility of analyses using **parseuas**, we recommend citing the ado with information on the used version, e.g., as in Roßmann and Gummer (2016a).

Moreover, as outlined in Section 4.1 (for an application, see Section 4.4), we recommend inspecting the frequencies of residual categories after applying **parseuas** to a data set. If user agent strings cannot be parsed in an optimal way, they are coded into these categories. This situation may be the result of either drawing on a data set including a large amount of uncommon user agents or technological developments (e.g., completely new browsers or operating systems) that have not yet been accounted for in the most recent version of **parseuas**. In the exemplary application based on 36,575 user agent strings that we collected over 7 years, the residual categories did not exceed 1% of all observations. Hence, we would recommend this threshold as a rule of thumb for researchers to face-validate the successful application of **parseuas**.

6. Conclusion

In this article, we introduced the new Stata command **parseuas** to extract detailed information from user agent strings. These data can be used for methodological and substantive research questions. Particularly in the field of web survey research, interest in paradata (e.g., device types) has been increasing. Apart from survey methodology, user agent strings are commonly available data on the user level when using web services. Thus, analyzing information from user agent strings is of great importance to researchers and practitioners in a multitude of fields ranging from computer sciences to market research.

As our example illustrates, **parseuas** provides Stata users with an easily applicable command to automatically generate meaningful data. Nevertheless, new user agents will emerge due to technological developments (e.g., new devices, browsers, and operating systems), or users may want to extract information that is not provided by the latest version of **parseuas**. Thus, our article details how to parse user agent strings, which should enable users to modify the **parseuas** command to serve their individual purposes.

In addition to detailed information on user agent strings and how to parse them, we have provided an overview of how to collect these paradata with frequently used web survey software and, more generally, by implementing JavaScript code.

Acknowledgments

We would like to acknowledge the support of GESIS, which ultimately led to the creation of **parseuas**. We would like to thank Andrei Artimof, Wolfgang Bandilla, Ulrich Krieger, and Bart Meuleman for testing and reviewing earlier versions of **parseuas**.

References

Bassili JN (1993). "Response Latency versus Certainty as Indexes of the Strength of Voting Intentions in a CATI Survey." *Public Opinion Quarterly*, **57**(1), 54–61. doi:10.1086/269354.

Biemer PP, Chen P, Wang K (2013). "Using Level-of-Effort Paradata in Non-Response Adjustments with Application to Field Surveys." *Journal of the Royal Statistical Society A*, **176**(1), 147–168. doi:10.1111/j.1467-985x.2012.01058.x.

Callegaro M (2010). "Do You Know Which Device Your Respondent Has Used to Take Your Online Survey?" *Survey Practice*, **3**(6), 1–13. doi:10.29115/sp-2010-0028.

Callegaro M (2013). "Paradata in Web Surveys." In F Kreuter (ed.), *Improving Surveys with Paradata: Analytic Uses of Process Information*, pp. 261–280. John Wiley & Sons, Hoboken.

Couper MP (2000). "Usability Evaluation of Computer-Assisted Survey Instruments." *Social Science Computer Review*, **18**(4), 384–396. doi:10.1177/089443930001800402.

Couper MP (2008). *Designing Effective Web Surveys*. Cambridge University Press, New York.

Couper MP, Kreuter F (2013). "Using Paradata to Explore Item Level Response Times in Surveys." *Journal of the Royal Statistical Society A*, **176**(1), 271–286. doi:10.1111/j.1467-985x.2012.01041.x.

Dillman DA, Smyth JD, Christian LM (2014). *Internet, Phone, Mail, and Mixed-Mode Surveys: The Tailored Design Method*. John Wiley & Sons, Hoboken.

Gummer T, Roßmann J (2015). "Explaining Interview Duration in Web Surveys: A Multilevel Approach." *Social Science Computer Review*, **33**(2), 217–234. doi:10.1177/0894439314533479.

Kaczmirek L (2008). "Internet Survey Software Tools." In NG Fielding, RM Lee, G Blank (eds.), *The Sage Handbook of Online Research Methods*, pp. 236–254. Sage, London.

Lugtig P, Toepoel V (2016). "The Use of PCs, Smartphones, and Tablets in a Probability-Based Panel Survey: Effects on Survey Measurement Error." *Social Science Computer Review*, **34**(1), 78–94. doi:10.1177/0894439315574248.

Mavletova A (2013). "Data Quality in PC and Mobile Web Surveys." *Social Science Computer Review*, **31**(6), 725–743. doi:10.1177/0894439313485201.

Mayerl J (2013). "Response Latency Measurement in Surveys. Detecting Strong Attitudes and Response Effects." *Survey Methods: Insights from the Field*. doi:10.13094/SMIF-2013-00005. Retrieved from https://surveyinsights.org/?p=1063.

Meyer M, Schoen H (2014). "Response Latencies and Attitude-Behavior Consistency in a Direct Democratic Setting: Evidence from a Subnational Referendum in Germany." *Political Psychology*, **35**(3), 431–440. doi:10.1111/pops.12039.

Peytchev A, Hill CA (2010). "Experiments in Mobile Web Survey Design: Similarities to Other Modes and Unique Considerations." *Social Science Computer Review*, **28**(3), 319–335. doi:10.1177/0894439309353037.

Rattinger H, Roßteutscher S, Schmitt-Beck R, Weßels B, Wolf C (2009–2015). *Long-Term Online Tracking, T1-T29 (ZA5334-ZA5351 and ZA5719-ZA5729)*. GESIS Data Archive, Cologne.

Roßmann J, Gummer T (2016a). ***parseuas**: Stata Module to Extract Detailed Information from User Agent Strings*. Version 1.3, URL http://EconPapers.repec.org/RePEc:boc:bocode:s457937.

Roßmann J, Gummer T (2016b). "Using Paradata to Predict and Correct for Panel Attrition." *Social Science Computer Review*, **34**(3), 312–332. doi:10.1177/0894439315587258.

Sinibaldi J, Trappmann M, Kreuter F (2014). "Which Is the Better Investment for Nonresponse Adjustment: Purchasing Commercial Auxiliary Data or Collecting Interviewer Observations?" *Public Opinion Quarterly*, **78**(2), 440–473. doi:10.1093/poq/nfu003.

StataCorp (2019). *Stata Statistical Software: Release 16*. StataCorp LLC, College Station. URL http://www.stata.com/.

Stern MJ (2008). "The Use of Client-Side Paradata in Analyzing the Effects of Visual Layout on Changing Responses in Web Surveys." *Field Methods*, **20**(4), 377–398. doi:10.1177/1525822x08320421.

Tourangeau R, Conrad F, Couper M (2013). *The Science of Web Surveys*. Oxford University Press, Oxford.

Zakas NC (2015). *Professional JavaScript for Web Developers*. 3rd edition. John Wiley & Sons, Indianapolis. doi:10.1002/9781118722176.

Affiliation:

Joss Roßmann, Tobias Gummer
GESIS – Leibniz Institute for the Social Sciences
68159 Mannheim, Germany
E-mail: joss.rossmann@gesis.org, tobias.gummer@gesis.org

Lars Kaczmirek
University of Vienna
1010 Vienna, Austria
E-mail: lars.kaczmirek@univie.ac.at

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2020, Volume 92, Code Snippet 1 *Submitted:* 2016-01-08
doi:10.18637/jss.v092.c01 *Accepted:* 2018-07-13