Overview of the vietnameseseConverter package

Juergen Niedballa (niedballa@izw-berlin.de)

2021-10-15

Contents

Introduction .......................... 1
  Background .................................. 1
  Installation .................................. 2

Example applications ................. 2
  Vectors ................................... 3
  Data frames ................................ 3
  Spatial data ............................... 8

Introduction

This package helps you read and use data from Vietnamese sources in R. Such data often use Vietnamese legacy character encodings such as TCVN (Vietnam Standards / Tieu chuan Viet Nam) and are still in use today. Such data are not read correctly in R (which doesn’t seem to support these Vietnamese encodings). To correct this problem and make such data available in R, this package converts data in legacy Vietnamese encodings to the correct Unicode characters. The main function is decodeVN.

The package currently supports conversion between the following encodings:

- Unicode (UTF-8)
- TCVN3
- VPS
- VISCII

It converts between any of these encodings, but most commonly one would want to convert legacy encodings to their correct Unicode characters. The package supports character vectors and data frames (the character columns thereof).

Background

For those interested here is some background on character sets and encodings.

A character set is a collection of characters used in languages.

A coded character set assigns a unique number to each character in a character set. The unique values in a coded character sets are known as code points (e.g. “U+0041 ” corresponds to capital letter A in Unicode).

Unicode (also known as Universal Coded Character Set) is a modern coded character set that contains over 100,000 characters from most writing systems of the world.

Character encodings are the method by which a coded character set is converted to binary. UTF-8 is a commonly used encoding for Unicode characters (UTF stands for Unicode Transformation Format).
Historically, the Vietnamese language could not be represented well using 128 characters in the ASCII standard. Hence, several character encodings were developed for the Vietnamese language with the aim of representing Vietnamese characters while fitting into 1 byte (=8-bit, allowing up to 255 characters). To achieve that, some some code points were reassigned and differ from today’s standards like Unicode. Thus, when reading data that use those Vietnamese encodings on systems that assume e.g. UTF-8 encoding (Unicode), we get gibberish text (also known as mojibake).

Let’s take a Vietnamese string that is supposed to be:

"Quảng Trị, An Đô, Thừa Thiên Huế"

If it is encoded using a legacy Vietnamese encoding, it might displayed as something like:

"Quãng Trị, An Đôn, Thọa Thiên Huế"
"Quãng Trị, An Đôn, Thọa Thiên Huế"
"Quãng Trị, An Đôn, Thọa Thiên Huế"
when it is read in R or other software (depending on the specific encoding of the data).

These days, almost all Vietnamese computer systems use Unicode and there is no need to use the legacy encodings anymore. Nevertheless, historic data may still be encoded in these legacy encodings and require conversion.

This package fixes this problem by converting the garbled strings back to their original (with or without the diacritics). When saving the output in R, it will use standard encodings and data will read correctly in the future.

You can check the output from this package using e.g. this website http://www.enderminh.com/minh/vnc/versions.aspx.

If you want to know more about the technicalities of encodings and character sets, see https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/.

Installation

The package can be installed from CRAN:

install.packages("vietnameseConverter")

and loaded into R via:

library(vietnameseConverter)

Example applications

Disclaimer: Printing Unicode characters from R function output in package vignettes is difficult. Often R will print the Unicode escape characters (the code point) instead of the actual characters, e.g. <U+1EBF> instead of “ế”.

This is a limitation in vignettes. I’ll try to circumvent it in this vignette as much as possible, e.g. by using DT::data.table() and by copy/pasting the correct console output below the output with Unicode characters. Even though it doesn’t print nicely in this vignette, the converted data are correct (as you can see if you use data.table(), View(), or print vectors in the R console.

Run the code from this vignette in your R console to see the actual output, or try the examples in ?decodeVNN
Vectors

Let’s consider a simple vector with garbled characters:

```r
c(string_garbled <- c("Qu¶ng TrÞ", "An §«n", "Thõa Thiªn HuÕ")

## [1] "Qu¶ng TrÞ" "An §«n" "Thõa Thiªn HuÕ"
```

It can be fixed using the `decodeVN` function:

```r
tmp <- decodeVN(string_garbled)

tmp

## [1] "Qu<U+1EA3>ng Tr<U+1ECB>" "An Ðôn" "Th<U+1EEB>a Thiên Hu<U+1EBF>"
```

In your R console, the Unicode characters will be printed correctly as:

```r
## [1] "Quảng Trị" "An Đôn" "Thừa Thiên Huế"
```

By default, the output contains character with diacritics (accents). We can also get output without the diacritics (plain ASCII letters) by setting argument `diacritics = FALSE`:

```r
decodeVN(string_garbled, diacritics = FALSE)

## [1] "Quang Tri" "An Don" "Thua Thien Hue"
```

By default, `decodeVN()` attempts to convert TCVN3 to Unicode characters (as shown in this example). If your data are in different encodings, set argument `from`.

For example, a string in VPS encoding is converted via:

```r
string_garbled_vps <- c("Quãng TrÎ", "An ñôn", "ThØa Thiên Hu‰")
decodeVN(string_garbled_vps, from = "VPS")

## [1] "Qu<U+1EA3>ng Tr<U+1ECB>" "An Ðôn" "Th<U+1EEB>a Thiên Hu<U+1EBF>"
```

Which in your R console will be

```r
## [1] "Quảng Trị" "An Đôn" "Thừa Thiên Huế"
```

There is also an argument `to`, which by default is set to `to = "Unicode"`. If you want output that simulates data encoded in one of the supported Vietnamese encodings, set argument `to` accordingly, e.g.

```r
string_tcvn_to_vps <- decodeVN(x = string_garbled, from = "TCVN3", to = "VPS")
string_tcvn_to_vps

## [1] "Quãng TrỊ" "An ñôn" "ThØa Thiên Hu‰"
```

This string is now garbled the VPS-way. It can still be converted to Unicode:

```r
decodeVN(string_tcvn_to_vps, from = "VPS")

## [1] "Qu<U+1EA3>ng Tr<U+1ECB>" "An Ðôn" "Th<U+1EEB>a Thiên Hu<U+1EBF>"
```

which is:

```r
## [1] "Quảng Trị" "An Đôn" "Thừa Thiên Huế"
```

Data frames

We can also convert entire data frames at once. It will only convert character columns and leave the other columns untouched. If you have factor columns, convert them to character first.
The package contains a list containing several data frames which simulate the problems with different encodings. These data frames show what the problems with different encodings look like when the data are loaded in an environment that assumes UTF-8 / native encoding). The data frame is based on the table of provinces in https://en.wikipedia.org/wiki/Provinces_of_Vietnam.

Loading the list of sample data.

```r
data(vn_samples)
```

vn_samples is a list of four data frames. See `?vn_samples` for details.

The first item `$Unicode` shows the correct characters.

```r
head(vn_samples$Unicode)
```

<table>
<thead>
<tr>
<th></th>
<th>Province_city</th>
<th>Administrative_center</th>
<th>Area_km2</th>
<th>Density_perkm2</th>
<th>HDI_2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B&lt;U+1EAF&gt;c Giang</td>
<td>B&lt;U+1EAF&gt;c Giang</td>
<td>3895.59</td>
<td>463</td>
<td>0.711</td>
</tr>
<tr>
<td>2</td>
<td>B&lt;U+1EAF&gt;c K&lt;U+1EA1&gt;n</td>
<td>B&lt;U+1EAF&gt;c K&lt;U+1EA1&gt;n</td>
<td>4859.96</td>
<td>65</td>
<td>0.685</td>
</tr>
<tr>
<td>3</td>
<td>Cao B&lt;U+1EB1&gt;ng</td>
<td>Cao B&lt;U+1EB1&gt;ng</td>
<td>6700.26</td>
<td>79</td>
<td>0.653</td>
</tr>
<tr>
<td>4</td>
<td>Hà Giang</td>
<td>Hà Giang</td>
<td>7929.48</td>
<td>108</td>
<td>0.586</td>
</tr>
<tr>
<td>5</td>
<td>L&lt;U+1EA1&gt;ng Son</td>
<td>L&lt;U+1EA1&gt;ng Son</td>
<td>8310.09</td>
<td>94</td>
<td>0.707</td>
</tr>
<tr>
<td>6</td>
<td>Phú Th&lt;U+1ECD&gt;</td>
<td>Phú Th&lt;U+1ECD&gt;</td>
<td>3534.56</td>
<td>414</td>
<td>0.715</td>
</tr>
</tbody>
</table>

Note: when printing a data frame in the R console, the Unicode characters are not displayed properly, which is a limitation of the print method for data frames in R (it's the same issue as mentioned in the disclaimer above). The characters are correct though and show correctly when printing the columns as vectors:

```r
head(vn_samples$Unicode$Province_city)
```

```r
[1] "Bắc Giang" "Bắc Kạn" "Cao Bằng" "Hà Giang" "Lạng Sơn" "Phú Thọ"
```

Which again is not formatted nicely in the vignette (but would be formatted correctly in your console). The string is:

```r
[1] "Bắc Giang" "Bắc Kạn" "Cao Bằng" "Hà Giang" "Lạng Sơn" "Phú Thọ"
```

It also shows correctly when using `View` (not shown here, it works in interactive sessions):

```r
View(vn_samples$Unicode)
```

It is also shown correctly when using `DT::datatable`:

```r
DT::datatable(vn_samples$Unicode)
```
The other data frames in `vn_samples` show garbled text from several Vietnamese encodings. Here’s the example in TCVN3 encoding:

```r
DT::datatable(vn_samples$TCVN3)
```

One can easily convert the entire data frame:
# take data frames out of list for easier readability
df_unicode <- vn_samples$Unicode
df_tcvn3 <- vn_samples$TCVN3

# conversion from TCVN3 to Unicode (default)
df_tcvn3_converted <- decodeVN(df_tcvn3)

# print output
DT::datatable(df_tcvn3_converted)

After conversion it is identical to the original with Unicode characters

all.equal(df_unicode, df_tcvn3_converted)

## [1] TRUE

Again, we can choose to return characters without accents by setting diacritics = FALSE:
df_tcvn3_converted2 <- decodeVN(df_tcvn3, diacritics = FALSE)
DT::datatable(df_tcvn3_converted2)
## Table of Provinces in Vietnam

<table>
<thead>
<tr>
<th>Province_city</th>
<th>Administrative_center</th>
<th>Area_km2</th>
<th>Density_perkm2</th>
<th>HDI_2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bắc Giang</td>
<td>Bắc Giang</td>
<td>3895.59</td>
<td>463</td>
<td>0.711</td>
</tr>
<tr>
<td>Bắc Kạn</td>
<td>Bắc Kạn</td>
<td>4859.96</td>
<td>65</td>
<td>0.685</td>
</tr>
<tr>
<td>Cao Bằng</td>
<td>Cao Bằng</td>
<td>6700.26</td>
<td>79</td>
<td>0.653</td>
</tr>
<tr>
<td>Hà Giang</td>
<td>Hà Giang</td>
<td>7929.48</td>
<td>108</td>
<td>0.586</td>
</tr>
<tr>
<td>Lạng Sơn</td>
<td>Lạng Sơn</td>
<td>8310.09</td>
<td>94</td>
<td>0.707</td>
</tr>
<tr>
<td>Phú Thọ</td>
<td>Việt Trì</td>
<td>3534.56</td>
<td>414</td>
<td>0.715</td>
</tr>
<tr>
<td>Quảng Ninh</td>
<td>Hạ Long</td>
<td>6178.21</td>
<td>214</td>
<td>0.784</td>
</tr>
<tr>
<td>Thái Nguyên</td>
<td>Thái Nguyên</td>
<td>3526.64</td>
<td>365</td>
<td>0.741</td>
</tr>
<tr>
<td>Tuyên Quang</td>
<td>Tuyên Quang</td>
<td>5867.9</td>
<td>134</td>
<td>0.699</td>
</tr>
<tr>
<td>Lào Cai</td>
<td>Lào Cai</td>
<td>6364.03</td>
<td>115</td>
<td>0.67</td>
</tr>
</tbody>
</table>

We can also use the `from` and `to` arguments for conversions between other encodings.

For example, to convert VISCII-encoded data to Unicode, use:

```r
df_viscii <- vn_samples$VISCII
dt::datatable(decodeVN(df_viscii, from = "VISCII"))
```

## Table of Provinces in Vietnam

<table>
<thead>
<tr>
<th>Province_city</th>
<th>Administrative_center</th>
<th>Area_km2</th>
<th>Density_perkm2</th>
<th>HDI_2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bắc Giang</td>
<td>Bắc Giang</td>
<td>3895.59</td>
<td>463</td>
<td>0.711</td>
</tr>
<tr>
<td>Bắc Kạn</td>
<td>Bắc Kạn</td>
<td>4859.96</td>
<td>65</td>
<td>0.685</td>
</tr>
<tr>
<td>Cao Bằng</td>
<td>Cao Bằng</td>
<td>6700.26</td>
<td>79</td>
<td>0.653</td>
</tr>
<tr>
<td>Hà Giang</td>
<td>Hà Giang</td>
<td>7929.48</td>
<td>108</td>
<td>0.586</td>
</tr>
<tr>
<td>Lạng Sơn</td>
<td>Lạng Sơn</td>
<td>8310.09</td>
<td>94</td>
<td>0.707</td>
</tr>
<tr>
<td>Phú Thọ</td>
<td>Việt Trì</td>
<td>3534.56</td>
<td>414</td>
<td>0.715</td>
</tr>
<tr>
<td>Quảng Ninh</td>
<td>Hạ Long</td>
<td>6178.21</td>
<td>214</td>
<td>0.784</td>
</tr>
<tr>
<td>Thái Nguyên</td>
<td>Thái Nguyên</td>
<td>3526.64</td>
<td>365</td>
<td>0.741</td>
</tr>
<tr>
<td>Tuyên Quang</td>
<td>Tuyên Quang</td>
<td>5867.9</td>
<td>134</td>
<td>0.699</td>
</tr>
<tr>
<td>Lào Cai</td>
<td>Lào Cai</td>
<td>6364.03</td>
<td>115</td>
<td>0.67</td>
</tr>
</tbody>
</table>
The same thing without diacritics:

```r
dt::datatable(decodeVN(df_viscki, from = "VISCII", diacritics = FALSE))
```

Spatial data

Since version 0.4.0, decodeVN() works directly on sf objects and preserves the geometry column. The workflow is identical to data frames shown above.

```r
library(sf)
sf_object <- st_read(...)sf_object_decoded <- decodeVN(sf_object)
```