UTF-8 Guide

Instructions for encoding statistical file(s) and text file(s) as UTF-8. Supplement to the 'User's Guide to ASTA' (Aflevering af Statistikfiler Til Arkiv – translation: Submission of Statistical data to the Archives).

The National Archives, July 2020

Version 2.0

Contents

0. Reading the UTF-8 guide

Public authorities, including research institutions, submit data to the National Archives in the form of information packages. Requirements for these submissions are described in the Danish National Archives' Executive Order on Information Packages. One of the requirements is that the data submitted must be encoded as UTF-8.

The UTF-8 guide is a technical guide that explains what UTF-8 encoding is, how to check whether the encoding is UTF-8, and how to encode characters as UTF-8 in SAS, Stata, SPSS, and R statistical programs or in a text editor.

A. The guide's target audience and application

The UTF-8 guide is aimed at those who produce information packages with data extractions from statistical file(s) for submission to the Archives.

B. Other guides

In addition to the UTF-8 guide, the Danish National Archives has created other guides that are relevant to the production and submission of information packages:

- Quick guide - for the production and testing of an information package with ASTA
- Guide to Schedule 9 in the Executive Order on Information Packages
- User guide to ASTA
- Guide to creating an information package with data from spreadsheets or csv files
- Guide to the program Skab archiveIndex
- Guide to the program Skab contextDocumentationIndex
- Guide to converting documents to TIFF format
- Sample information package with statistical data FD.18005

All guide materials can be found on the National Archives’ homepage www.sa.dk.

C. Law and legislation

Information about related legislation can be found on the National Archives' (Rigsarkivet) homepage www.sa.dk.

D. Definitions

Information packages with data from statistical file(s) in general consist~~s~~ of context documents – which should be submitted in archival formats designated by the Archives, the extracts of data and metadata from the statistical files in the submission, and two index files in xml format containing metadata about the submitted data and context documents.

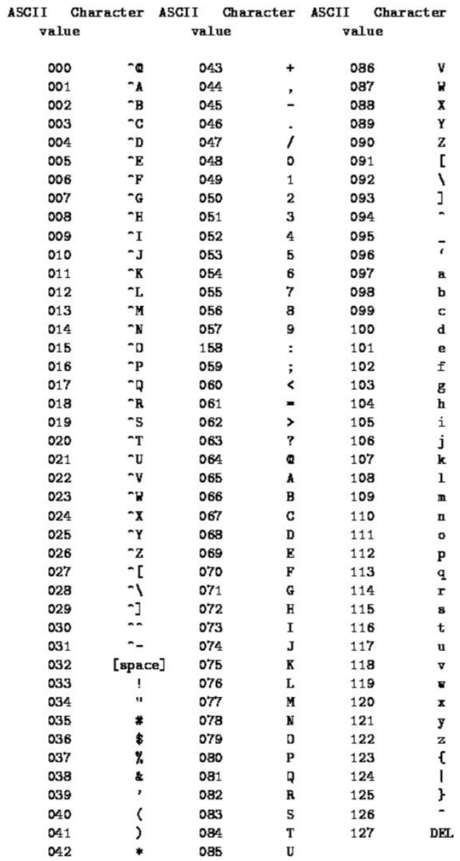
1. What is UTF-8 encoding?

Words and phrases in a text are made of characters, which are the letters that can be seen, e.g. a, b, c, å, and @. The computer uses its own ’language’, which represents all characters, numbers, and letters in the form of bytes, which are composed of 8 bits. Each byte can be represented with a value between 0-255.

When a text file is saved, a number is given to each letter (e.g. ‘A’ is given value 65, ‘B’ is given-value 66 and so forth). These numbers are then saved on the computer’s hard drive.

When the text is read again, the program attempts to translate and show these values as characters on the screen (e.g. Value 65 will show the letter ‘A’).

The oldest and most common mapping between values and characters is called ASCII. The complete ASCII mapping table is shown in Figure 1.1.



ASCII value	Character	ASCII value	Character	ASCII value	Character
000	^@	043	+	086	V
001	^A	044	,	087	W
002	^B	045	-	088	X
003	^C	046	.	089	Y
004	^D	047	/	090	Z
005	^E	048	0	091	[
006	^F	049	1	092	\
007	^G	050	2	093	]
008	^H	051	3	094	^
009	^I	052	4	095	_
010	^J	053	5	096	`
011	^K	054	6	097	a
012	^L	055	7	098	b
013	^M	056	8	099	c
014	^N	057	9	100	d
015	^O	158	:	101	e
016	^P	059	;	102	f
017	^Q	060	<	103	g
018	^R	061	=	104	h
019	^S	062	>	105	i
020	^T	063	?	106	j
021	^U	064	@	107	k
022	^V	065	A	108	l
023	^W	066	B	109	m
024	^X	067	C	110	n
025	^Y	068	D	111	o
026	^Z	069	E	112	p
027	^[	070	F	113	q
028	^\	071	G	114	r
029	^]	072	H	115	s
030	^^	073	I	116	t
031	^-	074	J	117	u
032	[space]	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

Figure 1.1 ASCII table with the translation between characters and decimal values (ASCII values)

In the early ages of computers, reading national characters such as Æ, Ø, and Å, for example, was a dream. One had to get by using the American characters. This changed when other countries started to exchange the less used characters in the ASCII table to show other characters, such as Æ, Ø, and Å. One of these new versions of the ASCII table was, for instance, Code page 865 (Nordic languages). The problem was (and still is) that one did not know if an ASCII text file was saved in one language or another when the file was shown and one needed to try both forms.

There were several attempts to solve this problem through the years by creating other text formats, such as ANSI, EBCDIC, and Unicode. The latest and best solution is called UTF-8, which can use all languages because it uses 1-4 bytes, hence it can represent a much larger number of characters (among them Æ, Ø, and Å).

2. What are the consequences if data is not encoded as UTF-8-encoding?

Regarding the submission of statistical file(s) in the form of an information package to the Archives, it is the responsibility of the submitting institution to extract data and metadata from statistical file(s) for the information package. The institution that produces the information package must ensure that all characters are correctly encoded as UTF-8 before extraction.

In a dataset originated from one of the newer versions of SAS, SPSS, Stata, or RStudio, the encoding in the statistical file is most likely encoded as UTF-8 because the newer versions of these statistical programs use this as the default encoding. If the statistical program setup is not Unicode, you can change this setup in 'Preference'/'Options' in the statistical program and then save the file(s) as Unicode before extracting data for the information package via ASTA, ensuring that all characters are displayed correctly.

If your dataset originates from another program or has another encoding (e.g. ANSI) and is imported into a statistical program that uses Unicode as a default, this may cause some characters to appear incorrectly (e.g. a word such as 'stå' may appear as 'st□’), as the transformation may affect the characters. If this happens, it is important to correct the wrong characters so others can use, read, and understand the dataset in the future.

The National Archives′ tool that can be used to test the information package (ASTA) before submission to the archives does not automatically test for the encoding in the data and metadata file. However, The National Archives visually tests all submitted data and metadata files after submission. If there are invalid characters in the submitted material that are not UTF-8 characters, the submitting institution will be notified, and some adjustments will be required in the data file. This will usually require a new data extraction and resubmission.

Therefore, it is important that you visually check your extracted data in the information package to make sure that all characters are displayed correctly and can be understood. Read more about this in sections 5 and 6.

3. How do you read and change the statistical file encoding in statistical programs?

Make sure to check that the encoding of the statistical file(s) is UTF-8 before extracting data for the information package.

Each statistical program has its own syntax to examine the encoding in a dataset and to change it to a different encoding. Below you will find procedures and syntaxes to do that for SAS, Stata, and SPSS. Using these, you can make sure the data is encoded as UTF-8-encoding~~s~~.

A. SAS - encoding syntax and procedures

Examine SAS file encoding

To identify a dataset’s encoding in a SAS file, follow these steps recommended by SAS[1]:

- Run the following SAS syntax to determine the encoding for a data set in SAS. You only need to replace libref.data_set_name with your library's name and file name (for example, "mylib.mydata").

SAS syntax

```
%let dsn=libref.data_set_name;
%let dsid=%sysfunc(open(&dsn,i));
%put &dsn ENCODING is: %sysfunc(attrc(&dsid,encoding));
```

Example

```
%let dsn=dgi.customerdaga;
%let dsid=%sysfunc(open(&dsn,i));
%put &dsn ENCODING is: %sysfunc(attrc(&dsid,encoding));
```

Another way to find the SAS file encoding is to run a "proc contents" that displays the file encoding in the output. In this way:

SAS syntax

```
PROC CONTENTS <option-1 <...option-n>>;
run;
```

Example

```
PROC CONTENTS  data=dgi.customerdata;
```

Change the SAS file encoding to UTF-8:

To change the encoding of a SAS file that has not been previously defined as UTF-8, the following syntax can be used[2]. Note that you must replace the following:

1) libref and its location
2) Specify the location in which you want to save the new UTF-8 file (Second line of the syntax)
3) Enter the desired dataset name for the new UTF-8 file (Fourth line of the syntax)

SAS syntax

```
libname inlib libref 'c:\xxxx';
libname outlib 'c:\yyy' outencoding='UTF-8';
proc copy noclone in=inlib out=outlib;
select dataset_name;
run;
```

[1] http://support.sas.com/kb/14/290.html

[2] http://support.sas.com/kb/15/597.html

```
Example
libname inlib dgi 'c:\temp';
libname outlib 'c:\temp\out' outencoding='UTF-8';
proc copy noclone in=inlib out=outlib;
select customerdata;
run;
```

B. SPSS - syntax and encoding procedures

Encoding in different versions of SPSS:

As described by IBM[3], SPSS:

- Up to version 15, all encoding in SPSS was based on code pages.
- From versions 16 to 20, Unicode (like UTF-8) is also supported. UTF-8 is called **"Unicode mode"** in SPSS 16. Note that UTF-8 encoding is supported in both datasets and syntax files.
- From SPSS version 21 and subsequent versions, the program asks whether "Unicode mode" should be used when the program starts.

Examine the SPSS file encoding

The following syntax can be run in SPSS to identify if SPSS setup is in Unicode:

```
SPSS syntax
SHOW UNICODE
```

Examine and change the SPSS file encoding

To identify and change an encoding in SPSS, click "Edit" and select 'Options' from the menu in SPSS before opening the dataset you want to examine (see Figure 3.1).

This opens a window showing all options. Select the tab ‘Language’ (see Figure 3.2). Mark "*Unicode (universal encoding)*" to select UTF-8 as SPSS default encoding for data and syntax. Click *'OK'*.

You can check if the change has been applied by looking at the bottom right of the SPSS program window, which should display '*Unicode: ON'* (see Figure 3.3).

If the text '*Unicode: OFF*' is displayed inside the red circle shown in Figure 3.3, the field “Character Encoding for Data and Syntax” in the Language tab under Options is marked as '*Locale's writing systems*' and not '*Unicode (universal encoding)*’.

[3] https://www.spss-tutorials.com/spss-unicode-mode/

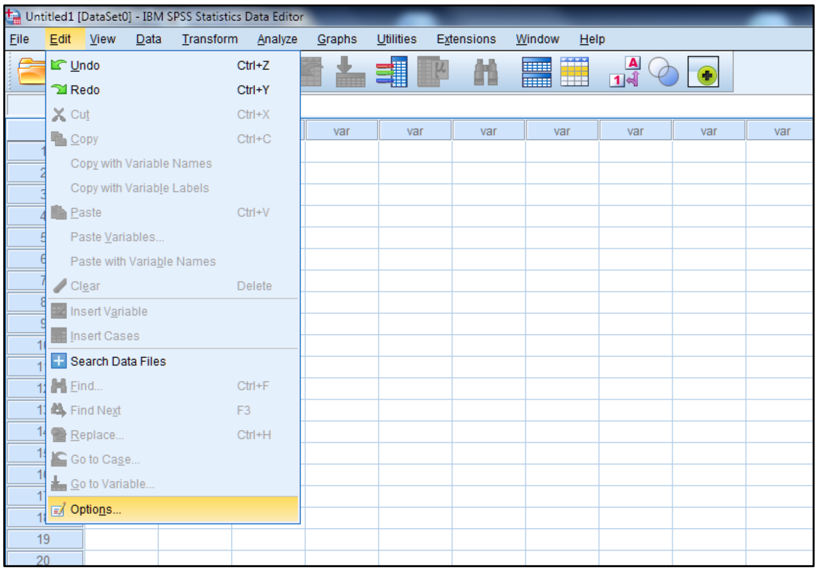
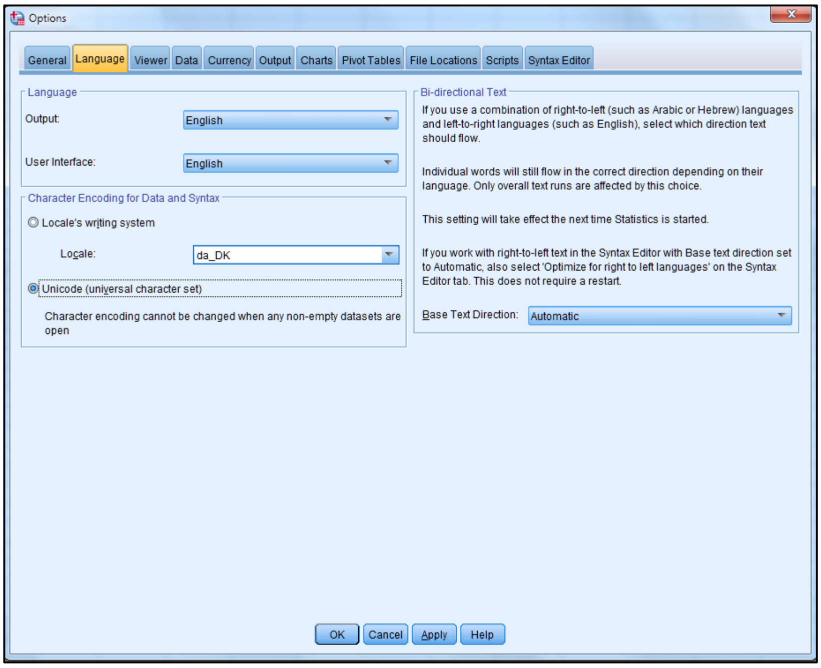


***Figure 3.1**: Select Edit > Options in SPSS*



Language

Output: English

User Interface: English

Character Encoding for Data and Syntax

○ Locale's writing system

Locale: da_DK

● Unicode (universal character set)

Character encoding cannot be changed when any non-empty datasets are open

Bi-directional Text

If you use a combination of right-to-left (such as Arabic or Hebrew) languages and left-to-right languages (such as English), select which direction text should flow.

Individual words will still flow in the correct direction depending on their language. Only overall text runs are affected by this choice.

This setting will take effect the next time Statistics is started.

If you work with right-to-left text in the Syntax Editor with Base text direction set to Automatic, also select 'Optimize for right to left languages' on the Syntax Editor tab. This does not require a restart.

Base Text Direction: Automatic

***Figure 3.2** "Language tab" under Options in SPSS*

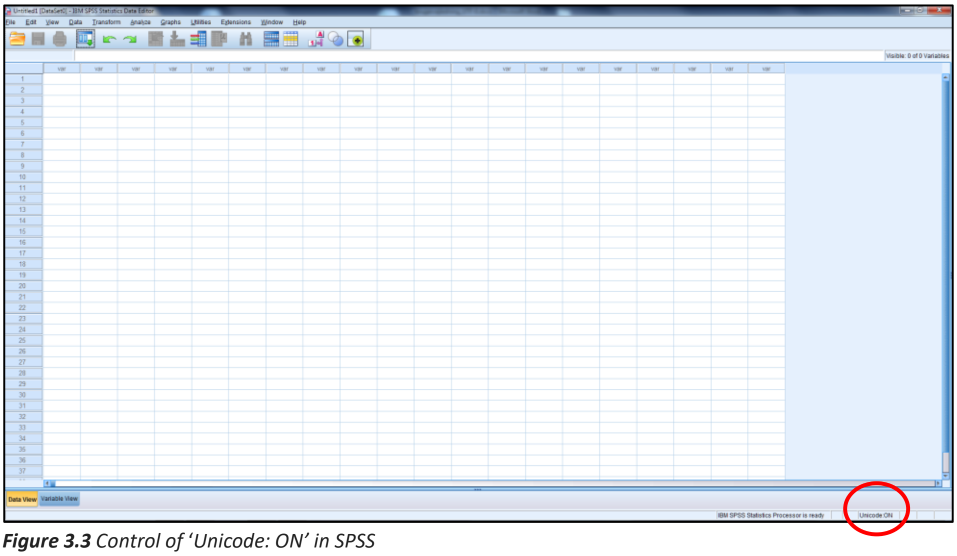
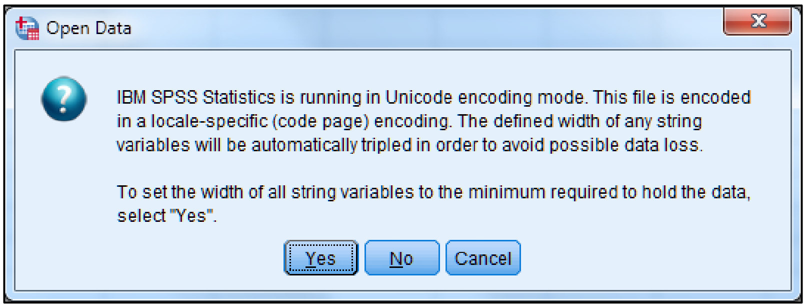

Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Graphs Utilities Extensions Window Help
Visible: 0 of 0 Variables
Data View
Variable View
IBM SPSS Statistics Processor is ready
Unicode:ON

Figure 3.3 *Control of 'Unicode: ON' in SPSS*

Change the SPSS encoding to UTF-8:

If you open a file that is not encoded as Unicode in SPSS, a pop-up window appears with the following message once you enable Unicode:



IBM SPSS Statistics is running in Unicode encoding mode. This file is encoded in a locale-specific (code page) encoding. The defined width of any string variables will be automatically tripled in order to avoid possible data loss.

To set the width of all string variables to the minimum required to hold the data, select "Yes".

Figure 3.4 *Pop-up window in SPSS*

You must select 'Yes' to optimize the number of bytes. The file is now saved in the UTF-8 format.

C. Stata - syntax and encoding procedures

Encoding in different versions of Stata:

As described by Stata[4]:

- Stata 13 and previous versions use ASCII by default for encoding.
- In Stata 14 and subsequent versions, UTF-8 is the default encoding for datasets, do-files, ado-files, and 'help' files.

[4] https://www.stata.com/manuals/dunicodeencoding.pdf

Examine Stata file encoding

To analyze the Stata file encoding in Stata, run the following syntax:

Stata syntax

```
unicode analyze datasetname.dta
```

Example

```
unicode analyze customerdata.dta
```

Change Stata file encoding to UTF-8

Stata can also translate files from 'extended ASCII' encoding to Unicode (UTF-8). First, define what encoding you want to translate the file to. To do this, run the following syntax:

Stata syntax

```
unicode encoding set encodingname
```

Example

```
unicode encoding set unicode
```

Next, you can use the following syntax to transform the Stata file into Unicode:

Stata syntax

```
unicode translate myfile.dta
```

Example

```
Unicode translate customerdata.dta
```

If you know the encoding of the source file (srcencoding) and the encoding you want to transform it into (dstencoding), you can apply the following syntax:

Stata syntax

```
unicode convertfile srcfilename destfilename , options
```

Example

```
unicode convertfile ” C:\Temp\customerdata.txt” ”
C:\Temp\customerdata2.txt”, srcencoding(ANSI1251)
dstencoding(UNICODE)
```

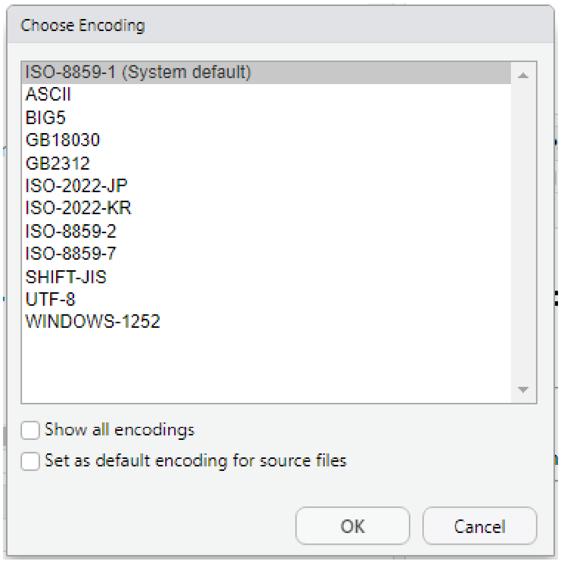
D. R and RStudio – syntax and encoding procedures

Encoding in RStudio[5]

Starting with RStudio version 0.93, the encodings of all Unicode-characters are supported through a 'platform native'. In other words, the program gives you the option to read and write files with the help of any type of encoding available in your system. You can do it by:

- Choose the encoding to read the files by clicking on 'File'> 'Reopen with encoding', which will re-read the chosen file from the disc with the new encoding.
- Save the opened file with a given encoding by clicking on 'File' > 'Save with encoding'.

Both commands *Reopen with encoding* and *Save with encoding* show the following dialog box, where you select the desired encoding to be read or to be saved on file. If it is a submission to the National Archives, select UTF-8 from the list of encodings.



Choose Encoding

ISO-8859-1 (System default)
ASCII
BIG5
GB18030
GB2312
ISO-2022-JP
ISO-2022-KR
ISO-8859-2
ISO-8859-7
SHIFT-JIS
UTF-8
WINDOWS-1252

□ Show all encodings
□ Set as default encoding for source files

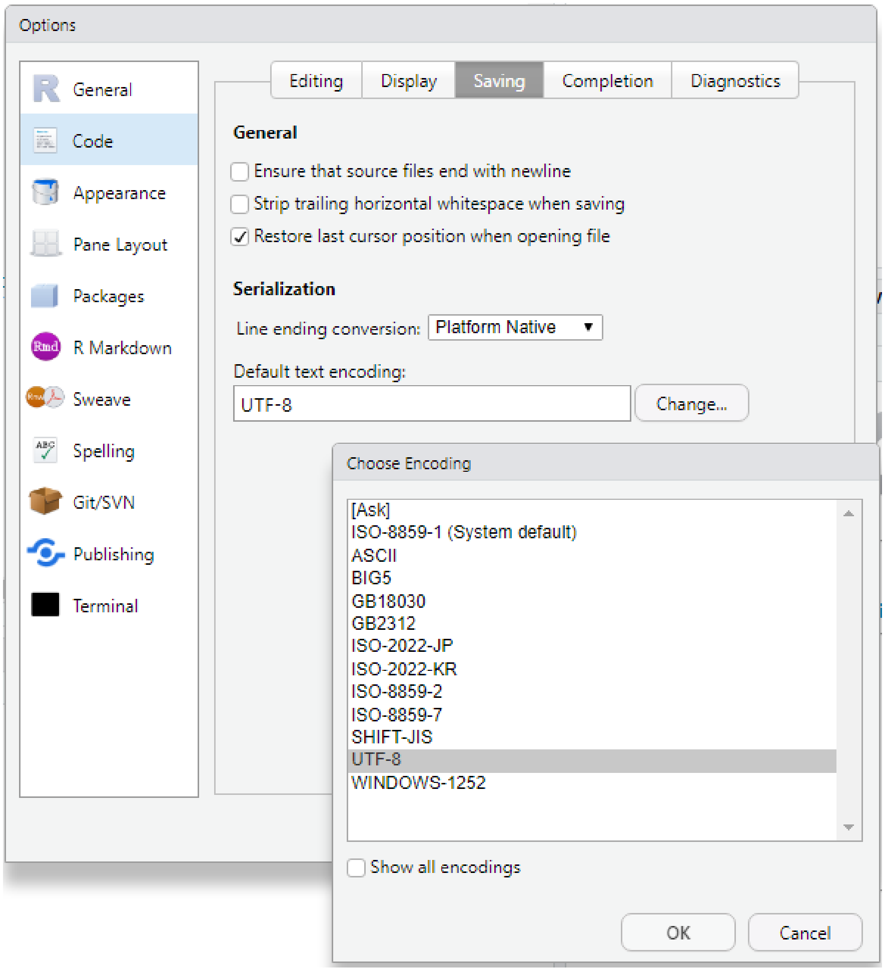
Changing the encoding in RStudio permanently

If you would like to change the standard encoding of your RStudio permanently, you can do the following:

1) Click on 'Tools' > 'General Options';
2) Select 'Code' in the menu to the left;
3) Select the folder 'Saving' from the menu on the top (see below);
4) And click on 'Change'.

[5] https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

When you click on ’Change’, a popup window appears, which gives you the option to choose either another encoding or "[ASK]" to be asked which encoding to use each time. In this way:



General

□ Ensure that source files end with newline

□ Strip trailing horizontal whitespace when saving

☑ Restore last cursor position when opening file

Serialization

Line ending conversion: Platform Native ▼

Default text encoding:

UTF-8

[Ask]
ISO-8859-1 (System default)
ASCII
BIG5
GB18030
GB2312
ISO-2022-JP
ISO-2022-KR
ISO-8859-2
ISO-8859-7
SHIFT-JIS
UTF-8
WINDOWS-1252

□ Show all encodings

Encoding in R

Attempting to change the encoding to UTF-8 in R is a complex task because the procedures vary according to the operative system of your computer. For further information on encoding in R, you can read more on the topic and its complexity in the article "Escaping from the character encoding hell in R on Windows" here: https://dss.iq.harvard.edu/blog/escaping-character-encoding-hell-r-windows

4. How to check character sets in a text file

When extracting data and metadata from the statistics file to ~~a~~ .csv and .txt files that comply with the required information package format for data files and metadata files, you must also visually verify that all characters are read correctly and are UTF-8 characters.

The extracted data file (e.g. table1.csv) may contain incorrect characters if: The statistical file from which the extract was made was not in UTF-8 (Unicode) before extraction; or if its original data file contains non-valid UTF-8 characters.

You can inspect .csv and .txt files for incorrect character information package as follows:

- Find the location of your information package. For example, in a folder called FD. 12345
- Locate the data file you want to check for incorrect UTF-8 characters (e.g. table1.csv) by clicking down the folder structure: FD.12345 > Data > table1 > table1.csv
- Right-click ‘12345.csv’ and select 'open with' from the pop-up list. Choose to open the file with a text editor, such as *Notepad* or Notepad*++.*

  NOTE: Do not double-click the file to open it, as this may trigger an automatic opening with Excel. Excel may automatically try to identify character sets and data formats in the file. This may lead to the data being loaded incorrectly.

- Inspect the contents of the data file by looking for strange-looking characters. Search for characters such as æ, ø, and å, as these often appear incorrectly if the character set is not UTF-8.
- If you find any incorrect characters, correct them in your original data file. After correction, a new extract must be made (e.g. with the ASTA program), and the extracted data file must once again be visually tested for readability and non-valid UTF-8 characters.

NOTE: A text file with æ, ø, and å encoded in ANSI will show the characters æ, ø, and å correct in a text editor such as Notepad. The above method cannot guarantee that the file is encoded as UTF-8 with correct UTF-8 characters. However, it can show you if the text file has characters that are not encoded in a way the text editor uses to show the characters in the text file.

If you want to identify the text file’s encoding, you can ~~see~~ find the hex-value for the characters in a binary file editor.

5. How to read UTF-8 hex-values in a text file

If you want to know whether or not a character is a valid UTF-8 character, you can examine the binary content of a character in the text file.

For this purpose, you must use a binary file editor (e.g. HxD file). This Hex editor displays the default numeric representation of a character in a binary format in the form of a hex value. Hexadecimal UTF-8 values for æ, ø, å, Æ, Ø, and Å are presented in Figure 5.1.

Unfortunately, given that the program HxD cannot show a UTF-8 encoding character representation, you cannot see the character correctly (under ’Decoded text’). On the other hand, you can see the character as it is displayed in ANSI, ASCII, Macintosh, or EBCDIC. You can still see the file’s correct hexadecimal representation of the character (see figure 5.3). When you mark a character on the text to the right (text shown in ANSI format), the equivalent hexadecimal representation of the binary number of the marked character is shown. Mark special characters such as æ, ø, and å to make sure that those special characters are shown with the correct UTF-8 encoding.

æ = **C3 E6** (shown in HxD as Ã¦)

ø = **C3 B8** (shown in HxD as Ã¸)

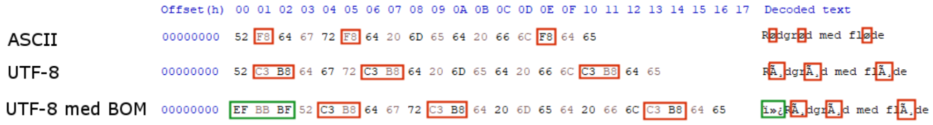
å = **C3 E5** (shown in HxD as Ã¥)

Æ = **C3 86** (shown in HxD as Ã†)

Ø = **C3 98** (shown in HxD as Ã˜)

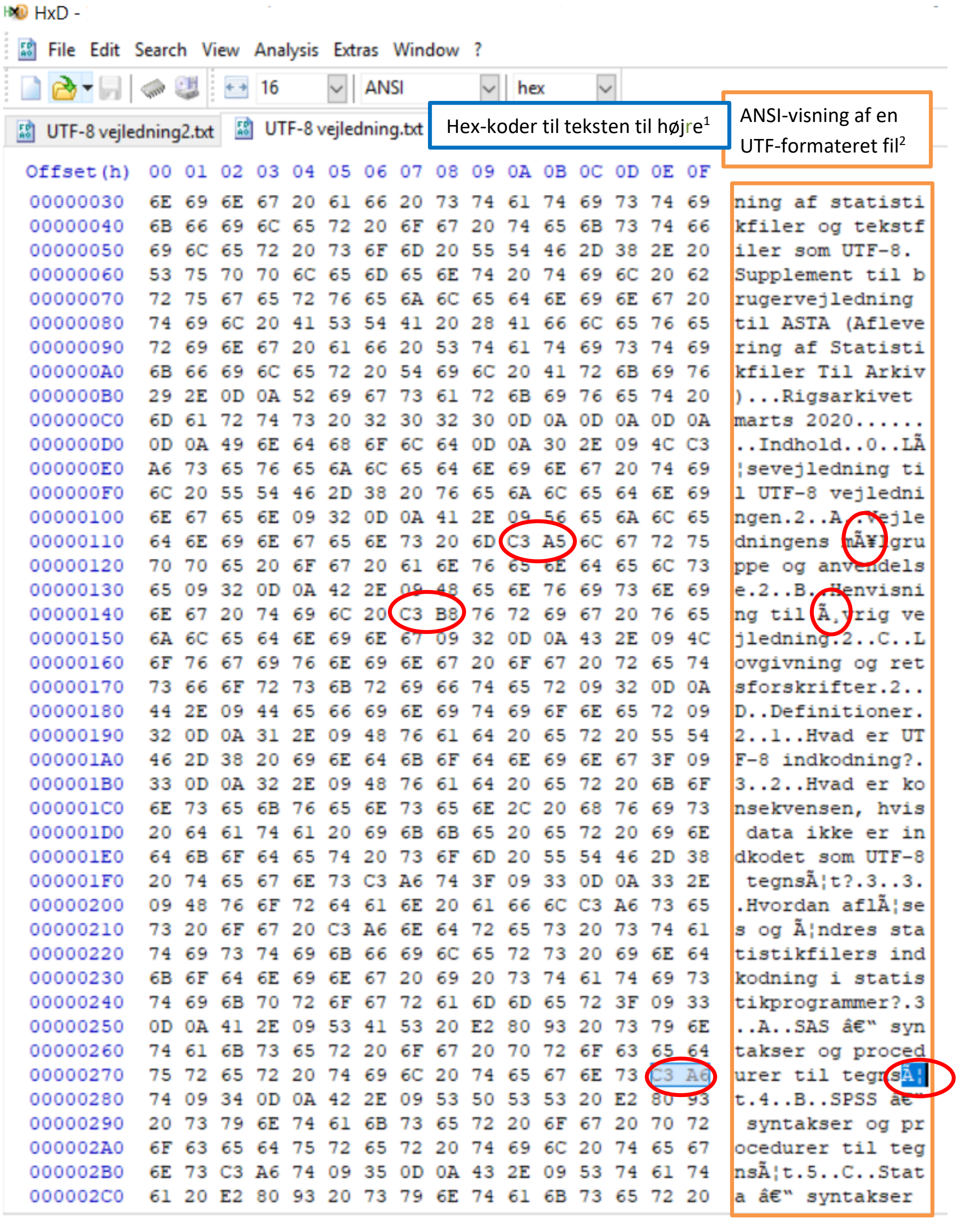
Å = **C3 85** (shown in HxD as Ã…)

Figure 5.1 Correct *hexadecimal representation of æ, ø, å, Æ, Ø, and Å in a UTF-8 encoded text file*



```
               Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17  Decoded text
ASCII          00000000  52 F8 64 67 72 F8 64 20 6D 65 64 20 66 6C F8 64 65                       Rødgrød med fløde
UTF-8          00000000  52 C3 B8 64 67 72 C3 B8 64 20 6D 65 64 20 66 6C C3 B8 64 65              RÃ¸dgrÃ¸d med flÃ¸de
UTF-8 med BOM  00000000  EF BB BF 52 C3 B8 64 67 72 C3 B8 64 20 6D 65 64 20 66 6C C3 B8 64 65     ï»¿RÃ¸dgrÃ¸d med flÃ¸de
```

Figure 5.2 *Comparison between hexadecimal representation of ASCII and UTF-8 characters*



Hex-koder til teksten til højre[1]

ANSI-visning af en UTF-formateret fil[2]

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000030  6E 69 6E 67 20 61 66 20 73 74 61 74 69 73 74 69  ning af statisti
00000040  6B 66 69 6C 65 72 20 6F 67 20 74 65 6B 73 74 66  kfiler og tekstf
00000050  69 6C 65 72 20 73 6F 6D 20 55 54 46 2D 38 2E 20  iler som UTF-8. 
00000060  53 75 70 70 6C 65 6D 65 6E 74 20 74 69 6C 20 62  Supplement til b
00000070  72 75 67 65 72 76 65 6A 6C 65 64 6E 69 6E 67 20  rugervejledning 
00000080  74 69 6C 20 41 53 54 41 20 28 41 66 6C 65 76 65  til ASTA (Afleve
00000090  72 69 6E 67 20 61 66 20 53 74 61 74 69 73 74 69  ring af Statisti
000000A0  6B 66 69 6C 65 72 20 54 69 6C 20 41 72 6B 69 76  kfiler Til Arkiv
000000B0  29 2E 0D 0A 52 69 67 73 61 72 6B 69 76 65 74 20  )...Rigsarkivet 
000000C0  6D 61 72 74 73 20 32 30 32 30 0D 0A 0D 0A 0D 0A  marts 2020......
000000D0  0D 0A 49 6E 64 68 6F 6C 64 0D 0A 30 2E 09 4C C3  ..Indhold..0..LÃ
000000E0  A6 73 65 76 65 6A 6C 65 64 6E 69 6E 67 20 74 69  ¦sevejledning ti
000000F0  6C 20 55 54 46 2D 38 20 76 65 6A 6C 65 64 6E 69  l UTF-8 vejledni
00000100  6E 67 65 6E 09 32 0D 0A 41 2E 09 56 65 6A 6C 65  ngen.2..A..Vejle
00000110  64 6E 69 6E 67 65 6E 73 20 6D C3 A5 6C 67 72 75  dningens mÃ¥lgru
00000120  70 70 65 20 6F 67 20 61 6E 76 65 6E 64 65 6C 73  ppe og anvendels
00000130  65 09 32 0D 0A 42 2E 09 48 65 6E 76 69 73 6E 69  e.2..B..Henvisni
00000140  6E 67 20 74 69 6C 20 C3 B8 76 72 69 67 20 76 65  ng til Ã¸vrig ve
00000150  6A 6C 65 64 6E 69 6E 67 09 32 0D 0A 43 2E 09 4C  jledning.2..C..L
00000160  6F 76 67 69 76 6E 69 6E 67 20 6F 67 20 72 65 74  ovgivning og ret
00000170  73 66 6F 72 73 6B 72 69 66 74 65 72 09 32 0D 0A  sforskrifter.2..
00000180  44 2E 09 44 65 66 69 6E 69 74 69 6F 6E 65 72 09  D..Definitioner.
00000190  32 0D 0A 31 2E 09 48 76 61 64 20 65 72 20 55 54  2..1..Hvad er UT
000001A0  46 2D 38 20 69 6E 64 6B 6F 64 6E 69 6E 67 3F 09  F-8 indkodning?.
000001B0  33 0D 0A 32 2E 09 48 76 61 64 20 65 72 20 6B 6F  3..2..Hvad er ko
000001C0  6E 73 65 6B 76 65 6E 73 65 6E 2C 20 68 76 69 73  nsekvensen, hvis
000001D0  20 64 61 74 61 20 69 6B 6B 65 20 65 72 20 69 6E   data ikke er in
000001E0  64 6B 6F 64 65 74 20 73 6F 6D 20 55 54 46 2D 38  dkodet som UTF-8
000001F0  20 74 65 67 6E 73 C3 A6 74 3F 09 33 0D 0A 33 2E   tegnsÃ¦t?.3..3.
00000200  09 48 76 6F 72 64 61 6E 20 61 66 6C C3 A6 73 65  .Hvordan aflÃ¦se
00000210  73 20 6F 67 20 C3 A6 6E 64 72 65 73 20 73 74 61  s og Ã¦ndres sta
00000220  74 69 73 74 69 6B 66 69 6C 65 72 73 20 69 6E 64  tistikfilers ind
00000230  6B 6F 64 6E 69 6E 67 20 69 20 73 74 61 74 69 73  kodning i statis
00000240  74 69 6B 70 72 6F 67 72 61 6D 6D 65 72 3F 09 33  tikprogrammer?.3
00000250  0D 0A 41 2E 09 53 41 53 20 E2 80 93 20 73 79 6E  ..A..SAS â€“ syn
00000260  74 61 6B 73 65 72 20 6F 67 20 70 72 6F 63 65 64  takser og proced
00000270  75 72 65 72 20 74 69 6C 20 74 65 67 6E 73 C3 A6  urer til tegnsÃ¦
00000280  74 09 34 0D 0A 42 2E 09 53 50 53 53 20 E2 80 93  t.4..B..SPSS â€“
00000290  20 73 79 6E 74 61 6B 73 65 72 20 6F 67 20 70 72   syntakser og pr
000002A0  6F 63 65 64 75 72 65 72 20 74 69 6C 20 74 65 67  ocedurer til teg
000002B0  6E 73 C3 A6 74 09 35 0D 0A 43 2E 09 53 74 61 74  nsÃ¦t.5..C..Stat
000002C0  61 20 E2 80 93 20 73 79 6E 74 61 6B 73 65 72 20  a â€“ syntakser
```

***Figure 5.3** How a hex editor displays a UTF-8 encoded character and its binary value*

[1] *Translation:"* Hex codes for the text on the right"

[2] Translation: "ANSI display of a UTF-formatted file"

6. Technical explanation of UTF-8 encoding, BOM, and character representation

E. More on the UTF-8 encoding

UTF-8 is an abbreviation for ”Unicode Transformation Format”. UTF-8 is one of the three standard encodings of a character that uses the Unicode representation as computer text (the others being UTF-16 and UTF-32). UTF-8 uses an algorithm to decode the data between a binary form, (e.g. ‘01100001’), which is used by computers, and a character (e.g. ‘a’), which is used by people when reading. '8' in UTF-8 means that the encoding uses 8-bit blocks (one byte) to represent a character. UTF-8 can be used in all languages because it uses 1-4 bytes to represent the characters. Hence, it can represent a much larger number of characters (among them Æ, Ø, and Å). ASCII and ANSI use just one byte per character.

Because UTF-8 is an effective way to store Unicode text and it supports many different languages, it has become the most commonly used Unicode encoding today.

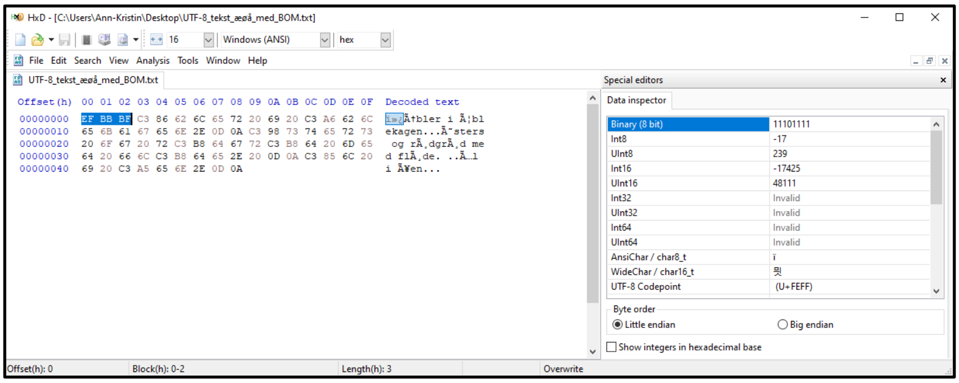
When UTF-8 was defined, the hope was that UTF-8 would be backward compliant with ASCII (see section 1). Therefore, the 127 U.S. alphabet letters and numbers in the UTF-8 table are identical to the ASCII table marks and take up only one byte of space. Other countries' national characters use two bytes (for example, the letter 'Æ' has been assigned the values 195 and 166). The letters a-z are represented in both ASCII, ANSI, and UTF-8 with a single byte of the same value. In other words, an ANSI file without æ, ø, and å is also a valid UTF-8 file.

When saving a text file, some programs give you the option to select the desired encoding for the character set in the file, e.g. UTF-8. Other times, the text file is automatically saved with the text program's default encoding/character set. It is also possible in some text editors to convert between different character sets, e.g. saving an ANSI encoded file with UTF-8 encoding. UTF-8 encoding and decoding of a text file is not always something you perform yourself, but something that must be supported within the programs you use to save (encoding) and display (decoding) the file. When programs translate the binary representations of the characters (e.g. 01100001) into legible characters (e.g. a), they will typically try to guess the correct encoding of the text file by looking for UTF-8 characters, trying to display the content correctly. This usually goes well, but it fails occasionally.

F. BOM (Byte Order Mark)

A UTF-8 file can include three specific byte values at its beginning with the hexadecimal values **EF BB BF** (see Figure 6.1). This is a so-called Byte Order Mark, also known as BOM. When a text file has this BOM, you can be reasonably sure that it is UTF-8 encoding. Unfortunately, the BOM mark is not required. Furthermore, it is also not possible to choose whether to add a BOM to the program when saving the file.

Because it is possible to copy a BOM into a text file in a binary editor, the presence of a BOM does not always mean that the file is encoded as UTF-8 unless there are also valid UTF-8 hexadecimal values in the file (see section 5).


HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]
16
Windows (ANSI)
hex
File Edit Search View Analysis Tools Window Help
UTF-8_tekst_æøå_med_BOM.txt
Special editors
Data inspector

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010   65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020   20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030   64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l 
00000040   69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Binary (8 bit)	11101111
Int8	-17
UInt8	239
Int16	-17425
UInt16	48111
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	ï
WideChar / char16_t	뻯
UTF-8 Codepoint	(U+FEFF)

Byte order
● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 0 Block(h): 0-2 Length(h): 3 Overwrite

Figure 6.1 *UTF-8 file with BOM (hexadecimal value: EF BB BF)*

G. Various representations of a character

The characters that compose a sentence, e.g. a, b, c, å, and @, exist in the computer in binary form, i.e. in the form of bytes. A byte in UTF-8 consists of 8 bits and each bit can have either a value of 0 or 1. For example, the binary value 01100001 represents the letter 'a' in both an ASCII, ANSI, and UTF-8 encoding. However, a character can also be represented by more than one byte, e.g. the letter 'æ' represented in the UTF-8 encoding by the 2 bytes 110000111 10100110.

Many encodings can translate computers' bytes into characters. Examples of different character encodings are ASCII, ANSI, EBCDIC, Unicode, and UTF-8. Depending on the selected encoding in a text editor, the computer translates (decodes) the bytes in the text file differently and displays the characters differently. If the text editor's choice of file encoding (for translating/decoding the binary values) does not match the encoding text file, visualization of the text will be corrupted. That is, you will not see the correct characters, e.g. å appears as □, or [, or another character.

The computer saves a word like 'hello' with 5 bytes (40 bits): 01001000 01100101 01101100 01101100 0110111. Because these binary numbers are often long and difficult to display in a table or program, they are often displayed in other forms, such as decimal, hexadecimal, or codepoints. In Table 6.1, you can see different representations of the letter 'a' in a text file encoded as UTF-8**.**

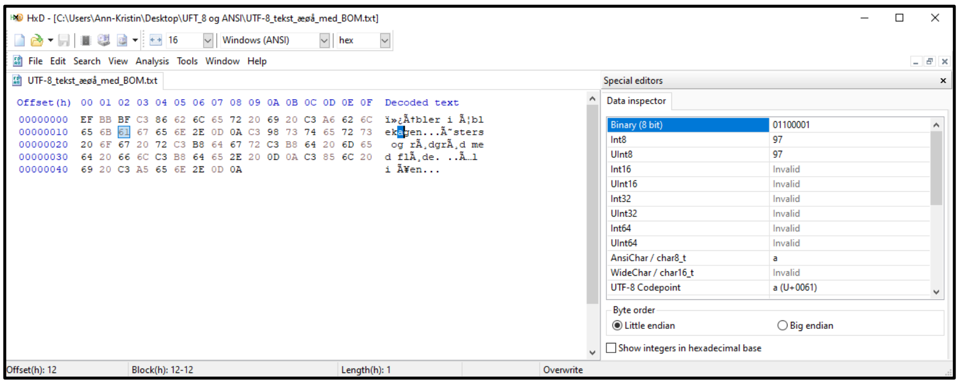
The binary number system is a 2-digit system consisting of the two numbers 0 and 1. A binary value can be converted to a decimal value (in the 10-digit system). The decimal representation of the binary value 01100001 for the sign 'a' is calculated as follows: 0x128 + 1x64 + 1x32 + 0x16 + 0x8 + 0x4 + 0x2 + 1x1 = 97.

Table 6.1. UTF-8 representations of a character

Character/ Letter	**Binary (UTF-8)** **128 64 32 16 8 4 2 1**	**Decimal (UTF-8)**	**Hexadecimal (UTF-8)**	**UTF-8 Codepoints**
A	01100001	97	61	U+0061
B	01100010	98	62	U+0062

Æ	11000011 10100110	195 166	C3 A6	U+00E6
Ø	11000011 10111000	195 184	C3 B8	U+00F8
Å	11000011 10100101	195 165	C3 A5	U+00E5
Æ	11000011 10000110	195 134	C3 86	U+00C6
Ø	11000011 10011000	195 152	C3 98	U+00D8
Å	11000011 10000101	195 133	C3 85	U+00C5
BOM			EF BB BF	U+FEFF

Several of these representations of the character can be seen in binary text editors, such as HxD. This is described in more detail in the previous section and illustrated in the following figures.



HxD - [C:\Users\Ann-Kristin\Desktop\UFT_8 og ANSI\UTF-8_tekst_æøå_med_BOM.txt]

16 | Windows (ANSI) | hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l 
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

Binary (8 bit)	01100001
Int8	97
UInt8	97
Int16	Invalid
UInt16	Invalid
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	a
WideChar / char16_t	Invalid
UTF-8 Codepoint	a (U+0061)

Byte order

● Little endian ○ Big endian

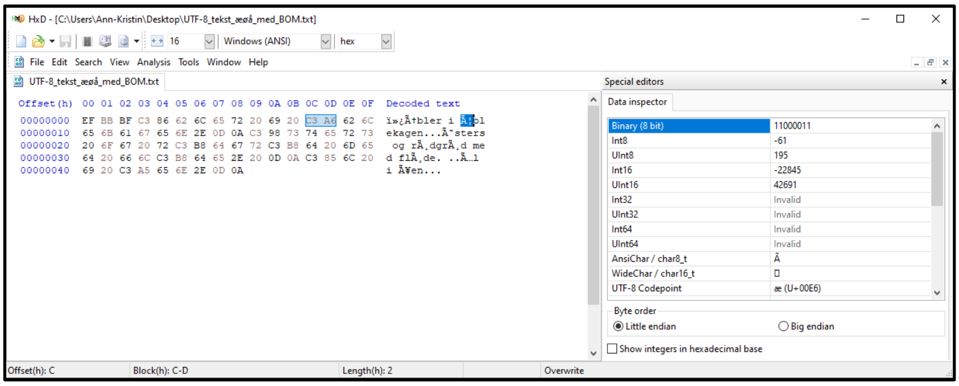
□ Show integers in hexadecimal base

Offset(h): 12 Block(h): 12-12 Length(h): 1 Overwrite

Figure 6.2 *UTF-8 file with a lowercase a (hexadecimal value: 61)*

NOTE that the character itself in UTF-8 text files in the figure below does not appear as ~~e~~ æ, ø, and å. The reason for this is that the HxD editor cannot decode/translate the text into UTF-8 view of the characters. Instead, what you see is how valid encoded UTF-8 hexadecimal values are translated/decoded into ANSI characters (decoded text).

NOTE that the binary value is displayed only for the first byte in two-byte characters. The same applies to the decimal value (UInt8). To read the binary and decimal values for each byte, each byte must be selected separately.



HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 · Windows (ANSI) · hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l 
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-22845
UInt16	42691
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	▯
UTF-8 Codepoint	æ (U+00E6)

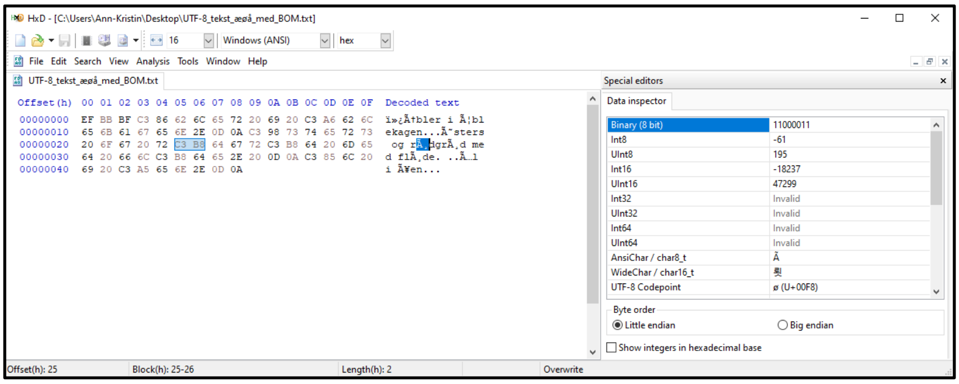
Byte order

● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): C | Block(h): C-D | Length(h): 2 | Overwrite

Figure 6.3 *UTF-8 file with lowercase æ (hexadecimal value: C3 A6)*



HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 · Windows (ANSI) · hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l 
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-18237
UInt16	47299
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	뗃
UTF-8 Codepoint	ø (U+00F8)

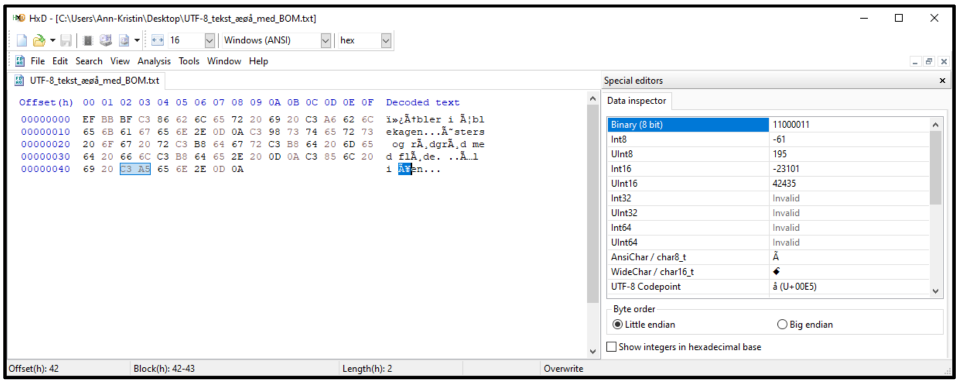
Byte order

● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 25 | Block(h): 25-26 | Length(h): 2 | Overwrite

Figure 6.4 *UTF-8 file with lowercase ø (hexadecimal value: C3 B8)*



HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 · Windows (ANSI) · hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l 
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-23101
UInt16	42435
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	◆
UTF-8 Codepoint	å (U+00E5)

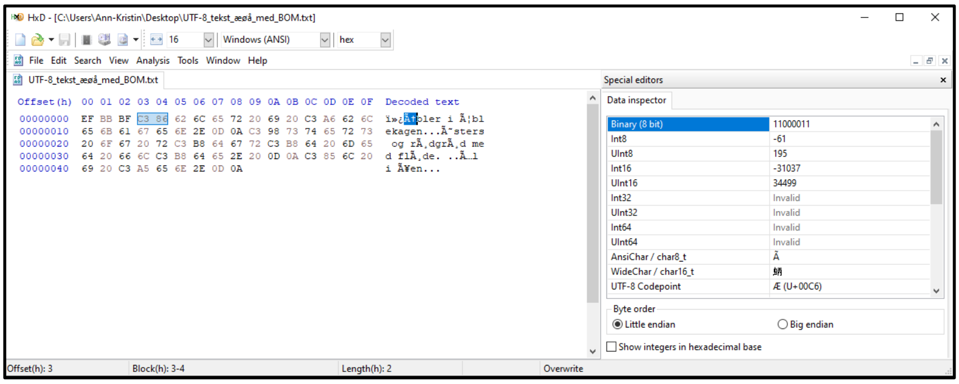
Byte order

● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 42 | Block(h): 42-43 | Length(h): 2 | Overwrite

Figure 6.5 *UTF-8 file with lowercase å (hexadecimal value: C3 A5)*


HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 | Windows (ANSI) | hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

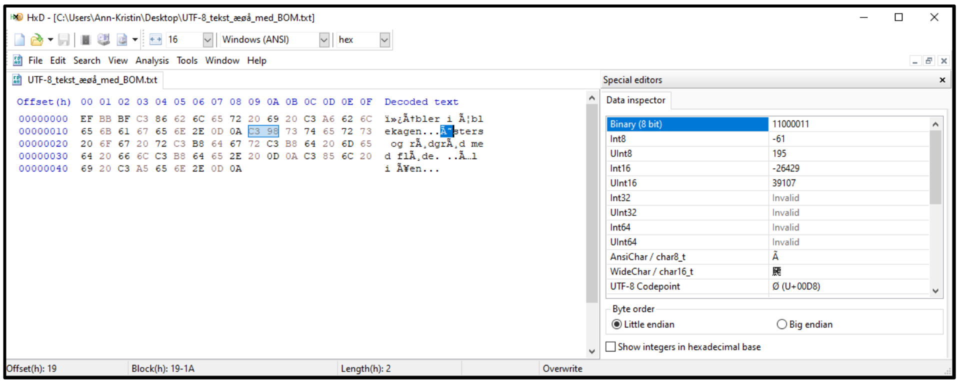
Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-31037
UInt16	34499
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	蛃
UTF-8 Codepoint	Æ (U+00C6)

Byte order: ● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 3 Block(h): 3-4 Length(h): 2 Overwrite

***Figure 6.6** UTF-8 file with uppercase Æ (hexadecimal value: C3 86)*


HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 | Windows (ANSI) | hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

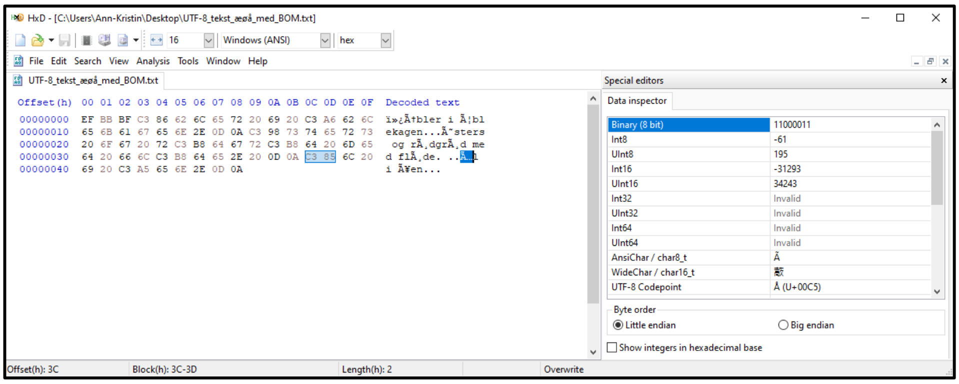
Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-26429
UInt16	39107
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	飃
UTF-8 Codepoint	Ø (U+00D8)

Byte order: ● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 19 Block(h): 19-1A Length(h): 2 Overwrite

***Figure 6.7** UTF-8 file with uppercase Ø (hexadecimal value: C3 98)*


HxD - [C:\Users\Ann-Kristin\Desktop\UTF-8_tekst_æøå_med_BOM.txt]

16 | Windows (ANSI) | hex

UTF-8_tekst_æøå_med_BOM.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  EF BB BF C3 86 62 6C 65 72 20 69 20 C3 A6 62 6C  ï»¿Ã†bler i Ã¦bl
00000010  65 6B 61 67 65 6E 2E 0D 0A C3 98 73 74 65 72 73  ekagen...Ã˜sters
00000020  20 6F 67 20 72 C3 B8 64 67 72 C3 B8 64 20 6D 65   og rÃ¸dgrÃ¸d me
00000030  64 20 66 6C C3 B8 64 65 2E 20 0D 0A C3 85 6C 20  d flÃ¸de. ..Ã…l
00000040  69 20 C3 A5 65 6E 2E 0D 0A                       i Ã¥en...
```

Special editors

Data inspector

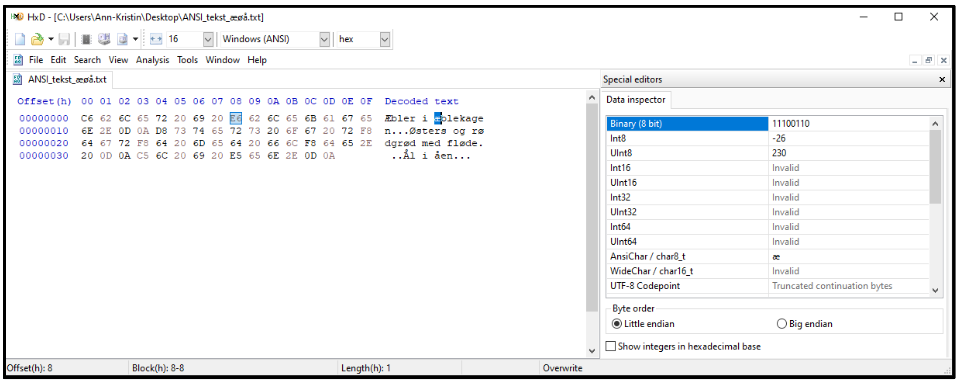
Binary (8 bit)	11000011
Int8	-61
UInt8	195
Int16	-31293
UInt16	34243
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	Ã
WideChar / char16_t	藃
UTF-8 Codepoint	Å (U+00C5)

Byte order: ● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 3C Block(h): 3C-3D Length(h): 2 Overwrite

***Figure 6.8** UTF-8 file with uppercase Å (hexadecimal value: C3 85)*


HxD - [C:\Users\Ann-Kristin\Desktop\ANSI_tekst_æøå.txt]
File Edit Search View Analysis Tools Window Help
ANSI_tekst_æøå.txt
Special editors
Data inspector

16 | Windows (ANSI) | hex

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  C6 62 6C 65 72 20 69 20 E6 62 6C 65 6B 61 67 65  Æbler i æblekage
00000010  6E 2E 0D 0A D8 73 74 65 72 73 20 6F 67 20 72 F8  n...Østers og rø
00000020  64 67 72 F8 64 20 6D 65 64 20 66 6C F8 64 65 2E  dgrød med fløde.
00000030  20 0D 0A C5 6C 20 69 20 E5 65 6E 2E 0D 0A         ..Ål i åen...
```

Binary (8 bit)	11100110
Int8	-26
UInt8	230
Int16	Invalid
UInt16	Invalid
Int32	Invalid
UInt32	Invalid
Int64	Invalid
UInt64	Invalid
AnsiChar / char8_t	æ
WideChar / char16_t	Invalid
UTF-8 Codepoint	Truncated continuation bytes

Byte order
● Little endian ○ Big endian

□ Show integers in hexadecimal base

Offset(h): 8 | Block(h): 8-8 | Length(h): 1 | Overwrite

Figure 6.9 *ANSI file with uppercase Æ (hexadecimal value: E6)*

NOTE that Figure 6.9 shows a file encoded with ANSI character sets. The hexadecimal value is E6, which is a reference to UTF-8 codepoint U+00E6 for lowercase ~~e~~ æ in UTF-8. The hexadecimal value E6 is *not* a valid UTF-8 hexadecimal value for small æ.

NOTE that æ, ø, and å are displayed correctly in Figure 6.9 because the text file is encoded as ANSI and the text editor translates/decodes these values into ANSI characters (Decoded text).

7. UTF-8 support

If you experience problems identifying character sets in files and changing character sets to UTF-8, contact the research data manager in the National Archives on the following e-mail: mailbox@sa.dk .

Feltkode ændret

Appendix 1: UTF-8 table with translation between character and hex-values

Unicode code point	character	UTF-8 (hex.)	Name
U+0020		20	SPACE
U+0021	!	21	EXCLAMATION MARK
U+0022	"	22	QUOTATION MARK
U+0023	#	23	NUMBER SIGN
U+0024	$	24	DOLLAR SIGN
U+0025	%	25	PERCENT SIGN
U+0026	&	26	AMPERSAND
U+0027	'	27	APOSTROPHE
U+0028	(	28	LEFT PARENTHESIS
U+0029	)	29	RIGHT PARENTHESIS
U+002A	*	2a	ASTERISK
U+002B	+	2b	PLUS SIGN
U+002C	,	2c	COMMA
U+002D	-	2d	HYPHEN-MINUS
U+002E	.	2e	FULL STOP
U+002F	/	2f	SOLIDUS
U+0030	0	30	DIGIT ZERO
U+0031	1	31	DIGIT ONE
U+0032	2	32	DIGIT TWO
U+0033	3	33	DIGIT THREE
U+0034	4	34	DIGIT FOUR
U+0035	5	35	DIGIT FIVE
U+0036	6	36	DIGIT SIX
U+0037	7	37	DIGIT SEVEN
U+0038	8	38	DIGIT EIGHT
U+0039	9	39	DIGIT NINE
U+003A	:	3a	COLON
U+003B	;	3b	SEMICOLON
U+003C	<	3c	LESS-THAN SIGN
U+003D	=	3d	EQUALS SIGN
U+003E	>	3e	GREATER-THAN SIGN
U+003F	?	3f	QUESTION MARK
U+0040	@	40	COMMERCIAL AT
U+0041	A	41	LATIN CAPITAL LETTER A
U+0042	B	42	LATIN CAPITAL LETTER B
U+0043	C	43	LATIN CAPITAL LETTER C
U+0044	D	44	LATIN CAPITAL LETTER D
U+0045	E	45	LATIN CAPITAL LETTER E
U+0046	F	46	LATIN CAPITAL LETTER F

U+0047	G	47	LATIN CAPITAL LETTER G
U+0048	H	48	LATIN CAPITAL LETTER H
U+0049	I	49	LATIN CAPITAL LETTER I
U+004A	J	4a	LATIN CAPITAL LETTER J
U+004B	K	4b	LATIN CAPITAL LETTER K
U+004C	L	4c	LATIN CAPITAL LETTER L
U+004D	M	4d	LATIN CAPITAL LETTER M
U+004E	N	4e	LATIN CAPITAL LETTER N
U+004F	O	4f	LATIN CAPITAL LETTER O
U+0050	P	50	LATIN CAPITAL LETTER P
U+0051	Q	51	LATIN CAPITAL LETTER Q
U+0052	R	52	LATIN CAPITAL LETTER R
U+0053	S	53	LATIN CAPITAL LETTER S
U+0054	T	54	LATIN CAPITAL LETTER T
U+0055	U	55	LATIN CAPITAL LETTER U
U+0056	V	56	LATIN CAPITAL LETTER V
U+0057	W	57	LATIN CAPITAL LETTER W
U+0058	X	58	LATIN CAPITAL LETTER X
U+0059	Y	59	LATIN CAPITAL LETTER Y
U+005A	Z	5a	LATIN CAPITAL LETTER Z
U+005B	[	5b	LEFT SQUARE BRACKET
U+005C	\	5c	REVERSE SOLIDUS
U+005D	]	5d	RIGHT SQUARE BRACKET
U+005E	^	5e	CIRCUMFLEX ACCENT
U+005F	_	5f	LOW LINE
U+0060	`	60	GRAVE ACCENT
U+0061	a	61	LATIN SMALL LETTER A
U+0062	b	62	LATIN SMALL LETTER B
U+0063	c	63	LATIN SMALL LETTER C
U+0064	d	64	LATIN SMALL LETTER D
U+0065	e	65	LATIN SMALL LETTER E
U+0066	f	66	LATIN SMALL LETTER F
U+0067	g	67	LATIN SMALL LETTER G
U+0068	h	68	LATIN SMALL LETTER H
U+0069	i	69	LATIN SMALL LETTER I
U+006A	j	6a	LATIN SMALL LETTER J
U+006B	k	6b	LATIN SMALL LETTER K
U+006C	l	6c	LATIN SMALL LETTER L
U+006D	m	6d	LATIN SMALL LETTER M
U+006E	n	6e	LATIN SMALL LETTER N
U+006F	o	6f	LATIN SMALL LETTER O
U+0070	p	70	LATIN SMALL LETTER P
U+0071	q	71	LATIN SMALL LETTER Q

U+0072	r	72	LATIN SMALL LETTER R
U+0073	s	73	LATIN SMALL LETTER S
U+0074	t	74	LATIN SMALL LETTER T
U+0075	u	75	LATIN SMALL LETTER U
U+0076	v	76	LATIN SMALL LETTER V
U+0077	w	77	LATIN SMALL LETTER W
U+0078	x	78	LATIN SMALL LETTER X
U+0079	y	79	LATIN SMALL LETTER Y
U+007A	z	7a	LATIN SMALL LETTER Z
U+007B	{	7b	LEFT CURLY BRACKET
U+007C	|	7c	VERTICAL LINE
U+007D	}	7d	RIGHT CURLY BRACKET
U+007E	~	7e	TILDE
U+00A0		c2 a0	NO-BREAK SPACE
U+00A1	¡	c2 a1	INVERTED EXCLAMATION MARK
U+00A2	¢	c2 a2	CENT SIGN
U+00A3	£	c2 a3	POUND SIGN
U+00A4	¤	c2 a4	CURRENCY SIGN
U+00A5	¥	c2 a5	YEN SIGN
U+00A6	¦	c2 a6	BROKEN BAR
U+00A7	§	c2 a7	SECTION SIGN
U+00A8	¨	c2 a8	DIAERESIS
U+00A9	©	c2 a9	COPYRIGHT SIGN
U+00AA	ª	c2 aa	FEMININE ORDINAL INDICATOR
U+00AB	«	c2 ab	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
U+00AC	¬	c2 ac	NOT SIGN
U+00AD		c2 ad	SOFT HYPHEN
U+00AE	®	c2 ae	REGISTERED SIGN
U+00AF	¯	c2 af	MACRON
U+00B0	°	c2 b0	DEGREE SIGN
U+00B1	±	c2 b1	PLUS-MINUS SIGN
U+00B2	²	c2 b2	SUPERSCRIPT TWO
U+00B3	³	c2 b3	SUPERSCRIPT THREE
U+00B4	´	c2 b4	ACUTE ACCENT
U+00B5	µ	c2 b5	MICRO SIGN
U+00B6	¶	c2 b6	PILCROW SIGN
U+00B7	·	c2 b7	MIDDLE DOT
U+00B8	¸	c2 b8	CEDILLA
U+00B9	¹	c2 b9	SUPERSCRIPT ONE
U+00BA	º	c2 ba	MASCULINE ORDINAL INDICATOR
U+00BB	»	c2 bb	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK

U+00BC	¼	c2 bc	VULGAR FRACTION ONE QUARTER
U+00BD	½	c2 bd	VULGAR FRACTION ONE HALF
U+00BE	¾	c2 be	VULGAR FRACTION THREE QUARTERS
U+00BF	¿	c2 bf	INVERTED QUESTION MARK
U+00C0	À	c3 80	LATIN CAPITAL LETTER A WITH GRAVE
U+00C1	Á	c3 81	LATIN CAPITAL LETTER A WITH ACUTE
U+00C2	Â	c3 82	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
U+00C3	Ã	c3 83	LATIN CAPITAL LETTER A WITH TILDE
U+00C4	Ä	c3 84	LATIN CAPITAL LETTER A WITH DIAERESIS
U+00C5	Å	c3 85	LATIN CAPITAL LETTER A WITH RING ABOVE
U+00C6	Æ	c3 86	LATIN CAPITAL LETTER AE
U+00C7	Ç	c3 87	LATIN CAPITAL LETTER C WITH CEDILLA
U+00C8	È	c3 88	LATIN CAPITAL LETTER E WITH GRAVE
U+00C9	É	c3 89	LATIN CAPITAL LETTER E WITH ACUTE
U+00CA	Ê	c3 8a	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
U+00CB	Ë	c3 8b	LATIN CAPITAL LETTER E WITH DIAERESIS
U+00CC	Ì	c3 8c	LATIN CAPITAL LETTER I WITH GRAVE
U+00CD	Í	c3 8d	LATIN CAPITAL LETTER I WITH ACUTE
U+00CE	Î	c3 8e	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
U+00CF	Ï	c3 8f	LATIN CAPITAL LETTER I WITH DIAERESIS
U+00D0	Ð	c3 90	LATIN CAPITAL LETTER ETH
U+00D1	Ñ	c3 91	LATIN CAPITAL LETTER N WITH TILDE
U+00D2	Ò	c3 92	LATIN CAPITAL LETTER O WITH GRAVE
U+00D3	Ó	c3 93	LATIN CAPITAL LETTER O WITH ACUTE
U+00D4	Ô	c3 94	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
U+00D5	Õ	c3 95	LATIN CAPITAL LETTER O WITH TILDE
U+00D6	Ö	c3 96	LATIN CAPITAL LETTER O WITH DIAERESIS
U+00D7	×	c3 97	MULTIPLICATION SIGN
U+00D8	Ø	c3 98	LATIN CAPITAL LETTER O WITH STROKE
U+00D9	Ù	c3 99	LATIN CAPITAL LETTER U WITH GRAVE
U+00DA	Ú	c3 9a	LATIN CAPITAL LETTER U WITH ACUTE
U+00DB	Û	c3 9b	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
U+00DC	Ü	c3 9c	LATIN CAPITAL LETTER U WITH DIAERESIS
U+00DD	Ý	c3 9d	LATIN CAPITAL LETTER Y WITH ACUTE
U+00DE	Þ	c3 9e	LATIN CAPITAL LETTER THORN
U+00DF	ß	c3 9f	LATIN SMALL LETTER SHARP S
U+00E0	à	c3 a0	LATIN SMALL LETTER A WITH GRAVE
U+00E1	á	c3 a1	LATIN SMALL LETTER A WITH ACUTE
U+00E2	â	c3 a2	LATIN SMALL LETTER A WITH CIRCUMFLEX
U+00E3	ã	c3 a3	LATIN SMALL LETTER A WITH TILDE
U+00E4	ä	c3 a4	LATIN SMALL LETTER A WITH DIAERESIS
U+00E5	å	c3 a5	LATIN SMALL LETTER A WITH RING ABOVE
U+00E6	æ	c3 a6	LATIN SMALL LETTER AE

U+00E7	ç	c3 a7	LATIN SMALL LETTER C WITH CEDILLA
U+00E8	è	c3 a8	LATIN SMALL LETTER E WITH GRAVE
U+00E9	é	c3 a9	LATIN SMALL LETTER E WITH ACUTE
U+00EA	ê	c3 aa	LATIN SMALL LETTER E WITH CIRCUMFLEX
U+00EB	ë	c3 ab	LATIN SMALL LETTER E WITH DIAERESIS
U+00EC	ì	c3 ac	LATIN SMALL LETTER I WITH GRAVE
U+00ED	í	c3 ad	LATIN SMALL LETTER I WITH ACUTE
U+00EE	î	c3 ae	LATIN SMALL LETTER I WITH CIRCUMFLEX
U+00EF	ï	c3 af	LATIN SMALL LETTER I WITH DIAERESIS
U+00F0	ð	c3 b0	LATIN SMALL LETTER ETH
U+00F1	ñ	c3 b1	LATIN SMALL LETTER N WITH TILDE
U+00F2	ò	c3 b2	LATIN SMALL LETTER O WITH GRAVE
U+00F3	ó	c3 b3	LATIN SMALL LETTER O WITH ACUTE
U+00F4	ô	c3 b4	LATIN SMALL LETTER O WITH CIRCUMFLEX
U+00F5	õ	c3 b5	LATIN SMALL LETTER O WITH TILDE
U+00F6	ö	c3 b6	LATIN SMALL LETTER O WITH DIAERESIS
U+00F7	÷	c3 b7	DIVISION SIGN
U+00F8	ø	c3 b8	LATIN SMALL LETTER O WITH STROKE
U+00F9	ù	c3 b9	LATIN SMALL LETTER U WITH GRAVE
U+00FA	ú	c3 ba	LATIN SMALL LETTER U WITH ACUTE
U+00FB	û	c3 bb	LATIN SMALL LETTER U WITH CIRCUMFLEX
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS
U+00FD	ý	c3 bd	LATIN SMALL LETTER Y WITH ACUTE
U+00FE	þ	c3 be	LATIN SMALL LETTER THORN
U+00FF	ÿ	c3 bf	LATIN SMALL LETTER Y WITH DIAERESIS