An end-to-end solution for using unicode with Blaise to support any language

- Alerk Amin (CentERdata)
- Richard Boreham (NatCen)
- Maurice Martens (CentERdata)
- Colin Miceli (NatCen)
Overview

- **Understanding Society**
  - Language requirements
- **Solution**
  - Translation of questionnaire
  - Questionnaire setup
  - Unicode inside Blaise
  - UNITIP
- **Demonstration**
- **Interviewer feedback**
Survey Design

• Design
  ▸ 40,000 households per wave
  ▸ 32.5 min CAPI for ALL adults (16+)
  ▸ Longitudinal

• 1000 adult interviews with each of the 5 groups
  ▸ Indian, Pakistani, Bangladeshi, Black-African, Black-Caribbean

• Also eligible for interview
  ▸ Sri Lankan, Chinese, Far Eastern, Middle Eastern & Iranian, Turkish, North-African, Asian-African
Language requirements

• 9 main languages are translated
  ▶ Welsh
  ▶ Bengali, Gujarati, Punjabi (Urdu), Punjabi (Gurmukhi), Urdu
  ▶ Arabic, Cantonese, Somali

• Multiple languages in a household
• Right-to-left languages
• Showcards
Solution overview

Paper based translations

English Qure

LMU

MBG

UNITIP
Translation process

- Translate
- Proof-read
- Check
- Query & retranslate
- Adjudicate
- Sign-off
Questionnaire setup

• **Tags**
  - MvEver (DE_MvEver)

• **Types**
  - Standard types at datamodel level
  - Non-translated types in different file

• **Textfills**
  - Separate procedure for each textfill
    He/she varies according to context
PROCEDURE txtLACal

PARAMETERS
IMPORT imLACSx: TBoyGirl
EXPORT exSFLCal_TFGender2: STRING

RULES
IF (imLACSx=Boy) THEN
    exSFLACal_TFGender2:='he'  \{SFLACal_TFGender[1]\}
ELSEIF (imLACSx=Girl) THEN
    exSFLACal_TFGender2:='she'  \{SFLACal_TFGender[2]\}
ENDIF

ENDPROCEDURE
RULES

IF (imLACSx=Boy) THEN

  IF UT.CurrentLanguage=L1 THEN
    exSFLACal_TFGender2:=‘he’ {SFLACal_TFGender[1]}
  ELSEIF UT.CurrentLanguage=L2 THEN
    exSFLACal_TFGender2:=‘ayay’ {SFLACal_TFGender[1]}

  ...

Unicode

• **UTF-8**
  - 1-4 bytes for every character
  - Diacritics are added to base character
    \[কী = ক + ় + ঃ + ী = 9 \text{ bytes}\]

• **Backwards compatible with ASCII**
  - 1 byte Latin characters are the same
  - No special characters codes (space, newline) in extra bytes
Using UTF-8 in Blaise

- Variable names are Latin characters
- Strings in Blaise are Extended ASCII
- Replace the ASCII strings with UTF-8
  - Text processing works perfectly
    - BLA -> BMI, fills
  - Rendering does not work
    - Blaise Editor, DEP
- Questionnaire modification
  - String Length
    - Latin strings are 1 byte per character
    - Other languages can be 1-4 bytes
    - String variables must be 2-4x larger
“No” in Bengali

- না = ন + আ = “na” + “aa”
- UTF-8
  - ন = E0 A6 A9
  - আ = E0 A6 BE
- Extended ASCII
  - E0 = à
  - A6 = ’
  - A9 = ¨
  - BE = ¾
- TYesNo = … No (2) "No" "à¨à’¾"
  - This is how Blaise renders the string
UNITIP

- Similar functionality as Blaise DEP
- All visual controls support UTF-8
- Blaise API
  - Blaise processes all texts as though they are ASCII
    Variable names (for fills) are all Latin characters
  - UNITIP gets the strings from Blaise, and renders it correctly as UTF-8

Question Text
Answer Categories
Demonstration
Interviewer feedback

- Translation pilot
  - Bi-lingual interviewers
  - Interviewer/translator pairs
  - Bengali and Punjabi Urdu (RTL)
- Timing of translated interview was not much longer than English interview
- Data entry was not as fast as Blaise DEP
- Increase quality of interview and responses
- Toggling between translation and English was helpful and used extensively
- Improvement over old paper-based system
Questions