Building a DDI Codebook using R

Adrian Dușa
University of Bucharest
2024-06-25

Audience

- data producers
- data archivists

- deposit to a public repository
- maintain in-house collection

- new to data documentation
- experts on documenting data

- already know about R
- never heard of it

- well resourced
- tight on funding

Secondary use

Data producers

- individual researchers
- research institutions
- government agencies
- public at large

All need to maintain a structured collection of their data.

(Meta)data package

The dataset:

- properly labelled variables and values
- cleaned for imperfections (answers where they should not be)

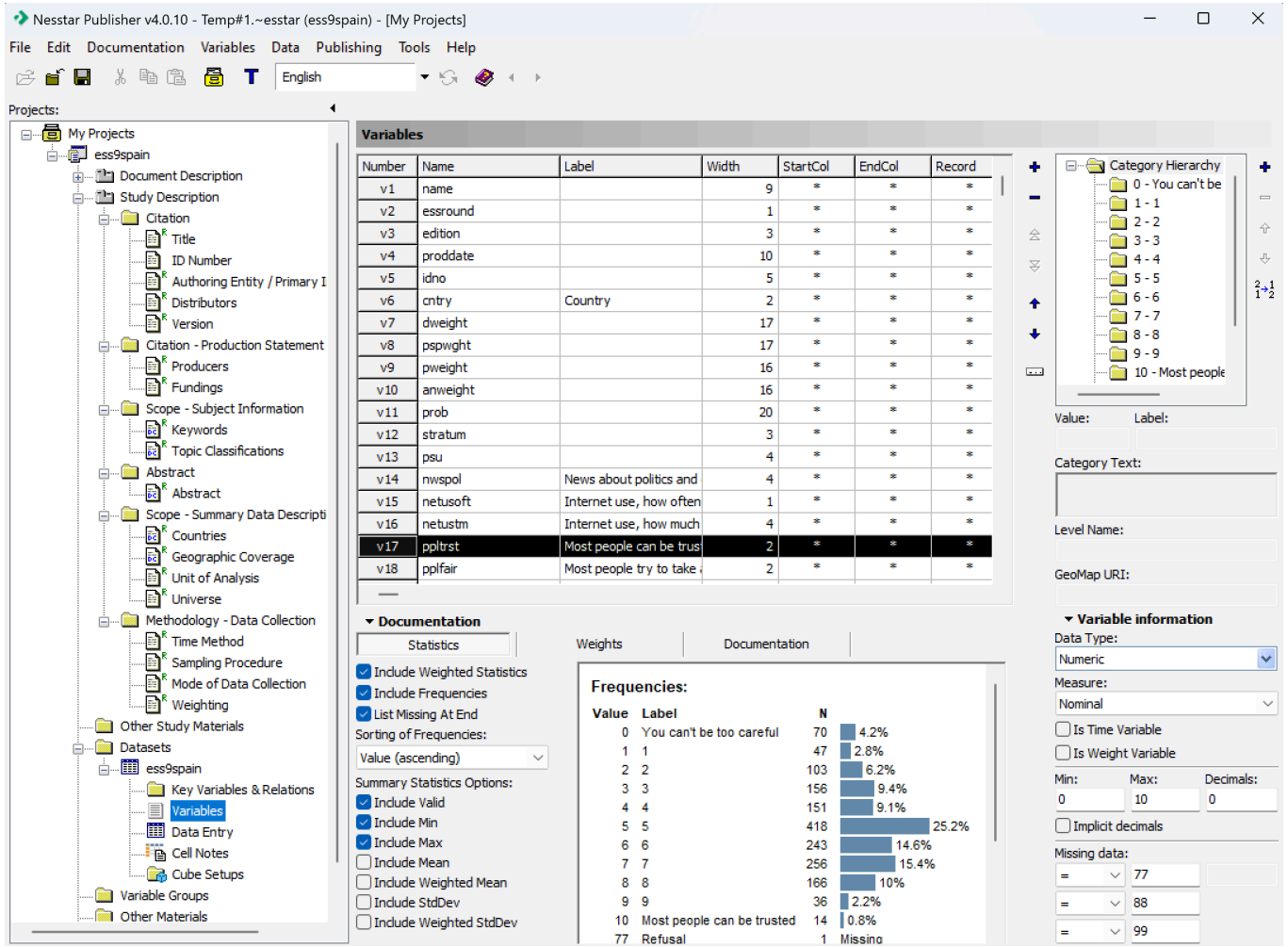
A description of the study:

- objective, abstract
- universe (who are the respondents)
- sampling procedure (how are they selected)
- geographical location
- collection time, etc

The dream goal

- a single, cross-platform
- self-contained
- open source software
- working out of the box
- that would allow anyone (researchers and archivists)
- to fully populate a DDI Codebook

Once upon a time...



Web environment

Node.js is an asynchronous, event driven Javascript runtime.

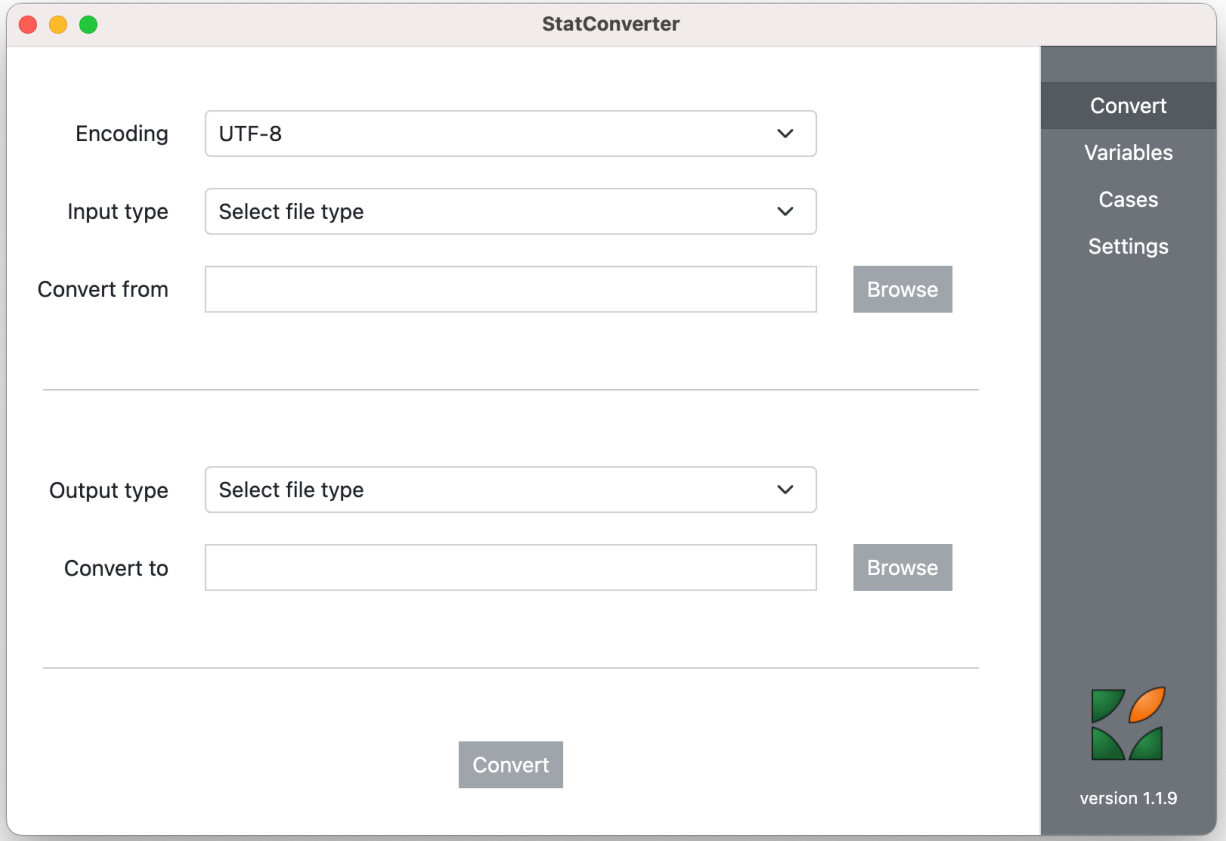
It allows developers to write server side high performance and networked applications.

Electron.js is a runtime framework that allows users to create desktop applications with HTML, CSS, and JavaScript.

It can build binary versions for all operating systems:

Write once, deploy everywhere.

StatConverter



https://roda.github.io/StatConverter/

Until that is done...

Command line

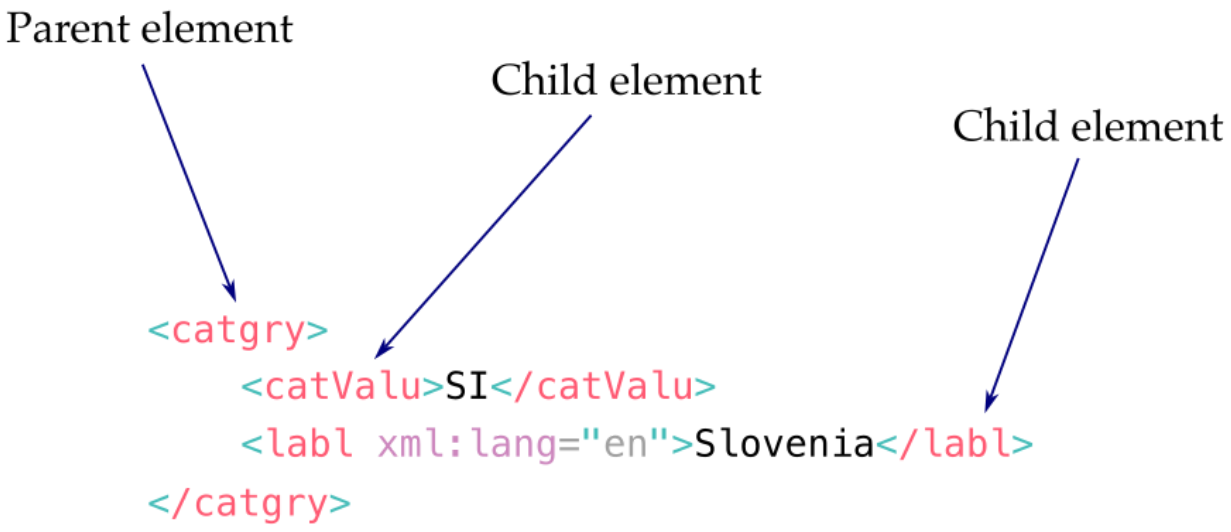
Package DDIwR has a wealth of useful commands to extract metadata and build a data description section of the DDI Codebook

It can gradually construct a DDI Codebook using a series of chained commands.

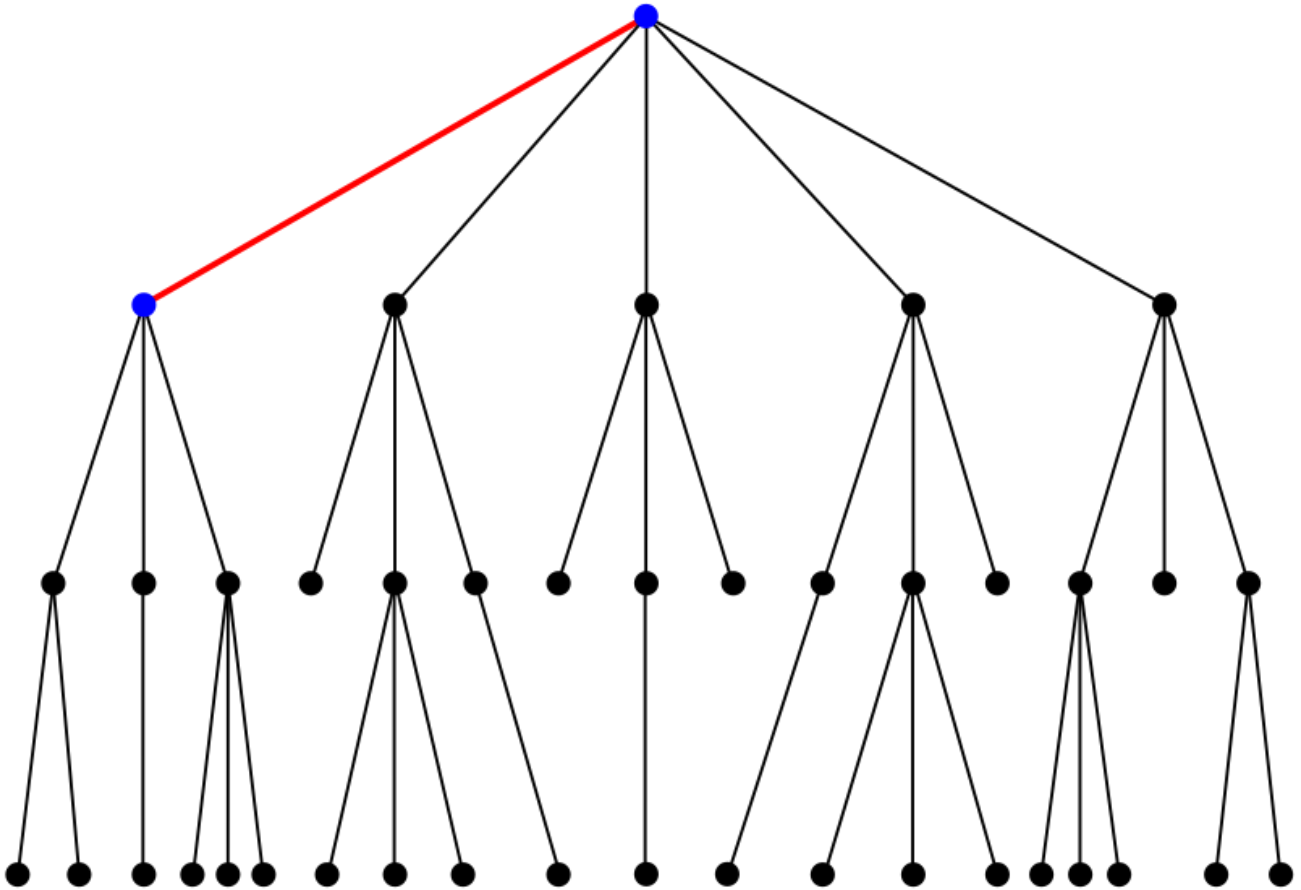
DDI Codebook

Corresponding structure in R

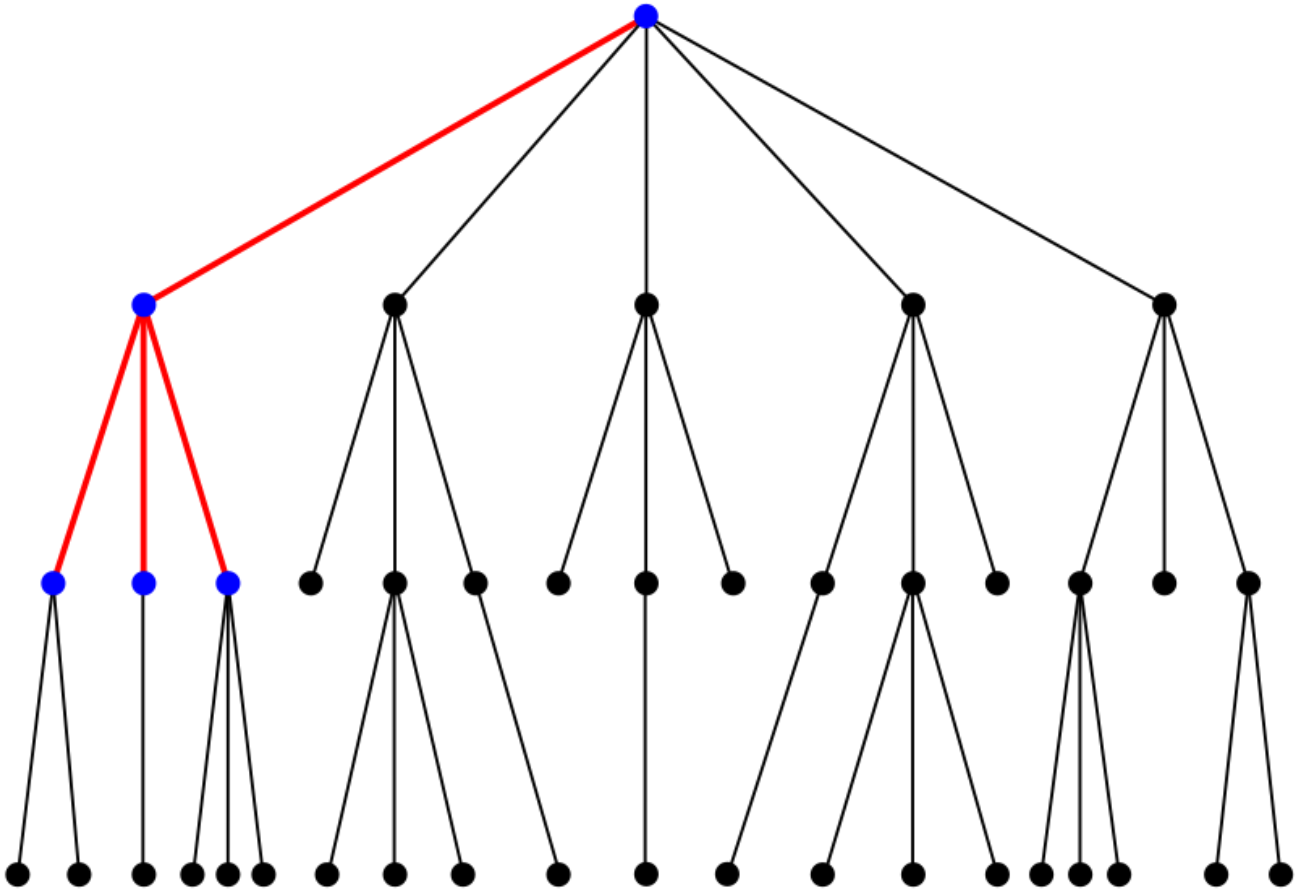
```
$catgry
$catgry$catValu
$catgry$catValu[[1]]
[1] "SI"


$catgry$labl
$catgry$labl[[1]]
[1] "Slovenia"

attr(,"xmlang")
[1] "en"
```



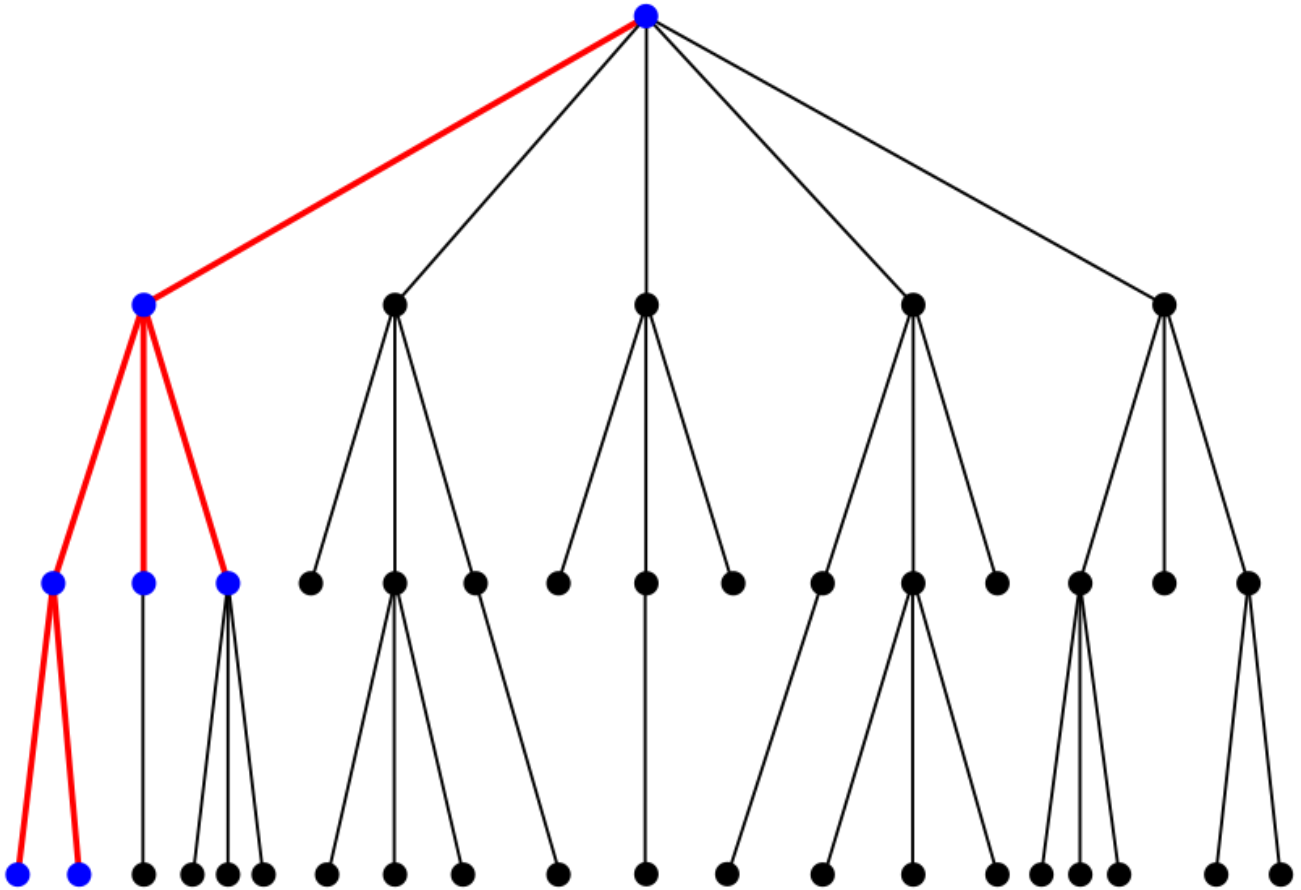
Top-down approach



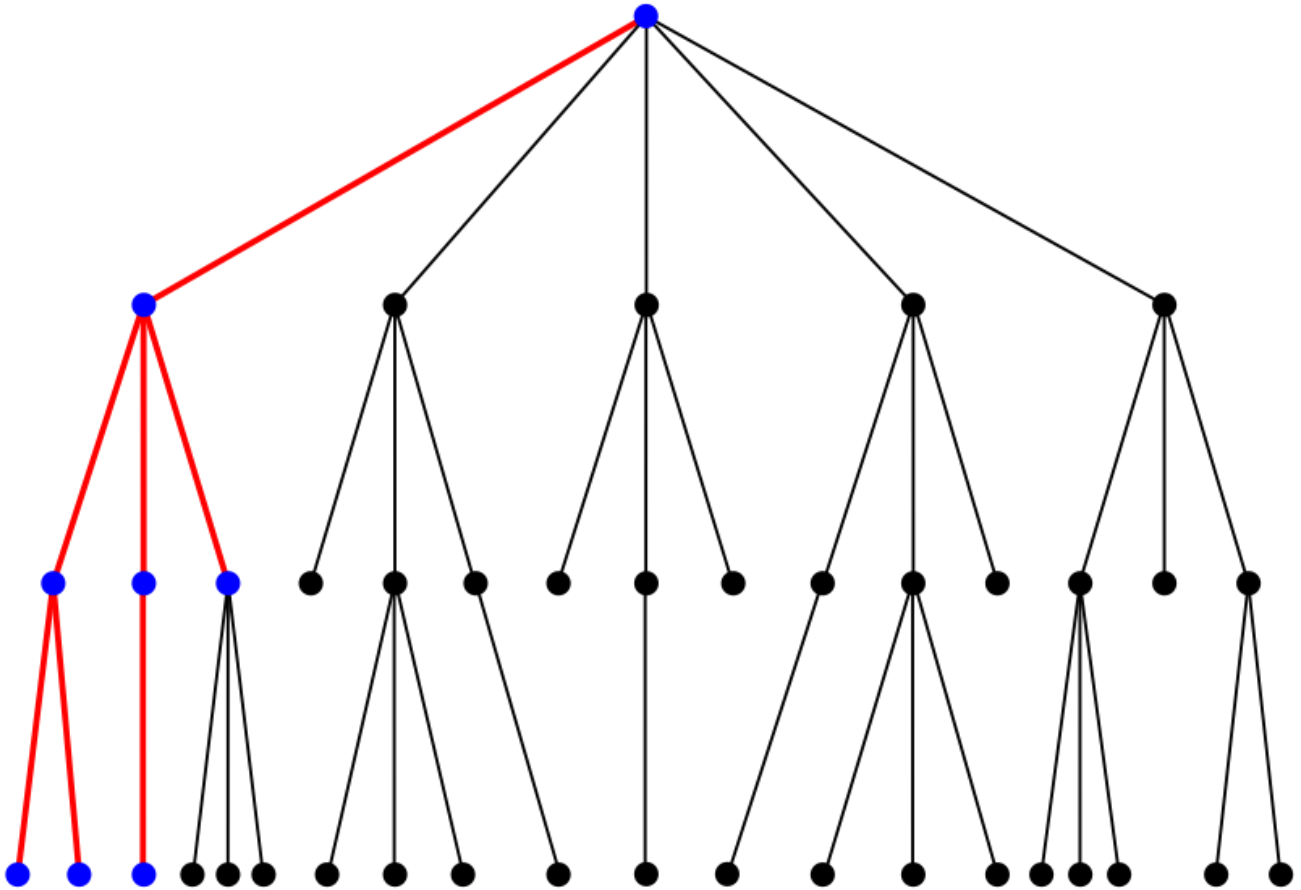
Top-down approach



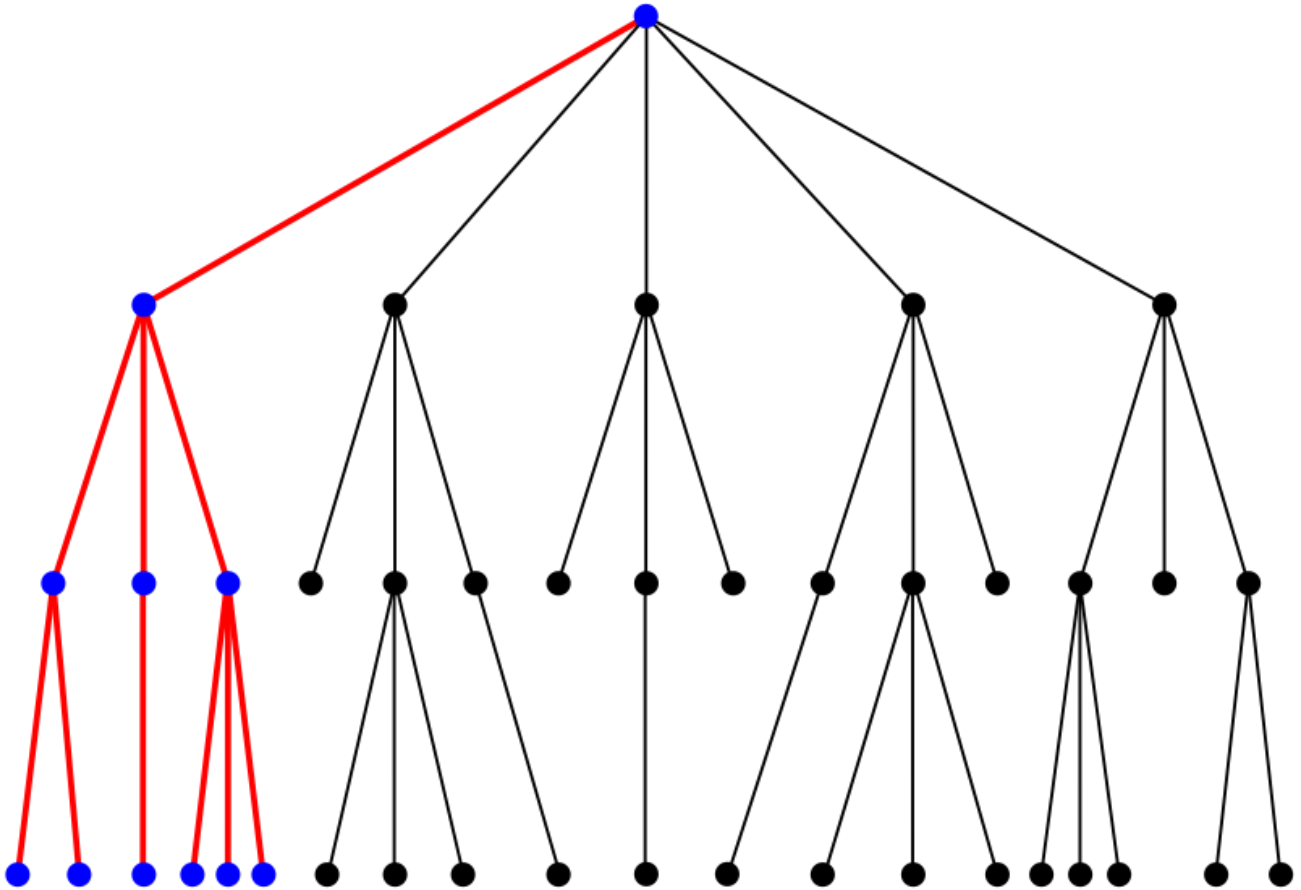
Top-down approach



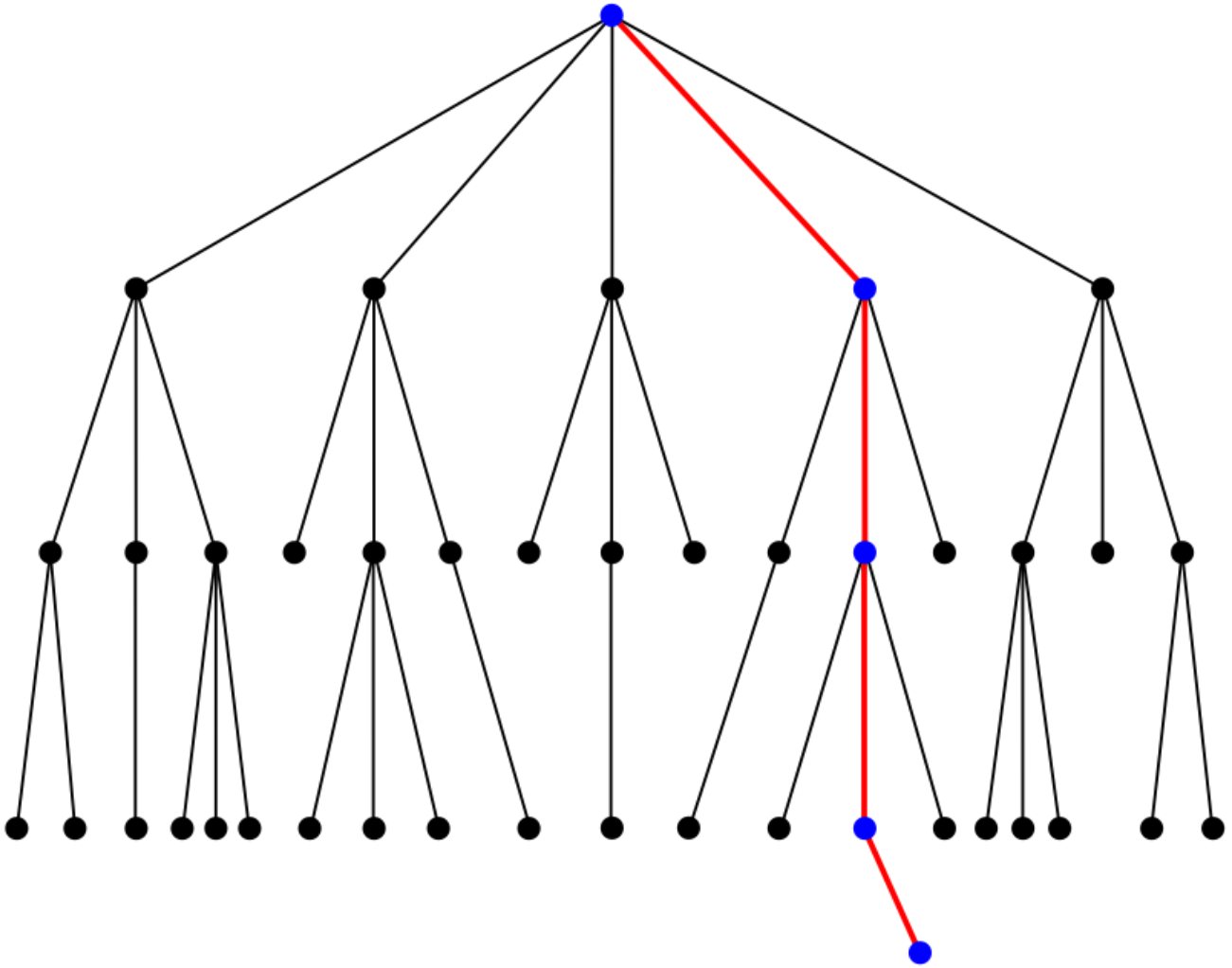
Top-down approach



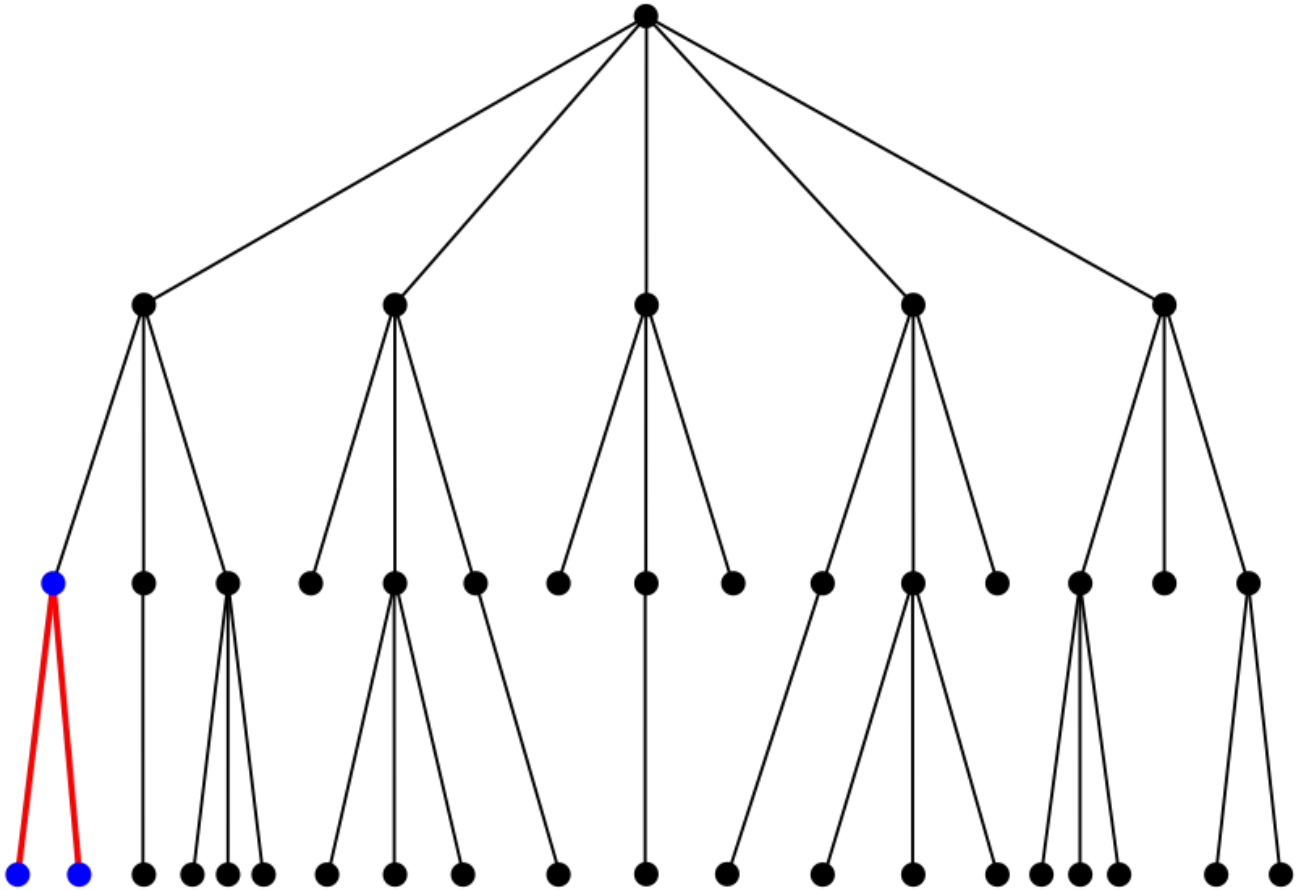
Top-down approach



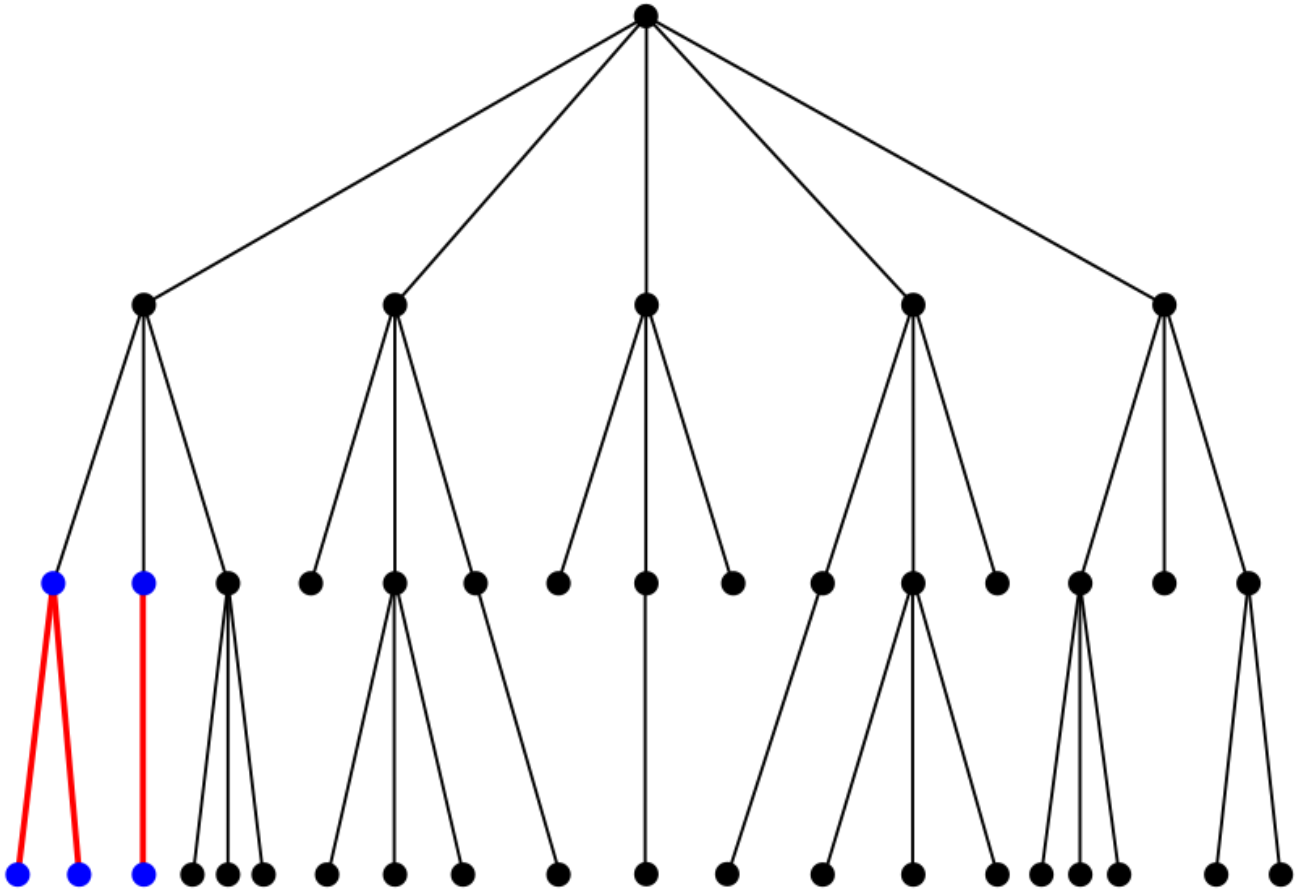
Top-down approach



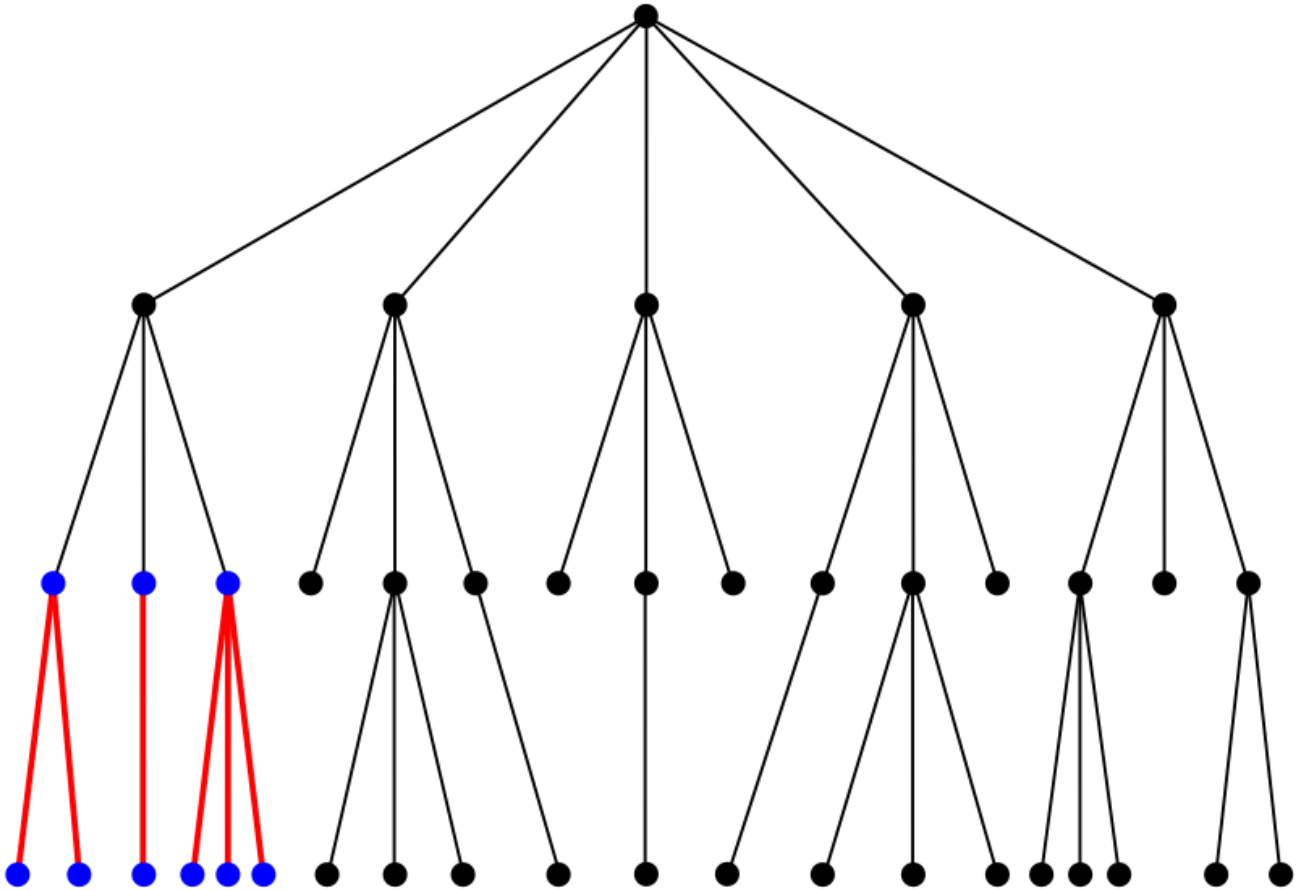
Bottom-up approach



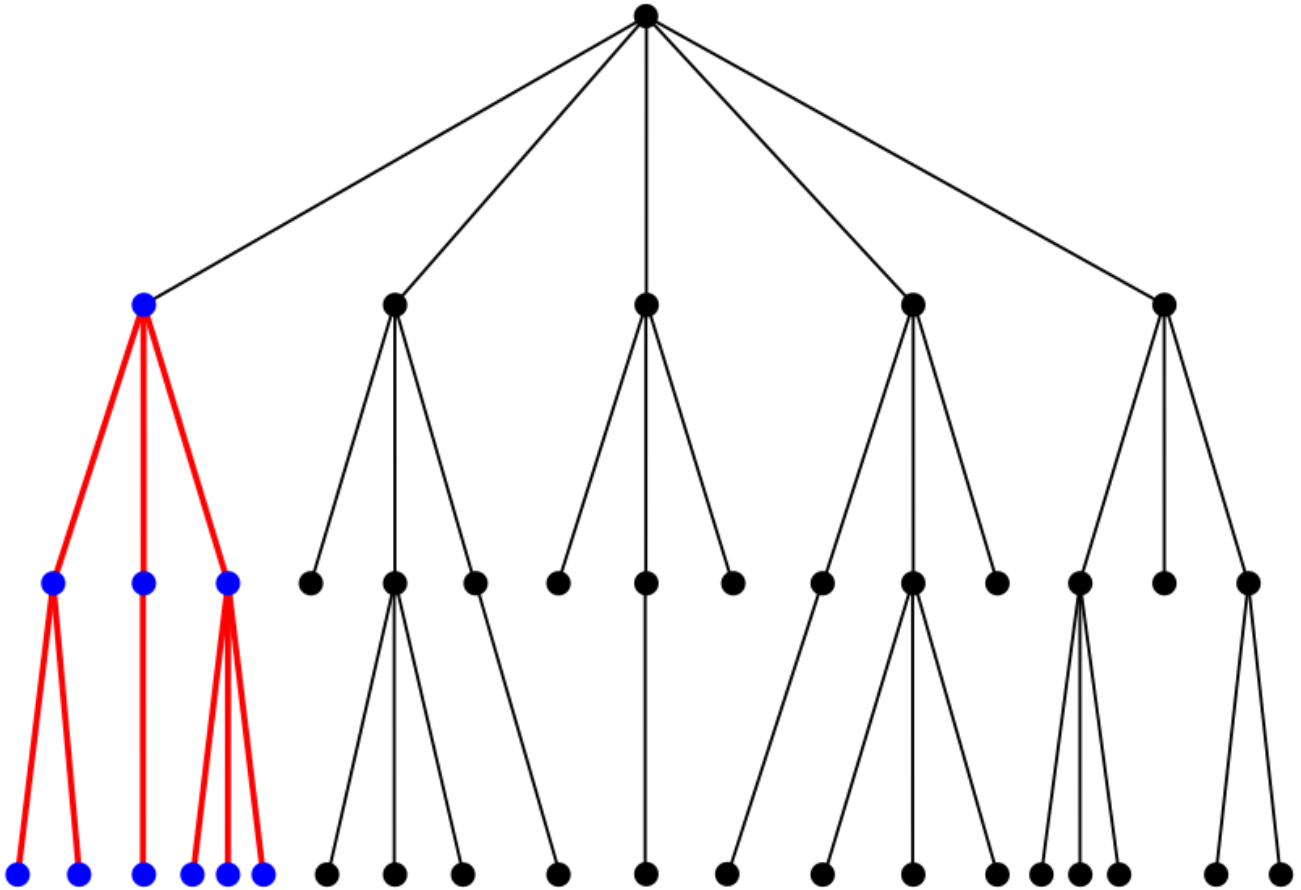
Bottom-up approach



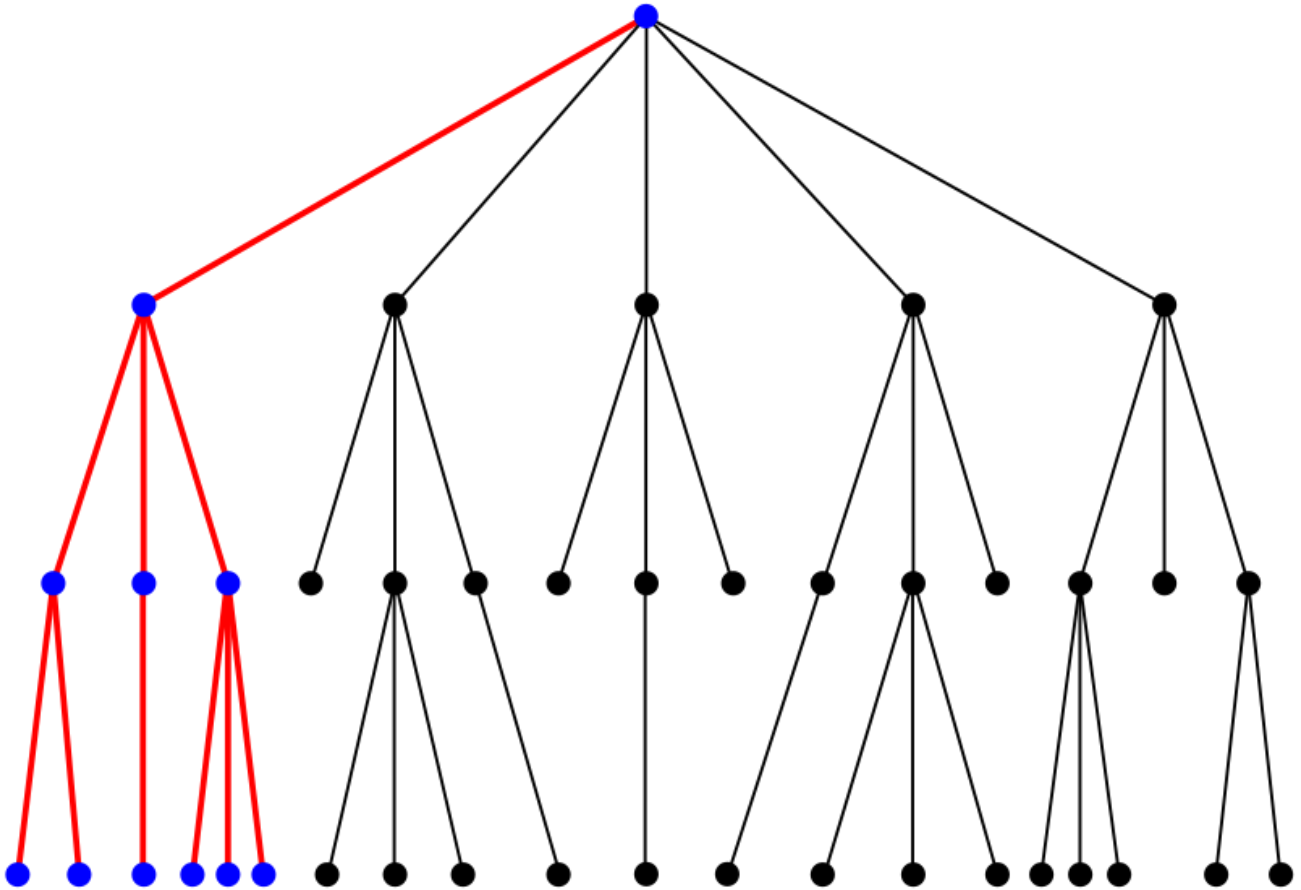
Bottom-up approach



Bottom-up approach



Bottom-up approach



Bottom-up R code

```r
catValu <- makeElement("catValu", content = "SI")

labl <- makeElement(
  "labl",
  content = "Slovenia",
  attributes = c(xmlang = "en")
)

catgry <- makeElement(
  "catgry",
  children = list(catValu, labl)
)
```

```xml
<catgry>
    <catValu>SI</catValu>
    <labl xml:lang="en">Slovenia</labl>
</catgry>
```

Notes

Step by step, can be tedious for manually creating elements

Definitely unfeasible to describe the variables (dataDscr)

Feasible for the smaller parts of the DDI Codebook (docDscr, stdyDscr etc.)

Not a problem for a script taking information from a database, or from a dataset

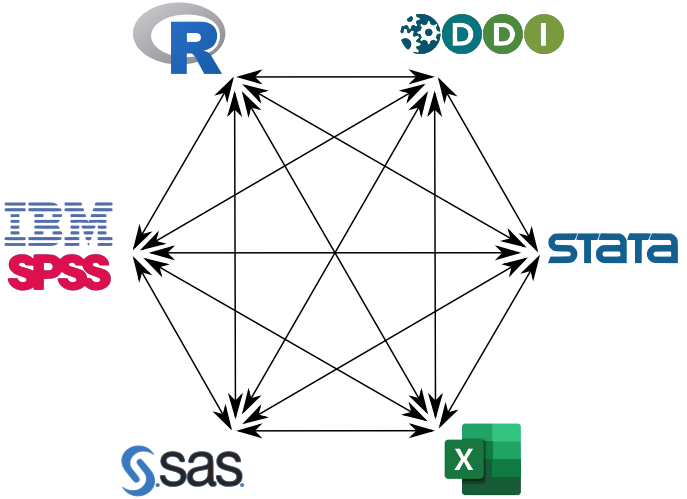
R package DDIwR

- handles DDI Codebook v2.6(!)
- much finer and grained control over the DDI XML output
- imports and export to and from: SPSS, Stata, SAS, R and Excel
- capable of reading metadata from social science datasets
- translates the metadata into an R list, compatible with DDI's XML structure

R package DDIwR

Official CRAN page:

https://cran.r-project.org/package=DDIwR



Solves an important base R shortcoming:
define and declare multiple, different missing values.

It is an open source software, with public code on GitHub:

https://github.com/dusadrian/DDIwR

How to use

First, install the package with all its dependencies:

```
install.packages("DDIwR", dependencies = TRUE)
```

Then, load the package to access its added functionality:

```
library("DDIwR")
```

```
catValu <- makeElement("catValu", content = "SI")
```

Help files are available:

```
?makeElement
```

Help file

R Help

Print

Help Search

makeElement {DDIwR} R Documentation

Make a DDI Codebook element

Description

Creates a standard DDI element.

Usage

```
makeElement(
  name,
  children = NULL,
  attributes = NULL,
  content = NULL,
  fill = FALSE,
  ...
)
```

Arguments

`name`
: Character, a DDI Codebook element name.

`children`
: A list of standard DDI codebook elements.

`attributes`
: A vector of named values.

`content`
: Character scalar.

`fill`
: Logical, fill the element with arbitrary values for its mandatory children and attributes

`...`
: Other arguments, see Details.

Help file: examples section

R Help

Print

Help Search

Details

The structure of a DDI element in R follows the usual structure of an XML node, as returned by the function `as_list()` from package **xml2**, with one additional (first) component named ".extra" to accommodate any other information that is not part of the DDI element.

In the DDI Codebook, most elements and their attributes are optional, but some are mandatory. In case of attributes, some become mandatory only if the element itself is present. The mandatory elements need to be present in the final version of the Codebook, to pass the validation against the XML schema.

By activating the argument `fill`, this function creates DDI elements containing all mandatory (sub)elements and (their) attributes, filled with arbitrary values that can be changed later on. Some recommended elements are also filled, as expected by the CESSDA Data Catalogue profile for DDI Codebook.

By default, the Codebook is assumed to have a single language for all elements. The argument `monolang` can be deactivated through the `"..."` gate, in which situation the appropriate elements will receive a default argument `xmlang = "en"`. For other languages, that argument can also be provided through the `"..."` gate.

One such DDI Codebook element is the `stdyDscr` (Study Description), with the associated mandatory children, for instance title, ID number, distributor, citation, abstract etc.

Value

A standard list element of class `"DDI"` with reserved component names.

Author(s)

Adrian Dusa

See Also

`addChildren` `getChildren` `showDetails`

Examples

Run examples

```
stdyDscr <- makeElement("stdyDscr", fill = TRUE)
```

Useful commands

```
showDetails("dataDscr")
```

```
Variable Description (optional, repeatable)
Description of variables.

This element does not have any specific attributes.

Examples:
There are no examples for this element.

Children:
- varGrp: Variable Group
- nCubeGrp: nCube Group
- var: Variable
- nCube: nCube
- notes: Notes and comments

Parents:
- codeBook: Codebook
```

element <dataDscr> (global)

Namespace: ddi:codebook:2_5
Type: dataDscrType
Content: complex, 8 attributes, 5 elements
Defined: globally in codebook.xsd; see XML source
Used: at 1 location

XML Representation Summary

```
<dataDscr
   ID                  = xs:ID
   xml-lang            = xs:NMTOKEN
   xml:lang            = xs:language
   source              = ("archive" | "producer") : "producer"
   elementVersion      = xs:string
   elementVersionDate  = (xs:dateTime | xs:date | xs:gYearMonth | xs:gYear)
   ddiLifecycleUrn     = xs:anyURI
   ddiCodebookUrn      = xs:anyURI
   >
   Content: varGrp*, nCubeGrp*, var*, nCube*, notes*
</dataDscr>
```

Content model elements (5):

nCube, nCubeGrp, notes, var, varGrp

Included in content model of elements (1):

codeBook

Known Usage Locations

- **Within global complexTypes (1):**

  codeBookType [ref]

Annotation

Variable Description

Description

Description of variables.

Useful commands

```
showAttributes("catgry")
```

```
Specific attributes, use globalAttributes() for the rest:
- missing (optional): (Y | N) : N
  Indicates whether this category group contains missing data or not.
- missType (optional): string
  Type of missing data, e.g., inap., don't know, no answer, etc.
- country (optional): string
  Allows for the denotation of country-specific category values
- sdatrefs (optional): IDREFS
  Records the ID values of all elements within the summary data description that apply to this
  category.
- access (optional): IDREFS
  ID values of all elements within the Data Access and etadata Access sections description that
  apply to this category.
- excls (optional): (true | false) : true
  Exclusiveness, should be set to "false" if the category can appear in more than one place in
  the classification hierarchy.
- catgry (optional): IDREFS
  References any child categories of this category element. Used to capture nested hierarchies of
  categories.
  ...
```

Useful commands

```
showExamples("catgry")
```

```
Examples:

<catLevel ID="Level1" levelnm="Broader sectors"/>

<catLevel ID="Level2" levelnm="Narrower sectors"/>

<catLevel ID="Level3" levelnm="Occupations"/>

<catgry ID="C1" catgry="C2" Level="Level1">
  <catValu>
    0
  </catValu>
  <labl>
    Management, professional and related occupations
  </labl>
</catgry>

  ...
```

Useful commands

```
showRelations("catgry")
```

```
Children:
- catValu: Category Value
- labl: Label
- txt: Descriptive Text
- catStat: Category Level Statistic
- mrow: Mathematical Row

Parents:
- var: Variable
```

Useful commands

```
showLineages("abstract")
```

```
codeBook/stdyDscr/stdyInfo/abstract
```

```
showLineages("txt")
```

```
codeBook/stdyDscr/stdyInfo/sumDscr/anlyUnit/txt
codeBook/stdyDscr/method/codingInstructions/txt
codeBook/stdyDscr/method/dataColl/collMode/txt
codeBook/stdyDscr/method/dataColl/collectorTraining/txt
codeBook/stdyDscr/method/anlyInfo/dataAppr/txt
...
codeBook/fileDscr/fileTxt/dataChck/txt
codeBook/dataDscr/nCube/anlysUnit/txt
codeBook/dataDscr/var/anlysUnit/txt
...

codeBook/dataDscr/varGrp/universe/txt
codeBook/dataDscr/var/txt
codeBook/dataDscr/varGrp/txt
codeBook/otherMat/txt
```

Practical example

Download a subset of the European Social Survey round 11 dataset (e.g. UK) and:

```
convert(
  "ESS11-subset.sav",
  to = "DDI",
  monolang = FALSE
)
```

(creates an XML file ~60k lines long)

```
codeBook <- getMetadata(
  "ESS11-subset.xml",
  ignore = "dataDscr"
)
names(codeBook)
```

```
[1] "docDscr"  "fileDscr" ".extra"
```

Practical example

```
abstract <- makeElement(
  "abstract",
  content = paste(
    "European Social Survey (ESS) is the most important academic",
    "survey in Europe..."
  ),
  attributes = c(xmlang = "en", source = "RODA")
)


stdyInfo <- makeElement("stdyInfo")
addChildren(abstract, to = stdyInfo)

stdyDscr <- makeElement("stdyDscr")
addChildren(stdyInfo, to = stdyDscr)

addChildren(stdyDscr, to = codeBook)
```

Practical example

```
abstract <- makeElement(
  "abstract",
  content = paste(
    "European Social Survey (ESS) is the most important academic",
    "survey in Europe..."
  ),
  attributes = c(xmlang = "en", source = "RODA")
)
```

or with a single chained command:

```
addChildren(
  makeElement("stdyDscr",
    children = makeElement("stdyInfo", children = abstract)
  ),
  to = codeBook
)
```

Practical example

Finally, update the XML Codebook file with the study description:

```
updateCodebook("ESS11-subset.xml", with = codeBook)
```

Many more things are possible, such as embedding the actual dataset into the XML Codebook:

```
ess <- convert("ESS11-subset.sav")

addChildren(
  makeNotes(ess),
  to = codeBook$fileDscr
)
```

and update it again

```
updateCodebook("ESS11-subset.xml", with = codeBook)
```

Thank you

dusa.adrian@unibuc.ro