Data quality tutorial

FishPi WP4

2016-05-19

Framework

Fishery data have to be inspected in order to detect errors before to use them in the stock assessment procedure (Chen 2003). Finding and correcting errors is one of the first tasks one needs to perform on a dataset in this case. Currently these checks are made at national level using mainly manual methods (based on graphs and numerical summary, see J. Vigneau and Mahévas (2007) for example). When the amount of data is large (as it will become with the implementation of the regional database) manual methods are (1) too time consuming and (2) are difficult to track in time (when and how the corrections were made). Hence automated procedures are needed and participate intrinsically to build the data quality.

This document is a tutorial related to the use of the R `fishPifct` package to assess data quality on fishery sampling data. The `fishPifct` package was developped for the work package 4 of the fishPi project (project DG-MARE 2014/19 WP4). Its main objectives are to provide to the end users a framework to assess the quality of sampling data related to fishery.

Data format specification

This framework concerns sampling data and leans on the `csPi` format in term of data structure. The `csPi` format is a format under development based on the `fishFrame` format. The fishFrame format is used in the Regional Database and by the COST packages (a collection of tools to deal with data compilation, COST (2006)). Its complete definition is given in Jansen et al. (2009). An update of this format, called csPi, is in discussion since 2014 (ICES 2014), and the version 2.1 of this format is used in this report. Tools to export `fishFrame` files in `csPi` as provided. And in order to insure a wide range of application, most of the packages functions works on `csPi` and `fishFrame` objects.

Methodology

This framework follows the recommandation of the reproducible research statment (Gentleman and Lang 2004). Consequently this report is self-consistent: the code used to process and to analyse the sampling data are embedded in the report itself. An effort was made to select computing tools who give to the users the ability to reproduce the analyses using only a computer and an internet connection (for installation purposes mainly). Therefore all the tools are open source software, available free of charges, and running on the three main operating systems available nowadays (Linux, Windows and Mac OS).

Software

Coding and analyses are carried out using the R environment (R Core Team 2016). R[1] is a free software environment for statistical computing and graphics. The reproducibility of the results presented in this report relies on the use of a dialect of the Mardown language called Pandoc for word processing using the Knitr R package. Markdown is a plain text formatting syntax designed so that it can optionally be converted to HTML using a tool by the same name. Pandoc[2] is a Markdown dialect which extends the conversion capability to word processing file (docx, doc and odt), html and pdf, among other formats. Pandoc understands a number

[1] http://www.r-project.org/.
[2] http://johnmacfarlane.net/pandoc/.

of markdown syntax extensions, including document metadata (title, author, date), footnotes, tables, figures and references. Knitr[3] is an R package (a set of functions extending the R capabilities). With this package, the R code used to process and analyze the data is included directly in the report. Results are then produced dynamically. This framework has demonstrated the capacity to improve the conduct and the presentation of data analysis in a way that another person can understand and replicate (Baumer et al. 2014).

For example, if the calculus of 1+1 is needed, the code to compute it is written in the report using special hooks, as in this simple example:

```
'''{r test00,warn=FALSE,cache=TRUE,echo=TRUE}
#comment: addition example.
1+1
'''
```

This code is evaluated during the compilation of the report by the knitr command and it prints the following result:

```
#comment: addition example.
1+1
```

```
## [1] 2
```

The result is 2. In this tutorial all the numerical values, tables and figures are produced following this procedure. The scripts and the report can be elaborated in a single integrated development environment (IDE), called Rstudio[4]. It includes a console and a syntax-highlighting editor that supports direct code execution, as well as tools for plotting, debugging and writing report. Consequently, all the tools and code presented here are already available to the end user.

Installation

This package is available
in the fihPifct repository on Github. The installation procedure is simple as :

```
install.packages("devtools")
library(devtools)
install_github("ldbk/fishPifct")
```

Issues

Technical problem support during the installation process (R version, missing packages...) is far beyond the scope of this tutorial. In case of problem, please contact your IT support.

Some users reported issues with the openxlsx package installation (needed to import and export csPi and csData object in excel file). Please read carefully the error messages R gives to you (the way to fix these errors are explained to you in these messages). The average procedure to fix them should be something like:

```
install.packages("installr")
installr::installr("Rtools")
```

During the installation, tick the PATH modification option. Then, restart your computer.

[3] http://yihui.name/knitr/.
[4] http://www.rstudio.com/.

COST library

If needed, COST related package (for windows) can be found here :

- https://dl.dropboxusercontent.com/u/6181692/COSTcore_1.4-0.zip
- https://dl.dropboxusercontent.com/u/6181692/COSTdbe_1.4-1.zip
- https://dl.dropboxusercontent.com/u/6181692/COSTeda_1.4.0.zip

and here for Unix system :

- https://dl.dropboxusercontent.com/u/6181692/COSTcore_1.4-0.tar.gz
- https://dl.dropboxusercontent.com/u/6181692/COSTdbe_1.4-1.tar.gz
- https://dl.dropboxusercontent.com/u/6181692/COSTeda_1.4.0.tar.gz

The COST manuel can be downloaded here :

- https://dl.dropboxusercontent.com/u/6181692/COST%20User%20Manual%20V1_1.pdf

Data

Format specification

In this tutorial, only the main characteristics of this format are illustrated. A detailed version of the `csPi` format specifications is given in ICES (2014) and in the help page of the `csPi` function.

`csPi` is an S4 object containing 10 slots :

```
library(pander);library(fishPifct)
pander(format_definition_csPi$slots,split.table=Inf)
```

slot_name	mandatory	definition_table
classVersion	TRUE	slot_classVersion
desc	FALSE	slot_desc
popData	FALSE	slot_popData
design	FALSE	slot_design
se	TRUE	slot_se
tr	TRUE	slot_tr
hh	FALSE	slot_hh
sl	FALSE	slot_sl
hl	FALSE	slot_hl
ca	FALSE	slot_ca

The slots `desc`, `popData`, `design` are not mandatory and serve as descriptive fields for future applications.

The slots `classVersion` provides the version number of the csPi format. This format is still in development, and keeping the format version will insure retrocompatibility with the future development of the package. The slots hold the sampling information : the sampling events description (`se`), the trip information (`tr`), the hauls caracteristics (`hh`), the species sampled (`sl`) and the correspondings length measurments (`hl`), and the biological parameters (`ca`). Each of these slots is a `data.frame` who lists the different parameters requested for each sample. Type of the vessel, its characteristic, the fishing location and the quantity landed, the scientific name of the sampled species, the length class of the fishes, the age, etc… are reported in these tables. These variables can be numeric, text or codelist. For each table, a group of variables represent the primary key and insure the links with the other tables. The next figure gives an overview of the structure of the table.

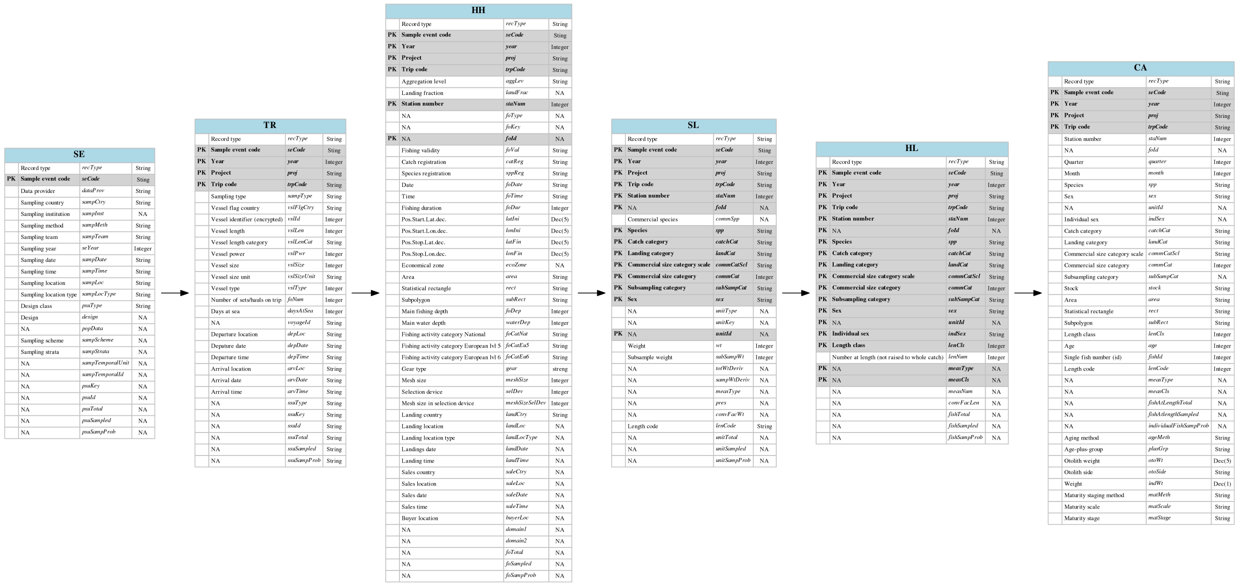


Figure 1: Overview of the csPi format

Example dataset

The data are generated based on the sole dataset coming from the COST package. The fishFrame COST format is exported in the csPi format using the function `csDataTocsPi`:

```r
library(fishPifct)
data(sole)
sole <- csDataTocsPi(sole.cs)
```

```
## No seObj provided. Trips are used as sampling events. Use only for testing !
```

```r
head(sole)
```

```
## An object of class "csPi"
## Slot "classVersion":
## [1] "2.1"
##
## Slot "desc":
## [1] "Commercial Sampling Data format for the fishPi project"
##
## Slot "popData":
## [1] "Named population data object"
##
## Slot "design":
```

```
## [1] "Design description"
##
## Slot "history":
## [1] "modification history"
##
## Slot "se":
##   recType  seCode dataProv sampCtry sampInst sampMeth sampTeam seYear
## 1      se  ARY178      FRA      FRA   Obsmer       NA   Obsmer   2006
## 2      se DIL1196      FRA      FRA   Obsmer       NA   Obsmer   2006
## 3      se DIL1197      FRA      FRA   Obsmer       NA   Obsmer   2006
## 4      se  ELR214      FRA      FRA   Obsmer       NA   Obsmer   2006
## 5      se  ELR219      FRA      FRA   Obsmer       NA   Obsmer   2006
## 6      se   FAD73      FRA      FRA   Obsmer       NA   Obsmer   2006
##   sampDate sampTime sampLoc sampLocType psuType design popData sampScheme
## 1       NA       NA      NA          NA      NA     NA      NA         NA
## 2       NA       NA      NA          NA      NA     NA      NA         NA
## 3       NA       NA      NA          NA      NA     NA      NA         NA
## 4       NA       NA      NA          NA      NA     NA      NA         NA
## 5       NA       NA      NA          NA      NA     NA      NA         NA
## 6       NA       NA      NA          NA      NA     NA      NA         NA
##   sampStrata sampTemporalUnit sampTemporalId psuKey psuId psuTotal
## 1         NA               NA             NA     NA    NA       NA
## 2         NA               NA             NA     NA    NA       NA
## 3         NA               NA             NA     NA    NA       NA
## 4         NA               NA             NA     NA    NA       NA
## 5         NA               NA             NA     NA    NA       NA
## 6         NA               NA             NA     NA    NA       NA
##   psuSampled psuSampProb
## 1          1           1
## 2          1           1
## 3          1           1
## 4          1           1
## 5          1           1
## 6          1           1
##
## Slot "tr":
##   recType  seCode year   proj trpCode sampType vslFlgCtry vslId vslLen
## 1      tr  ARY178 2006 Obsmer  ARY178        S        FRA    98     NA
## 2      tr DIL1196 2006 Obsmer DIL1196        S        FRA    85     NA
## 3      tr DIL1197 2006 Obsmer DIL1197        S        FRA    21     NA
## 4      tr  ELR214 2006 Obsmer  ELR214        S        FRA    42     NA
## 5      tr  ELR219 2006 Obsmer  ELR219        S        FRA    43     NA
## 6      tr   FAD73 2006 Obsmer   FAD73        S        FRA    41     NA
##   vslLenCat vslPwr vslSize vslSizeUnit vslType foNum daysAtSea voyageId
## 1      <NA>     NA      NA        <NA>       1    27         4     <NA>
## 2      <NA>     NA      NA        <NA>       1    30         5     <NA>
## 3      <NA>     NA      NA        <NA>       1     5         2     <NA>
## 4      <NA>     NA      NA        <NA>       1    56        13     <NA>
## 5      <NA>     NA      NA        <NA>       1    13        10     <NA>
## 6      <NA>     NA      NA        <NA>       3     3         4     <NA>
##   depLoc depDate depTime arvLoc arvDate arvTime ssuType ssuKey ssuId
## 1   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
## 2   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
## 3   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
```

```
## 4   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
## 5   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
## 6   <NA>    <NA>    <NA>   <NA>    <NA>    <NA>    <NA>   <NA>  <NA>
##   ssuTotal ssuSampled ssuSampProb
## 1     <NA>       <NA>        <NA>
## 2     <NA>       <NA>        <NA>
## 3     <NA>       <NA>        <NA>
## 4     <NA>       <NA>        <NA>
## 5     <NA>       <NA>        <NA>
## 6     <NA>       <NA>        <NA>
##
## Slot "hh":
##   recType seCode year   proj trpCode aggLev landFrac staNum foType foKey
## 1      hh ARY178 2006 Obsmer  ARY178      H     <NA>      1   <NA>  <NA>
## 2      hh ARY178 2006 Obsmer  ARY178      H     <NA>      2   <NA>  <NA>
## 3      hh ARY178 2006 Obsmer  ARY178      H     <NA>      3   <NA>  <NA>
## 4      hh ARY178 2006 Obsmer  ARY178      H     <NA>      4   <NA>  <NA>
## 5      hh ARY178 2006 Obsmer  ARY178      H     <NA>      5   <NA>  <NA>
## 6      hh ARY178 2006 Obsmer  ARY178      H     <NA>      6   <NA>  <NA>
##   foId foVal catReg sppReg     foDate foTime foDur   latIni   lonIni
## 1 <NA>     V    All    All 2006-04-03   <NA>   150 50.05360 1.623667
## 2 <NA>     V    All    All 2006-04-03   <NA>   180 51.19133 1.786333
## 3 <NA>     V    Non    Non 2006-04-03   <NA>   165 51.11667 1.672667
## 4 <NA>     V    All    All 2006-04-03   <NA>   195 51.03933 1.568667
## 5 <NA>     V    All    All 2006-04-03   <NA>   190 51.03933 1.666667
## 6 <NA>     V    Non    Non 2006-04-04   <NA>   180 51.11667 1.672667
##   latFin lonFin ecoZone   area rect subRect foDep waterDep      foCatNat
## 1     NA     NA    <NA> 27.7.d 29F1    <NA>    NA       40 7D-OTB-Merlan
## 2     NA     NA    <NA> 27.4.c 31F1    <NA>    NA       40 4C-OTB-Merlan
## 3     NA     NA    <NA> 27.4.c 31F1    <NA>    NA       40 4C-OTB-Merlan
## 4     NA     NA    <NA> 27.4.c 31F1    <NA>    NA       40 4C-OTB-Merlan
## 5     NA     NA    <NA> 27.4.c 31F1    <NA>    NA       40 4C-OTB-Merlan
## 6     NA     NA    <NA> 27.4.c 31F1    <NA>    NA       40 4C-OTB-Merlan
##   foCatEu5       foCatEu6 gear meshSize selDev meshSizeSelDev landCtry
## 1  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
## 2  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
## 3  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
## 4  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
## 5  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
## 6  OTB_DEF OTB_DEF_80_0_0 <NA>       80      0             NA      FRA
##   landLoc landLocType landDate landTime saleCtry saleLoc saleDate saleTime
## 1    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
## 2    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
## 3    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
## 4    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
## 5    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
## 6    <NA>        <NA>     <NA>     <NA>     <NA>    <NA>     <NA>     <NA>
##   buyerLoc domain1 domain2 foTotal foSampled foSampProb
## 1     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
## 2     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
## 3     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
## 4     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
## 5     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
## 6     <NA>    <NA>    <NA>    <NA>      <NA>       <NA>
```

```
##
## Slot "sl":
##   recType  seCode year   proj trpCode staNum foId commSpp         spp
## 1      sl DIL1197 2006 Obsmer DIL1197      1 <NA>    <NA> Solea solea
## 2      sl DIL1197 2006 Obsmer DIL1197      1 <NA>    <NA> Solea solea
## 3      sl DIL1197 2006 Obsmer DIL1197      2 <NA>    <NA> Solea solea
## 4      sl DIL1197 2006 Obsmer DIL1197      2 <NA>    <NA> Solea solea
## 5      sl DIL1197 2006 Obsmer DIL1197      3 <NA>    <NA> Solea solea
## 6      sl DIL1197 2006 Obsmer DIL1197      3 <NA>    <NA> Solea solea
##   catchCat landCat commCatScl commCat subSampCat  sex unitType unitKey
## 1      LAN     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
## 2      DIS     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
## 3      LAN     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
## 4      DIS     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
## 5      LAN     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
## 6      DIS     HUC         EU    <NA>       <NA> <NA>     <NA>    <NA>
##   unitId    wt subSampWt totWtDeriv sampWtDeriv measType pres convFacWt
## 1   <NA> 11000        NA       <NA>        <NA>     <NA> <NA>      <NA>
## 2   <NA> 10000      1560       <NA>        <NA>     <NA> <NA>      <NA>
## 3   <NA> 26321        NA       <NA>        <NA>     <NA> <NA>      <NA>
## 4   <NA> 12000      2570       <NA>        <NA>     <NA> <NA>      <NA>
## 5   <NA> 73000        NA       <NA>        <NA>     <NA> <NA>      <NA>
## 6   <NA>  7000      4217       <NA>        <NA>     <NA> <NA>      <NA>
##   lenCode unitTotal unitSampled unitSampProb
## 1      cm      <NA>        <NA>         <NA>
## 2      cm      <NA>        <NA>         <NA>
## 3      cm      <NA>        <NA>         <NA>
## 4      cm      <NA>        <NA>         <NA>
## 5      cm      <NA>        <NA>         <NA>
## 6      cm      <NA>        <NA>         <NA>
##
## Slot "hl":
##   recType  seCode year   proj trpCode staNum foId         spp catchCat
## 1      hl DIL1197 2006 Obsmer DIL1197      1 <NA> Solea solea      DIS
## 2      hl DIL1197 2006 Obsmer DIL1197      1 <NA> Solea solea      DIS
## 3      hl DIL1197 2006 Obsmer DIL1197      1 <NA> Solea solea      DIS
## 4      hl DIL1197 2006 Obsmer DIL1197      1 <NA> Solea solea      DIS
## 5      hl DIL1197 2006 Obsmer DIL1197      2 <NA> Solea solea      DIS
## 6      hl DIL1197 2006 Obsmer DIL1197      2 <NA> Solea solea      DIS
##   landCat commCatScl commCat subSampCat  sex unitId indSex lenCls lenNum
## 1     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    180      3
## 2     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    190      6
## 3     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    200      5
## 4     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    210      1
## 5     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    160      1
## 6     HUC         EU    <NA>       <NA> <NA>   <NA>   <NA>    170      1
##   measType measCls measNum convFacLen fishTotal fishSampled fishSampProb
## 1     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
## 2     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
## 3     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
## 4     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
## 5     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
## 6     <NA>    <NA>    <NA>       <NA>      <NA>        <NA>         <NA>
##
```

```
## Slot "ca":
##   recType seCode year   proj trpCode staNum foId quarter month         spp
## 1      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
## 2      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
## 3      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
## 4      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
## 5      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
## 6      ca     12 2006 BioPar      12    999 <NA>       2     4 Solea solea
##   sex unitId indSex catchCat landCat commCatScl commCat subSampCat stock
## 1   M   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
## 2   M   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
## 3   M   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
## 4   F   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
## 5   F   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
## 6   M   <NA>   <NA>      LAN     HUC       <NA>    <NA>       <NA>  <NA>
##     area rect subRect lenCls age fishId lenCode measType measCls
## 1 27.7.d 28E9    <NA>    330   5      1      cm     <NA>    <NA>
## 2 27.7.d 28E9    <NA>    340   7      2      cm     <NA>    <NA>
## 3 27.7.d 28E9    <NA>    320   7      3      cm     <NA>    <NA>
## 4 27.7.d 28E9    <NA>    320   4      4      cm     <NA>    <NA>
## 5 27.7.d 28E9    <NA>    350   7      5      cm     <NA>    <NA>
## 6 27.7.d 28E9    <NA>    340   9      6      cm     <NA>    <NA>
##   fishAtLengthTotal fishAtlengthSampled individualFishSampProb
## 1              <NA>                <NA>                   <NA>
## 2              <NA>                <NA>                   <NA>
## 3              <NA>                <NA>                   <NA>
## 4              <NA>                <NA>                   <NA>
## 5              <NA>                <NA>                   <NA>
## 6              <NA>                <NA>                   <NA>
##                                    ageMeth plusGrp otoWt otoSide indWt
## 1 Otoliths - slides with transmitted light       -    NA    <NA>   355
## 2 Otoliths - slides with transmitted light       -    NA    <NA>   360
## 3 Otoliths - slides with transmitted light       -    NA    <NA>   339
## 4 Otoliths - slides with transmitted light       -    NA    <NA>   416
## 5 Otoliths - slides with transmitted light       -    NA    <NA>   412
## 6 Otoliths - slides with transmitted light       -    NA    <NA>   411
##   matMeth matScale matStage
## 1  Visual      1-7        2
## 2  Visual      1-7        2
## 3  Visual      1-7        2
## 4  Visual      1-7        5
## 5  Visual      1-7        4
## 6  Visual      1-7        2
```

The csPi object is named sole in our example.

Handling csPi ojects

A collection of methods gives to the user the ability to explore and visualize a csPi objects:

```
methods(class="csPi")
```

```
## [1] dim     export  head    summary tail
```

```
## see '?methods' for accessing help and source code
```

Their behaviours are similar to the generic one (ie `dim` gives the dimension of all the `csPi` slots).

Import and export in spreadsheet

Fishing data rely usually on national database. Correction procedures in these systems can be a tedious work, not really in accordance to quick corrections (during working groups, to harmonize datasets between countries for example). Manual data corrections are difficult and spreadsheet is nowadays the common tools to correct locally the data. A local import/export procedure is available to export the `csPi` in excel file format. Thus, the user can use a spreadsheet to do some corrections in the tables and then import directly the corrected tables in a `csPi` object in R.

In this package the `import` and `export` functions do these transformation easily:

```
export(sole,file="sole.xlsx",type="xlsx")
```

```
## [1] "sole.xlsx"
```

```
#use a spreadsheet to open the sole.xlsx file and do some corrections if needed
#save the file, and import it in R with:
solecorrected<-importxlsx(file="sole.xlsx")
```

Data quality checks

Data structure checks

A seminal step in data quality is to check the structure of the data. The structure check includes the ordered verification of :

- the objects' slots: name, existence, mandatory or not.
- the slots' tables: dimension, variables names, mandatory or not, uniqueness of the primary keys if applicable.

- the tables'variables: their types - numeric (integer or real, lower and upper limits), character, codelist (a list of authorized values)-, nullable, mandatory or not.

The data structure definition is given for `csPi` objects by the list `format_definition_csPi`. This list is built from the excel file `format_definition_csPi.xlsx` in the data directory of the installation directory of the package. Providing the excel file gives to the end user the possibility to modify the data structure check (for example the lower and upper limits of the length class, or a limited list of métier).

Slot definition

A slot definition is a table reporting the characteristics of a given slot :

slot_name	mandatory	definition_table
base	TRUE	slot_base

Here the slot names `base` is mandatory and its definition is given by the table `slot_base`. During the structure check, each slot is checked against its definition given by the structure definition list.

Table definition

A table definition is a table reporting the characteristics of a given table. For example here, the first 8 lines of the `tr` table definition :

```
library(pander);library(fishPifct)
pander(format_definition_csPi$slot_tr[1:8,],split.table=Inf)
```

column_name	nullable	mandatory	pk	type_name	category
recType	FALSE	TRUE	FALSE	type_recType	codelist
seCode	FALSE	TRUE	TRUE	type_seCode	text
year	FALSE	TRUE	TRUE	type_year	numeric
proj	FALSE	TRUE	TRUE	type_proj	text
trpCode	FALSE	TRUE	TRUE	type_trpCode	text
sampType	FALSE	TRUE	FALSE	type_sampType	codelist
vslFlgCtry	FALSE	TRUE	FALSE	type_ctry	codelist
vslId	FALSE	FALSE	FALSE	type_vslId	text

Each table's column is checked against its definition. For example, the `trpCode` variable has to be non nullable, is mandatory and is part of the primary key of the `tr` table. It is a text variable (`category`), and its category definition is referenced in the `type_trpCode` of the definition file (or the excel sheet with this name).

Variable checks

After the table definition, each variable are checked according to their types. For example in the previous table `vslFlagCtry` is non nullable, mandatory and is not included in the primary key. The variable's type is a codelist, and the corresponding authorized value are registered in the `codelist_type` list of the format description, namely the list `codelist_ctry` (here the first 10 lines):

```
pander(format_definition_csPi$codelist_ctry[1:10,],split.table=Inf)
```

CODE	DESCRIPTION
ABW	Aruba
AFG	Afghanistan
AGO	Angola
AIA	Anguilla
ALA	Åland Islands
ALB	Albania
AND	Andorra
ARE	United Arab Emirates
ARG	Argentina
ARM	Armenia

This list is the list of the ISO 3166-1 alpha-3 country codes. Limiting this list strengths the data quality check, according to the end user needs.

For the variables with a numeric type, the `numeric_type` list of the data definition brings information related to the numerical limits and if the numbers are integer (number of samples, age…) or real (probability…).

For example here, the first 8 lines of the `numeric_type` table definition:

```
library(pander);library(fishPifct)
pander(format_definition_csPi$numeric_type[1:8,],split.table=Inf)
```

type_name	is_integer	min	max
type_year	TRUE	1900	2020
type_psuTotal	TRUE	0	1e+07
type_psuSampled	TRUE	0	2000
type_psuSampProb	FALSE	0	1
type_vslLen	TRUE	3	160
type_vslPwr	TRUE	4	8500
type_vslSize	TRUE	1	2500
type_foNum	TRUE	1	300

In this example, `year` is an integer between 1900 and 2020. As previously stated the modification of the data structure is open to the end user needs.

Notes

The data structure checks were developped by the sister project of fishPi related to the Mediterranean area, for fishFrame object (https://git.outils-is.ird.fr/billet/SDEFQuality/wikis/home). Consequently, this data structure check is applicable to any object structure, and it can be extended to landings or effort file in a near future for example.

Outputs

The results of the data structure checks are given in a report summarizing all the checks, if these checks pass, and why. Using the `sole` dataset previously loaded:

```
#generating a report in an R object
structurecheck<-validateData(obj=sole,formatDb=format_definition_csPi,report="list")
```

The meta information related to the check are:

```
pander(structurecheck$meta,split.table=Inf)
```

parameter	value
format_name	csPi
format_version	2.1
validate_date	2016-05-19 04:08:50
dataset_container	object
format_container	object

The 10 first lines of the slots checks are:

```
pander(structurecheck$struct[1:10,],split.table=Inf)
```

slot	column	test	result	message
classVersion	NA	Slot exists ?	OK	Found
classVersion	classVersion	Column exists ?	ERROR	Not found
desc	NA	Slot exists ?	OK	Found
desc	desc	Column exists ?	ERROR	Not found
popData	NA	Slot exists ?	OK	Found
popData	popData	Column exists ?	ERROR	Not found
design	NA	Slot exists ?	OK	Found
design	design	Column exists ?	ERROR	Not found
se	NA	Slot exists ?	OK	Found
se	recType	Column exists ?	OK	Found

The 10 first lines of the variables checks are:

```
pander(structurecheck$data[1:10,],split.table=Inf)
```

slot	column	test	result	message
se	recType	is valid code list ?	OK	All values are valid codes
se	recType	is null ?	OK	All values are not null
se	seCode	is text ?	OK	All values are text
se	seCode	is null ?	OK	All values are not null
se	dataProv	is text ?	OK	All values are text
se	dataProv	is null ?	OK	All values are not null
se	sampCtry	is valid code list ?	OK	All values are valid codes
se	sampCtry	is null ?	OK	All values are not null
se	sampInst	is text ?	OK	All values are text
se	sampInst	is null ?	OK	All values are not null

The tables are explicits and doesn't need any comments. To generate a complete report in pdf or html format :

```
#generating a pdf report
renderValidationReport(obj=sole,formatDb=format_definition_csPi,
               title="test",reportFormat="pdf")
```

```
## Report generated [/tmp/Rtmp74bQdi/dataValidationReport_20160519_040851_692d273a0761.pdf]

## [1] "/tmp/Rtmp74bQdi/dataValidationReport_20160519_040851_692d273a0761.pdf"
```

```
#a copy of this report can be found in
system.file('data',
'dataValidationReport_20160518_235140_29d51c808f9b.pdf',
        package='fishPifct')
```

```
## [1] "/home/moi/R/x86_64-pc-linux-gnu-library/3.3/fishPifct/data/dataValidationReport_20160518_235140_
```

Consistency check

In this section, the consistency of the information between the 'csPi' slots is checked, e.g. identification of trips without fishing operations. To do so, the function `consistency` performs hierachical anti jointure between related table and generates a simple table reporting the `trpCode` who have to be checked between the tables:

```
#consistency check generating a pdf report
consistencycheck<-consistency(sole)
pander(consistencycheck,split.table=Inf)
```

test	message	check
tr->se	0 tr records have no correspondings se records	orphans tr trpCode:
hh->tr	0 hh records have no correspondings tr records	orphans hh trpCode:
sl->hh	0 sl records have no correspondings hh records	orphans sl trpCode:
hl->sl	0 hl records have no correspondings sl records	orphans hl trpCode:

In our sole example, no consistency errors were detected.

Outliers detection

The literature on outliers is extensive, and cover all the areas of science, but determining whether or not an observation is an outlier is ultimately a subjective exercise and hence makes automation a difficult task. Here we will use the definition of Barnett and Lewis (1994) for outlier: "Indicate that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". Outlier detection methods can be divide between univariate methods (looking at only one variable) and multivariate methods (looking at more than one variable and their relationships). For example univariate methods spot observations reported in tons instead of kilos in landings, while multivariate methods can identify wrong weigths in a size-weight relationship. Then outlier detection methods can be categorized between parametric (statistical) methods and non-parametric methods that are model free. Statistical parametric methods either assume a known underlying distribution of the data or, at least, they are based on statistical estimates of unknown distribution parameters. Observations that deviate from the model assumptions are flagged as outliers. Here we focus on two generic non parametric methods for numerical and non numerical univariate data. The function `outliers` do the outliers detection for a `csPi` object.

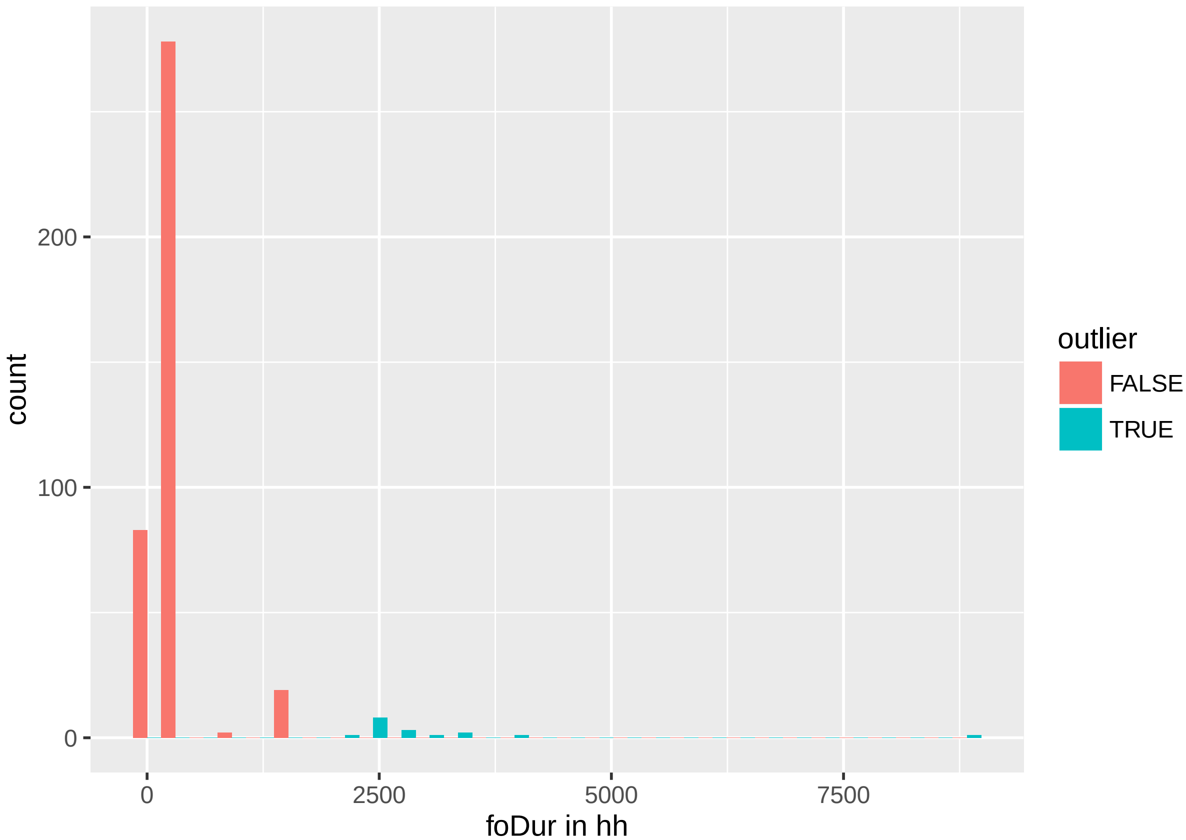
Numeric variables

The adjusted outlyingness index is used to detect outliers. It's a non parametric methods, adapted to skewed data. The function `adjOutliness` of the package `robustBase` is used. More details of this method can be found in the help page of this function. An example on the fishing duration (variable `foDur` of slot `hh`):

```
tabaoutlier<-outliers(sole,slot="hh",var="foDur")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 48 rows containing non-finite values (stat_bin).
```



High fishing duration values are flagged as outliers, as presented in the figure. The function output gives to the user the complete lines who includes the outliers:

```
#10 first lines and 5 first columns of the outliers
pander(tabaoutlier[1:10,1:5],split.table=Inf)
```

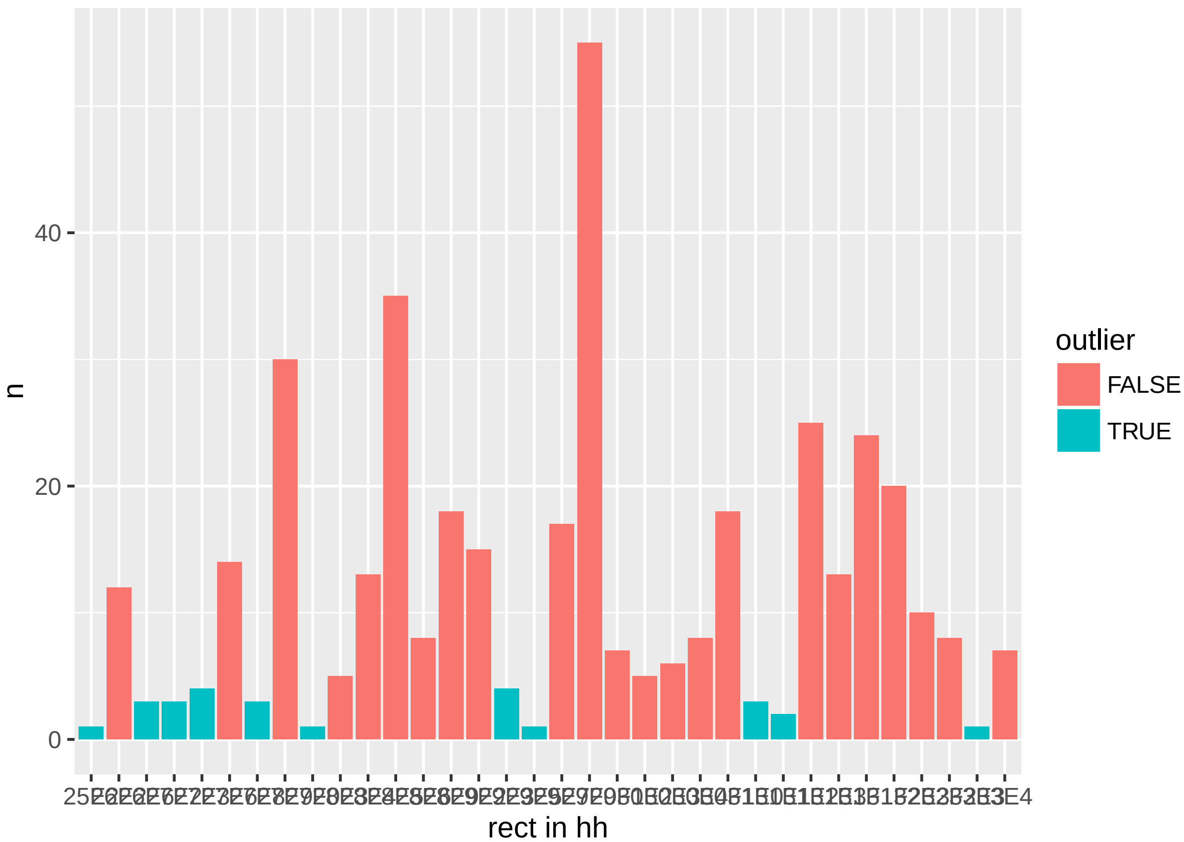
	recType	seCode	year	proj	trpCode
63	hh	ELR214	2006	Obsmer	ELR214
274	hh	MAC3	2006	Obsmer	MAC3
275	hh	MAC3	2006	Obsmer	MAC3
276	hh	MAC3	2006	Obsmer	MAC3
277	hh	MAC3	2006	Obsmer	MAC3
278	hh	MAC3	2006	Obsmer	MAC3
279	hh	MAC3	2006	Obsmer	MAC3
280	hh	MAC3	2006	Obsmer	MAC3
281	hh	MAC3	2006	Obsmer	MAC3
282	hh	MAC3	2006	Obsmer	MAC3

Text and codelist variables

For a non-numerical variable, the outliers are detected using the occurence of the modality of the value, expressed in percentage and a treshold (by default 1%). If a modality is expressed less than this treshold, an outlier is considered detected. The treshold can be fixed by the user. Here an example using the statistical rectangle fished :

```r
tabaoutlier<-outliers(sole,slot="hh",var="rect")
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



Rare fished rectangle are flagged as outliers. The function output gives to the user the complete lines who includes the outliers:

```r
#10 first lines and 5 first columns of the outliers
pander(tabaoutlier[1:10,1:5],split.table=Inf)
```

	recType	seCode	year	proj	trpCode
65	hh	ELR214	2006	Obsmer	ELR214
68	hh	ELR214	2006	Obsmer	ELR214
77	hh	ELR214	2006	Obsmer	ELR214
78	hh	ELR214	2006	Obsmer	ELR214
79	hh	ELR214	2006	Obsmer	ELR214
81	hh	ELR214	2006	Obsmer	ELR214
82	hh	ELR214	2006	Obsmer	ELR214
83	hh	ELR214	2006	Obsmer	ELR214
90	hh	ELR214	2006	Obsmer	ELR214
110	hh	ELR214	2006	Obsmer	ELR214

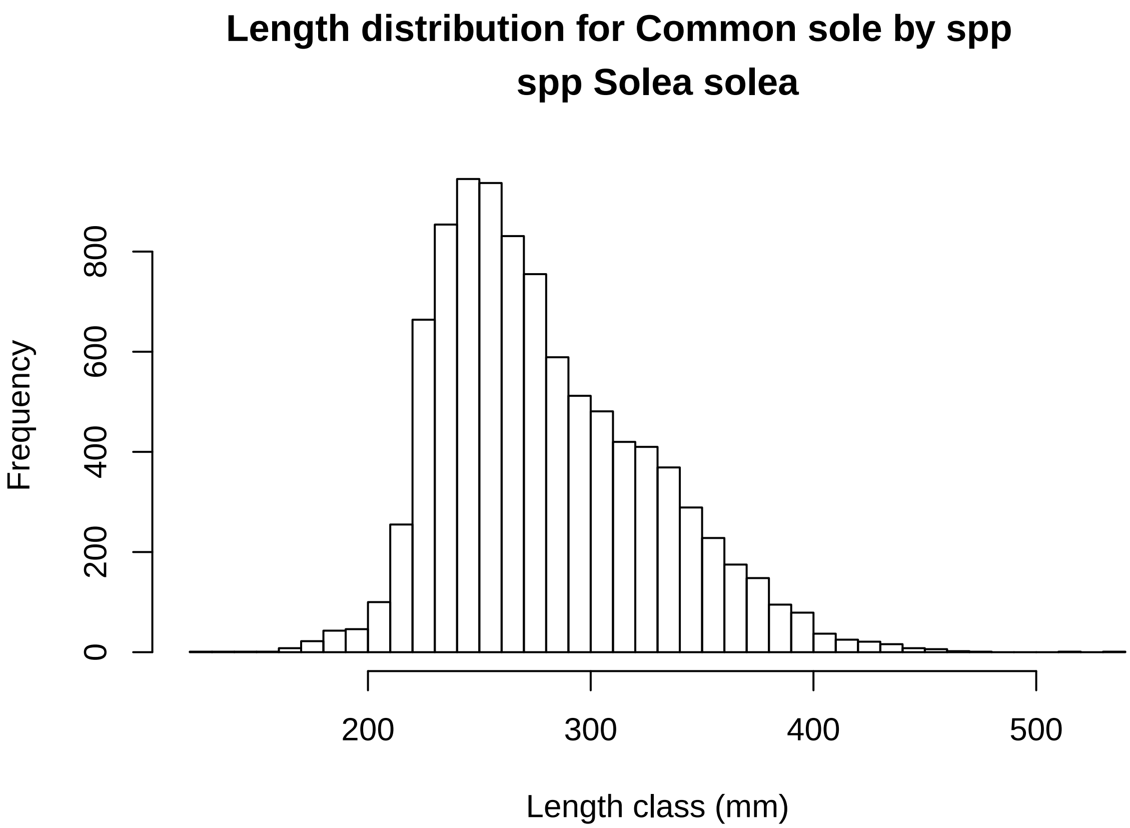
Plots

Maps

Generic functions

The function `lengthHist` plots histograms of the length frequency data from the hl table of csPi object.

```
lengthHist(sole)
```



The function `agelenPlot` plots age given length from the ca table of a `csPi`object.

```
agelenPlot(sole)
```

```
## Warning in agelenPlot(sole): Only LAN fraction present in data
```

About this vignette

This vignette was built using the vignette engine `knitr::rmarkdown` in the **knitr** package. You can find the source in the fihPifct repository on Github, or if the `fishPifct` package is installed on your computer:

```
system.file('doc', 'tutorial.Rmd', package='fishPifct')
```

```
## [1] "/home/moi/R/x86_64-pc-linux-gnu-library/3.3/fishPifct/doc/tutorial.Rmd"
```

References

Barnett, V, and T Lewis. 1994. *Outliers in Statistical Data. 3rd Ed.* Wiley.

Baumer, Ben, Mine Cetinkaya-Rundel, Andrew Bray, Linda Loi, and Nicholas J. Horton. 2014. "R Markdown: Integrating a Reproducible Analysis Tool into Introductory Statistics." *Technology Innovations in Statistics Education* 8 (1).

Chen, Y. 2003. "Quality of Fisheries Data and Uncertainty in Stock Assessment." *Scientia Marina* 67 (1): 75–87.

COST. 2006. "Studies and Pilot Projects for Carrying Out the Common Fisheries Policy Call for Proposal Ref : FISH/2006/15 – Lot 2 Project No :SI2.467814 Project Acronym :COST." EUROPEAN COMMISSION.

Gentleman, Robert, and Duncan Temple Lang. 2004. "Statistical Analyses and Reproducible Research." Bioconductor Project Working Papers. Working Paper 2. http://www.bepress.com/bioconductor/paper2.

ICES. 2014. "Report of the Workshop on Developing the RDB Data Format for Design Based Sampling and Estimation (WKRDB 2014-1) 27-31 October 2014 Aberdeen, Scotland, United-Kingdom ICES WKRDB 2014-1 REPORT 2014 ICES ACOM c OMMITTEE ICES CM 2014:68." ICES.

Jansen, Teunis, Henrik Degel, Joel Vigneau, and Ernesto Jardim. 2009. "Definition of Standard Data-Exchange Format for Sampling, Landings, and Effort Data Commercial Fisheries." ICES COOPERATIVE RESEARCH REPORT 296. ICES.

R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Vigneau, Joël, and Stéphanie Mahévas. 2007. "Detecting Sampling Outliers and Sampling Heterogeneity When Catch-at-Length Is Estimated Using the Ratio Estimator." *ICES Journal of Marine Science: Journal Du Conseil* 64 (5): 1028–32. doi:10.1093/icesjms/fsm077.