FLFleet Tutorial

Clara Ulrich
<clu@aqua.dtu.dk>
&
François Bastardie
<fba@aqua.dtu.dk>

2. December 2010

Table of Contents

1 Introduction

`FLFleet` is a rather complex and multi-level S4 Class in **FLR**, embedding the information necessary to fully describe a fishing fleet in a typical mixed-fisheries context, e.g. when several types of vessels exploit several stocks using several fishing gears over time. A simpler structure could be preferred in simpler single-species fisheries, but of course, the degree of complexity to include in a `FLFleet` object depends of the question asked and the data available.

Central to the fleet descriptor is the explicit representation of both fishing vessels and their activity, where the former are described in terms of fleets, or fleet segments, and the latter is described through the notion of métier. We follow here the definitions used by the Data Collection Framework of the European Union (2008) : *A Fleet segment* is a group of vessels with the same length class and predominant fishing gear during the year. Vessels may have different fishing activities during the reference period, but will generally be assigned to only one fleet segment. *A Métier* is a group of fishing operations targeting a similar (assemblage of) species, using similar gear, during the same period of the year and/or within the same area, and which are characterized by a similar exploitation pattern.

2 Section

2.1 Structure of the FLFleet object

`FLFleet` is part of the FLCore library so you need first to load the FLCore library in the R environment.

```
> library(FLCore)
```

`FLFleet` is defined along a three-leveled tree structure. First, the root of `FLFleet` contains data at the vessel/fleet segment levels, independent of activity and catches, such as total effort, capacity, fixed costs and crew share.

Then, data dealing with the various types of activity of the fleet are dealt with at the metier level using `FLMetier` class, which includes slots for proportion of effort and variable costs. The `FLFleet` class has a slot named *metiers* which holds an object of class `FLMetiers` (a list of `FLMetier` objects).

Finally, the catch information related to the various species or stocks within each metier are dealt with a `FLCatch` class. The `FLMetier` class has a slot named *catches* which holds an object of `FLCatches` (a list of `FLCatch` objects)

The tree structure allows a flexible use of the `FLFleet` object, if for example not all species are caught by all metiers or if the dimensions are not the same across branches. However, it can also make it a

cumbersome and memory intensive object to work with.

In addition to the standard name, description and range slots, the `FLFleet` class contains the following slots:

- `effort` FLQuant holding the fishing effort
- `fcost` FLQuant holding the fixed costs
- `capacity` FLQuant holding the capacity of the fishing fleet
- `crewshare` FLQuant holding the crew share - total or per unit capacity
- `metiers` FLMetiers holding a list of FLMetier objects

Which can be inspected by:

```
> getSlots("FLFleet")
     effort       fcost    capacity   crewshare     metiers        name
  "FLQuant"   "FLQuant"   "FLQuant"   "FLQuant" "FLMetiers" "character"
       desc       range
"character"   "numeric"
```

In order to allow for age or length based catches within fleets where additional economic data is desirable, the first dimension of `FLQuants` in the list of `FLCatches` is allowed to differ from the first dimension in the effort, capacity and cost slots.
Now the structure of the `FLMetier`:

```
> getSlots("FLMetier")
       gear    effshare       vcost     catches        name        desc
"character"   "FLQuant"   "FLQuant" "FLCatches" "character" "character"
      range
  "numeric"
```

In addition to the standard name, description and range slots, the `FLMetier` class contains the following slots:

- `gear` character : name of the gear
- `effshare` FLQuant : proportion of total fleet effort allocated to this metier
- `vcost` FLQuant : variable costs per unit of effort
- `catches` FLCatches : a list of FLCatch objects

And finally the structure of the `FLCatch`:

```
> getSlots("FLCatch")
    landings   landings.n  landings.wt landings.sel     discards   discards.n
   "FLQuant"    "FLQuant"    "FLQuant"    "FLQuant"    "FLQuant"    "FLQuant"
discards.wt discards.sel      catch.q        price         name         desc
   "FLQuant"    "FLQuant"    "FLQuant"    "FLQuant"  "character"  "character"
       range
   "numeric"
```

The `FLCatch` class is similar to `FLStock` but is obviously concerned with catches rather than stocks. The `FLCatch` class stores stock or species specific information on landings, discards and catches. The slots of the class (in addition to the standard name, desc, range slots) are:

- `landings` FLQuant holding the total landings
- `landings.n` FLQuant holding the landings in numbers by quant
- `landings.wt` FLQuant holding the weight of landings by quant
- `landings.sel` FLQuant holding the selectivity of the landings by quant
- `discards` FLQuant holding the total discards in weight for all quants
- `discards.n` FLQuant holding the discards in numbers by quant
- `discards.wt` FLQuant holding the weight of the discards by quant
- `discards.sel` FLQuant holding the selectivity of the discards by quant
- `catch.q` FLQuant holding the catchability coefficient
- `price` FLQuant holding the price by age and weight

There are no slots for `catch`, `catch.n`, `catch.wt` and `catch.sel`, as it can easily be calculated from the landings and discards information held in the object.

An example of `FLFleet` object is loaded in FLCore, the `bt4` object:

```
> data(bt4)
> summary(bt4)
An object of class "FLFleet"

Name: beam trawl fleet
Description: Example of an FLFleet
Range: min max pgroup minyear maxyear
0 0 NA 1957 2001
Quant: age

effort        : [ 1 45 1 1 1 1 ], units =  NA
fcost         : [ 1 45 1 1 1 1 ], units =  NA
capacity      : [ 1 45 1 1 1 1 ], units =  NA
crewshare     : [ 1 45 1 1 1 1 ], units =  NA

Metiers:
TBB :
ple : [ 15 45 1 1 1 1 ]
sol : [ 10 45 1 1 1 1 ]
```

The summary function is then a practical way for checking the dimensions [in brackets] of each of the FLQuant slots, e.g. in this case, the effort slot has 1 quant, 45 years, 1 unit, 1 season, 1 area, 1 iteration, while the FLmetier TBB has 2 FLcatch objects i.e. ple (for plaice) and sol (for sole) and the FLquant slots of these FLcatch have 15 quant (ages), 45 years, 1 unit, 1 season, 1 area and 1 iteration for ple, where sol has only 10 in the first dimension.

2.2 List classes

As composite objects, all the above also have `list` classes:

- `FLFleets`
- `FLMetiers`
- `FLCatches`

These classes can be used to store a number of composite objects. This can be used when modelling multifleet and multispecies systems, or to collate runs of models. These classes also have constructors and methods. To collate the two fleets `bt4` and `bt4`, one could use the `FLFleets` class, with its constructor

```
> fleets4 <- FLFleets(bt4, bt4)
> name(fleets4[[1]]) <- "my fleet 1"
> name(fleets4[[2]]) <- "my fleet 2"
> names(fleets4) <- lapply(fleets4, name)
> lapply(fleets4, name)
$`my fleet 1`
[1] "my fleet 1"

$`my fleet 2`
[1] "my fleet 2"
```

Now we have a object with two fleets. The individual fleets can be accessed using the normal accessor:

```
> a.fleet <- fleets4[[1]]
```

The number of fleets in this object can be determined using the `length` function:

```
> length(fleets4)
[1] 2
```

For these list objects, the `lapply` function will usually be your best friend to manipulate several objects at once

```
> lapply(fleets4, "effort")
$ my fleet 1
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age    1957   1958   1959   1960   1961   1962   1963   1964   1965   1966
  all 0.2922 0.3608 0.3657 0.3440 0.3967 0.3669 0.5092 0.5159 0.6000 0.4904
     year
age    1967   1968   1969   1970   1971   1972   1973   1974   1975   1976
  all 0.6825 0.6957 0.6909 0.6367 0.6594 0.6532 0.7105 0.6753 0.6603 0.5671
     year
age    1977   1978   1979   1980   1981   1982   1983   1984   1985   1986
  all 0.6192 0.7236 0.6592 0.5993 0.5945 0.6968 0.7250 0.8168 0.7684 0.7435
     year
age    1987   1988   1989   1990   1991   1992   1993   1994   1995   1996
  all 0.6814 0.9264 0.6842 0.6136 0.7600 0.8004 0.8528 0.9007 0.9348 0.9764
     year
age    1997   1998   1999   2000   2001
  all 1.0080 0.9079 1.1441 0.8264 0.6617

units:  NA

$ my fleet 2
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age    1957   1958   1959   1960   1961   1962   1963   1964   1965   1966
  all 0.2922 0.3608 0.3657 0.3440 0.3967 0.3669 0.5092 0.5159 0.6000 0.4904
     year
age    1967   1968   1969   1970   1971   1972   1973   1974   1975   1976
  all 0.6825 0.6957 0.6909 0.6367 0.6594 0.6532 0.7105 0.6753 0.6603 0.5671
     year
age    1977   1978   1979   1980   1981   1982   1983   1984   1985   1986
  all 0.6192 0.7236 0.6592 0.5993 0.5945 0.6968 0.7250 0.8168 0.7684 0.7435
     year
age    1987   1988   1989   1990   1991   1992   1993   1994   1995   1996
  all 0.6814 0.9264 0.6842 0.6136 0.7600 0.8004 0.8528 0.9007 0.9348 0.9764
     year
age    1997   1998   1999   2000   2001
  all 1.0080 0.9079 1.1441 0.8264 0.6617

units:  NA
```

2.3 Creating a FLFleet object

To create a new `FLFleet` object, just type:

```
> empty.fleet <- FLFleet()
> summary(empty.fleet)
An object of class "FLFleet"

Name:
Description:
Range: min max minyear maxyear
NA NA 1 1
Quant: quant

effort        : [ 1 1 1 1 1 1 ], units =  NA
fcost         : [ 1 1 1 1 1 1 ], units =  NA
capacity      : [ 1 1 1 1 1 1 ], units =  NA
crewshare     : [ 1 1 1 1 1 1 ], units =  NA

Metiers:
NA :
NA : [ 1 1 1 1 1 1 ]
```

But it is more useful to first determine the dimensions of the fleet data (the `effort`, `capacity`, `vcost`, `fcost` and `crewshare` slots) and catch data (the `catches` slot). This can be done by creating an `FLQuant` object of the correct size, using the `FLQuant` constructor and using it to initialise an `FLCatch` object that in turn can be used to initialise an `FLFleet` object:

```
> CatchDims <- FLQuant(NA, dim = c(10, 5, 1, 1, 1, 1))
> ot4 <- FLFleet(FLCatch(CatchDims))
> summary(ot4)
An object of class "FLFleet"

Name:
Description:
Range: min max minyear maxyear
1 10 1 5
Quant: quant

effort        : [ 1 1 1 1 1 1 ], units =  NA
fcost         : [ 1 1 1 1 1 1 ], units =  NA
capacity      : [ 1 1 1 1 1 1 ], units =  NA
crewshare     : [ 1 1 1 1 1 1 ], units =  NA

Metiers:
NA :
NA : [ 10 5 1 1 1 1 ]
```

As only a single `catchname` is given, `ot4` only holds data on one catch species and consequently the slot `catches` contains only 1 `FLCatch` object.

To create a fleet with two catches and one gear, simply give the appropriate information to the constructor:

```
> CatchDims <- FLQuant(NA, dim = c(10, 5, 1, 1, 1, 1))
> ot4 <- FLFleet(FLCatches(FLCatch(name = "sp1", CatchDims), FLCatch(name =
"sp2",
+     CatchDims)))
> summary(ot4)
An object of class "FLFleet"

Name:
Description:
Range: min max minyear maxyear
1 10 1 5
Quant: quant

effort        : [ 1 1 1 1 1 1 ], units =  NA
fcost         : [ 1 1 1 1 1 1 ], units =  NA
capacity      : [ 1 1 1 1 1 1 ], units =  NA
crewshare     : [ 1 1 1 1 1 1 ], units =  NA

Metiers:
NA :
sp1 : [ 10 5 1 1 1 1 ]
sp2 : [ 10 5 1 1 1 1 ]
```

A more complicated example involves two metiers, each with two catch objects, which also do not have the same numbers of ages. For complicated fleet objects it is often easier to construct it in stages. First of all create the catch objects:

```
> catchDims1 <- FLQuant(NA, dim = c(10, 5, 1, 1, 1, 1))
> catchDims2 <- FLQuant(NA, dim = c(8, 5, 1, 1, 1, 1))
> catchDims3 <- FLQuant(NA, dim = c(9, 5, 1, 1, 1, 1))
> catches1 <- FLCatches(FLCatch(name = "sp1", catchDims1), FLCatch(name = "sp2",
+     catchDims1))
> catches2 <- FLCatches(FLCatch(name = "sp1", catchDims1), FLCatch(name = "sp3",
+     catchDims3))
```

Now create the metiers that have these catches:

```
> metiers <- FLMetiers(FLMetier(name = "met1", catches1), FLMetier(name = "met2",
+     catches2))
> names(metiers) <- lapply(metiers, name)
> names(metiers)
[1] "met1" "met2"
```

And finally create the fleet, also setting the effort slot:

```
> effort.quant <- FLQuant(NA, dim = c(1, 5, 1, 1, 1, 1))
> a.fleet <- FLFleet(effort = effort.quant, metiers)
> summary(a.fleet)
An object of class "FLFleet"

Name:
Description:
Range: min max minyear maxyear
1 9 1 5
Quant: quant

effort        : [ 1 5 1 1 1 1 ], units =  NA
fcost         : [ 1 5 1 1 1 1 ], units =  NA
capacity      : [ 1 5 1 1 1 1 ], units =  NA
crewshare     : [ 1 5 1 1 1 1 ], units =  NA

Metiers:
met1 :
sp1 : [ 10 5 1 1 1 1 ]
sp2 : [ 10 5 1 1 1 1 ]
met2 :
sp1 : [ 10 5 1 1 1 1 ]
sp3 : [ 9 5 1 1 1 1 ]
```

For your information, it is also possible create a `FLFleet` object just by converting a `FLStock` object:

```
> data(ple4)
> a.fleet <- as(ple4, "FLFleet")
> summary(a.fleet)
An object of class "FLFleet"

Name:
Description:
Range: min max pgroup minyear maxyear minfbar maxfbar
1 10 10 1957 2008 2 6
Quant: age

effort        : [ 1 52 1 1 1 1 ], units =  NA
fcost         : [ 1 52 1 1 1 1 ], units =  NA
capacity      : [ 1 52 1 1 1 1 ], units =  NA
crewshare     : [ 1 52 1 1 1 1 ], units =  NA

Metiers:
NA :
Plaice in IV : [ 10 52 1 1 1 1 ]
```

2.4 FLFleet accessors

Because of the complex structure of the `FLFleet` object, the accessors are of capital importance here to keep the object manageable. Without accessors, getting a slot can be quite lengthy:

```
> bt4@metiers[[1]]@catches[["sol"]]@landings.n[, ac(1992:2001)]
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

    year
age  1992  1993  1994  1995  1996  1997  1998  1999  2000  2001
  1    980    54   718  4801   172  1590   244   287  2351   884
  2   6832 50451  7804 12767 18824  6047 56648 15762 15073 25846
  3  44378 16768 87403 16822 16190 23651 15141 72470 32738 21595
  4  16204 31409 13550 68571 16964  7325 14934  8187 42803 19876
  5  38319 13869 18739  6308 27257  5108  3496  6111  3288 16730
  6   2477 24035  5711  7307  3858 12793  1941  1212  2477  1427
  7   3041  1489 11310  1995  4780  1201  4768   664   804   834
  8    741  1184   464  6015   943  2326   794  1984   435   274
  9    399   461   916   295  3305   333  1031   331   931   168
  10  1180   842   908   668   988  1688   846   812   714   724

units:  NA
```

Fortunately, a number of accessors have been created to ease your life. To access the fleet level slots (`effort, fcost, capacity, crewshare, metiers,` etc.) only the fleet name is required:

```
> effort(bt4)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957   1958   1959   1960   1961   1962   1963   1964   1965   1966
  all 0.2922 0.3608 0.3657 0.3440 0.3967 0.3669 0.5092 0.5159 0.6000 0.4904
     year
age   1967   1968   1969   1970   1971   1972   1973   1974   1975   1976
  all 0.6825 0.6957 0.6909 0.6367 0.6594 0.6532 0.7105 0.6753 0.6603 0.5671
     year
age   1977   1978   1979   1980   1981   1982   1983   1984   1985   1986
  all 0.6192 0.7236 0.6592 0.5993 0.5945 0.6968 0.7250 0.8168 0.7684 0.7435
     year
age   1987   1988   1989   1990   1991   1992   1993   1994   1995   1996
  all 0.6814 0.9264 0.6842 0.6136 0.7600 0.8004 0.8528 0.9007 0.9348 0.9764
     year
age   1997   1998   1999   2000   2001
  all 1.0080 0.9079 1.1441 0.8264 0.6617

units:  NA
```

To access metier level slots (e.g. `effshare`) the fleet name and metier name or number are required:

```
> effshare(bt4, 1)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970
  all 1    1    1    1    1    1    1    1    1    1    1    1    1    1
     year
age   1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
  all 1    1    1    1    1    1    1    1    1    1    1    1    1    1
     year
age   1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
  all 1    1    1    1    1    1    1    1    1    1    1    1    1    1
     year
age   1999 2000 2001
  all 1    1    1

units:  NA
```

This respectively returns a `FLQuant` of the effort share.

To access catch level slots (e.g. `landings.n`, `catch.q` etc) the fleet name, metier name or number, and catch name or number are required):

```
> landings(bt4, 1, 1)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957   1958   1959   1960   1961   1962   1963   1964   1965   1966
  all  63542  68983  77615  86962  84667  90500 105091 109718  96594 100012
     year
age   1967   1968   1969   1970   1971   1972   1973   1974   1975   1976
  all 106704 108376 114962 133769 110296 119461 124128 108531 102161 110857
     year
age   1977   1978   1979   1980   1981   1982   1983   1984   1985   1986
  all 118996 118209 145596 138078 137342 153587 144941 158625 163116 167406
     year
age   1987   1988   1989   1990   1991   1992   1993   1994   1995   1996
  all 155608 156864 172327 158685 153625 127510 119907 113368  99003  82952
     year
age   1997   1998   1999   2000   2001
  all  83720  72685  81804  83559  82689

units:  tonnes
> landings(bt4, "TBB", "sol")
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957  1958  1959  1960  1961  1962  1963  1964  1965  1966  1967  1968
  all 12067 14287 13832 18620 23566 26877 26164 11342 17043 33340 33439 33179
     year
age   1969  1970  1971  1972  1973  1974  1975  1976  1977  1978  1979  1980
  all 27559 19685 23652 21086 19309 17989 20773 17326 18003 20280 22598 15807
     year
age   1981  1982  1983  1984  1985  1986  1987  1988  1989  1990  1991  1992
  all 15403 21579 24927 26839 24248 18201 17368 21590 21805 35120 33513 29341
     year
age   1993  1994  1995  1996  1997  1998  1999  2000  2001
  all 31491 33002 30467 22651 14901 20868 23475 22641 19944

units:  NA
```

Extracting metiers and catches from a `FLFleet` can be tedious, but fortunately, a number of shortcuts are available using square and double square brackets:

```
> class(bt4["TBB"])
[1] "FLFleet"
attr(,"package")
[1] "FLCore"
> class(bt4[["TBB"]])
[1] "FLMetier"
attr(,"package")
[1] "FLCore"
```

and also for the catches. The example below returns a FLFleet with only the ”ple” catch for ”TBB” metier:

```
> class(bt4["TBB", "ple"])
[1] "FLFleet"
attr(,"package")
[1] "FLCore"
```

now, returns a FLFleet with only the “ple” catch for all metiers:

```
> class(bt4[, "ple"])
[1] "FLFleet"
attr(,"package")
[1] "FLCore"
```

and a new one that returns the 'ple' FLCatch from metier 'TBB':

```
> class(bt4[["TBB"]][["ple"]])
[1] "FLCatch"
attr(,"package")
[1] "FLCore"
```

Or a combination of all elements, which in this example returns a FLFleet with only the 'ple' catch for 'TBB' metier

```
> class(bt4[, c("ple", "sol")])
[1] "FLFleet"
attr(,"package")
[1] "FLCore"
```

2.5 Exporting in/Importing from data frame

Complex objects can also be transformed into a 2D representation, using `as.data.frame()` to coerce the object into a `data.frame`:

```
> head(as.data.frame(bt4), 10)
     slot age year   unit season   area iter   data metier catch
1  effort all 1957 unique    all unique    1 0.2922     NA    NA
2  effort all 1958 unique    all unique    1 0.3608     NA    NA
3  effort all 1959 unique    all unique    1 0.3657     NA    NA
4  effort all 1960 unique    all unique    1 0.3440     NA    NA
5  effort all 1961 unique    all unique    1 0.3967     NA    NA
6  effort all 1962 unique    all unique    1 0.3669     NA    NA
7  effort all 1963 unique    all unique    1 0.5092     NA    NA
8  effort all 1964 unique    all unique    1 0.5159     NA    NA
9  effort all 1965 unique    all unique    1 0.6000     NA    NA
10 effort all 1966 unique    all unique    1 0.4904     NA    NA
```

`as.data.frame()` for a `FLFleet` object will return the value of all slots of that fleet, which is maybe

more than what you need, especially if you are dealing with several fleets.

We can define our own extraction file in order to coerce only the slot we are interested in into a data frame, using a number of sequential `lapply()`:

```
> get.slot.fleet <- function(fleets, slot.) {
+     slt. <- eval(parse("", text = slot.))
+     res <- lapply(fleets, function(x) {
+         mt. <- lapply(x@metiers, function(x1) {
+             res. <- as.data.frame(slt.(x1))
+             names(res.)[which(names(res.) == "data")] <- slot.
+             res.$fleet <- x@name
+             res.$metier <- x1@name
+             return(res.)
+         })
+         mt. <- do.call(rbind, mt.)
+     })
+     res <- do.call(rbind, res)
+     return(res)
+ }
> a.df <- get.slot.fleet(fleets4, "landings")
> head(a.df, 5)
                  age year   unit season   area iter landings qname      fleet
my fleet 1.TBB.1 all 1957 unique    all unique    1    63542   ple my fleet 1
my fleet 1.TBB.2 all 1958 unique    all unique    1    68983   ple my fleet 1
my fleet 1.TBB.3 all 1959 unique    all unique    1    77615   ple my fleet 1
my fleet 1.TBB.4 all 1960 unique    all unique    1    86962   ple my fleet 1
my fleet 1.TBB.5 all 1961 unique    all unique    1    84667   ple my fleet 1
                 metier
my fleet 1.TBB.1    TBB
my fleet 1.TBB.2    TBB
my fleet 1.TBB.3    TBB
my fleet 1.TBB.4    TBB
my fleet 1.TBB.5    TBB
```

(Note that for convenience we only show the 20 first lines of the output data.frame here).

We may want to do the reverse operation i.e. create a `FLFleet` from some data.frames (at least one) by filling the `FLQuant` slots. Remenber that to create a `FLQuant` we need the data.frame with specific column names.

For example, starting by reusing the get.slot.fleet function we have just created to get a data frame, we first change the columns name to have the right ones and then use `as.FLQuant`:

```
> a.df1 <- get.slot.fleet(fleets4, "landings.n")
> a.df1 <- a.df1[a.df1$fleet == "my fleet 1", ]
> colnames(a.df1)[colnames(a.df1) %in% "landings.n"] <- "data"
> landings.n <- as.FLQuant(a.df1[, 1:7])
> a.df2 <- effort(fleets4[[1]])
> colnames(a.df2)[colnames(a.df2) %in% "effort"] <- "data"
> effort <- as.FLQuant(a.df2[, 1:7])
> a.fleet <- FLFleet(effort = effort, FLMetier(name = "my.metier",
+     catches = FLCatch(name = "my.catch", landings.n = landings.n)))
> summary(a.fleet)
An object of class "FLFleet"

Name:
Description:
Range: min max minyear maxyear
1 15 1957 2001
Quant: age

effort        : [ 1 7 1 1 1 1 ], units =  NA
fcost         : [ 1 7 1 1 1 1 ], units =  NA
capacity      : [ 1 7 1 1 1 1 ], units =  NA
crewshare     : [ 1 7 1 1 1 1 ], units =  NA

Metiers:
met :
my.catch : [ 15 45 1 1 1 1 ]
```

This illustrates that we can create the `FLFleet` objects using data previously stored into data.frames as long as the colums names of the data.frame are "age" "year" "unit" "season" "area" "iter" "data".

2.6 Methods with FLFleet in FLCore

Few standard FLR methods are defined for `FLFleet`, `FLMetier` and `FLCatch` objects. We illustrate them below for `FLFleet` but the same apply for the two other classes.

`revenue` calculates the revenue from elements of an `FLFleet` object as the sum of the landings weight-at-quant times its price by quant, or landings*price if quant-based information is not available.

`window` extracts the subset of the `FLFleet` object observed between the times `start` and `end`.

```
> summary(window(bt4, end = 1999))
An object of class "FLFleet"

Name: beam trawl fleet
Description: Example of an FLFleet
Range: min max pgroup minyear maxyear
0 0 NA 1957 1999
Quant: age

effort        : [ 1 43 1 1 1 1 ], units =  NA
fcost         : [ 1 43 1 1 1 1 ], units =  NA
capacity      : [ 1 43 1 1 1 1 ], units =  NA
crewshare     : [ 1 43 1 1 1 1 ], units =  NA

Metiers:
TBB :
ple : [ 15 43 1 1 1 1 ]
sol : [ 10 43 1 1 1 1 ]
```

`propagate` extends the `FLFleet` object alongside the 6th dimension, `iter`, by extending all the slots at all levels. Default for `fill.iter` is `TRUE`, i.e. all iterations will be filled with the initial values. The opposite action is to extract the values within one single iteration, by using the `iter` method.

```
> bt_iter <- propagate(bt4, 10, fill.iter = FALSE)
> summary(bt_iter)
An object of class "FLFleet"

Name: beam trawl fleet
Description: Example of an FLFleet
Range: min max pgroup minyear maxyear
0 0 NA 1957 2001
Quant: age

effort        : [ 1 45 1 1 1 10 ], units =  NA
fcost         : [ 1 45 1 1 1 10 ], units =  NA
capacity      : [ 1 45 1 1 1 10 ], units =  NA
crewshare     : [ 1 45 1 1 1 10 ], units =  NA

Metiers:
TBB :
ple : [ 15 45 1 1 1 10 ]
sol : [ 10 45 1 1 1 10 ]
> iter(effort(bt_iter), 5)
An object of class "FLQuant"
, , unit = unique, season = all, area = unique

     year
age   1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970
  all NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
     year
age   1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
  all NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
     year
age   1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
  all NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
     year
age   1999 2000 2001
  all NA   NA   NA

units:  NA
```

2.7 Coding your own methods and functions

The methods and functions explained below are not included in, but we use them repeatedly and can provide a useful basis for `FLFleet`'s users. For example, calculating the partial F of a `FLFleet` for a given stock `st`, knowing the effort, the catchability and the selectivity by metier, but assuming that not all metiers do catch the stock `st`, can be a bit tricky. There you could use something like that (defined here when `catch.q` is not defined by quant but on average, whereas selectivity is by quant (age/length):

```
> partialF <- function(fleet = fl, stock = st) {
+     if (stock %in% unique(unlist(lapply(fleet@metiers, function(x)
names(x@catches))))) {
+         e. <- effort(fleet)
+         mt. <- lapply(fleet@metiers, function(x) {
+             if (stock %in% names(x@catches)) {
+                 eff <- e. * effshare(x)
+                 Q <- catch.q(x)[[stock]]
+                 sel <- landings.sel(x)[[stock]]
+                 harv <- sweep(sweep(sel, 2:6, eff, FUN = "*"),
+                   2:6, Q, FUN = "*")
+             }
+             else harv <- 0
+             harv[is.na(harv)] <- 0
+             return(harv)
+         })
+     }
+     else mt. <- 0
+     return(fleet)
+ }
```

Which can then be summed across metiers.

Note that the use of the function `sweep` is crucial to compute some operations between slots when the dimensions of these slots may differ e.g. between `effort` (a scalar) and `selectivity` (age-structured).

3 Final thoughts

In spite of it complex structure, `FLFleet` has proven to be useful and handy enough to address a number of mixed-fisheries issues, see some examples of references below. A number of improvements are still ongoing, of which one of the most useful will be the improved linkage of `FLFleet` with the forward projection functions in the FLash package, in order to address fleet-specific constraints in Management Strategies Evaluations.

Examples of references using `FLFleet:`

Bastardie, F., Vinther, M., Nielsen, J.R., Ulrich, C., and Storr-Paulsen, M., 2010. Stock-based vs. Fleet-based evaluation of the multiannual management plan for the cod stocks in the Baltic Sea. Fish. Res. 101, 188-202

Bastardie, F., Nielsen, J. R., and Kraus, G. 2010. The eastern Baltic cod fishery: a fleet-based management strategy evaluation framework to assess the cod recovery plan of 2008. – ICES Journal of Marine Science, 67: 71–86.

Baudron, A., Ulrich, C., Nielsen, J. R., and Boje, J. 2010. Comparative evaluation of a mixed-fisheries effort-management system based on the Faroe Islands example. – ICES Journal of Marine Science, 67: 1036–1050

ICES, 2009. Report of the Workshop on Mixed Fisheries Advice for the North Sea (WKMIXFISH), 26-28 August 2009, ICES HQ, Copenhagen, Denmark. ICES CM 2009/ACOM:47 http://www.ices.dk/workinggroups/ViewWorkingGroup.aspx?ID=360

Ulrich, C., Reeves, S.A., Vermard, Y., Holmes, S., and Vanhee W., Subm. Reconciling single-species TACs in the North Sea demersal fisheries using the Fcube mixed-fisheries advice framework. Can. J. of Fish. Aquat. Sci., subm.