S4 classes - Gene sequence class using slots

FUll example of using S4 to create a gene sequence class.

Define the super class and sub classes

Source: https://github.com/cran/sequences/blob/master/R/DataClasses.R

```
setClass("GenericSeq",
         slots = c(id = "character",
                   alphabet = "character",
                   sequence =  "character"),
         contains = "VIRTUAL",
         validity = function(object) {
           isValid <- TRUE                        # check if valid
           if (length(object@sequence) > 0) {
             chars <- casefold(unique(unlist(strsplit(object@sequence, ""))))
             isValid <- all(chars %in% casefold(object@alphabet))    # casefold same as tolower()
           }
           if (!isValid)
             cat("Some characters are not defined in the alphabet.\n")
           return(isValid)
         })


setClass("DnaSeq",
         contains = "GenericSeq",                      # subclass of GenericSeq
         prototype = list(
           id = paste("my DNA sequence", date()),
           alphabet = c("A","C","G","T"),
           sequence = character())
         )

setClass("RnaSeq",
         contains = "GenericSeq",                       # subclass of GenericSeq
         prototype = list(
           id = paste("my RNA sequence",date()),
           alphabet = c("A","C","G","U"),
           sequence = character())
         )

# constructors for DNA and RNA
DnaSeq <- function(id, sequence)
    new("DnaSeq", id = id, sequence = sequence)

RnaSeq <- function(id, sequence)
    new("RnaSeq", id = id, sequence = sequence)
```

Define the generics

Source: https://github.com/cran/sequences/blob/master/R/AllGenerics.R

```
setGeneric("id", function(object, ...) standardGeneric("id"))
```

```
## [1] "id"
```

```
setGeneric("id<-", function(object,value) standardGeneric("id<-"))
```

```
## [1] "id<-"
```

```
setGeneric("alphabet", function(object, ...) standardGeneric("alphabet"))
```

```
## [1] "alphabet"
```

```
## There is already a 'seq' method (see ?seq),
## although not a generic one (see isGeneric(seq))
setGeneric("seq", function(...) standardGeneric("seq"))
```

```
## [1] "seq"
```

```
setGeneric("seq<-", function(object,value) standardGeneric("seq<-"))
```

```
## [1] "seq<-"
```

```
## Same note that above for print
setGeneric("print", function(x, ...) standardGeneric("print"))
```

```
## [1] "print"
```

```
setGeneric("rev",function(x) standardGeneric("rev"))
```

```
## [1] "rev"
```

```
setGeneric("comp",function(object, ...) standardGeneric("comp"))
```

```
## [1] "comp"
```

```
setGeneric("transcribe", function(object, ...) standardGeneric("transcribe"))
```

```
## [1] "transcribe"
```

Generic sequence methods

Source: https://github.com/cran/sequences/blob/master/R/methods-GenSeq.R

```
setMethod("show", "GenericSeq",
          function(object) {
            cat("Object of class",class(object),"\n")
            cat(" Id:",id(object),"\n")
            cat(" Length:",length(object),"\n")
            cat(" Alphabet:",alphabet(object),"\n")
            cat(" Sequence:",seq(object), "\n")
          })


setMethod("print", "GenericSeq",
          function(x) {
            sq <- strsplit(seq(x), "")[[1]]
            cat(">",id(x),"\n")
            cat(" 1   ")
            for (i in 1:length(x)) {
```

```
              if ((i %% 10)==0) {
                cat("\n")
                cat(i,"  ")
              }
              cat(sq[i])
            }
            cat("\n")
          })


setMethod("id", "GenericSeq", function(object, ...) object@id)

setMethod("id<-", "GenericSeq",
          function(object,value) object@id <- value)

setReplaceMethod("id",
                 signature(object="GenericSeq",
                           value="character"),
                 function(object, value) {
                   object@id <- value
                   if (validObject(object))
                     return(object)
                 })


setMethod("alphabet", "GenericSeq", function(object, ...) object@alphabet)
setMethod("length", "GenericSeq", function(x) nchar(x@sequence))
setMethod("seq", "GenericSeq", function(object, ...) object@sequence)

setReplaceMethod("seq",
                 signature(object="GenericSeq",
                           value="character"),
                 function(object, value) {
                   object@sequence <- value
                   if (validObject(object))
                     return(object)
                 })

setMethod("rev","GenericSeq",
          function(x) paste(rev(strsplit(seq(x),"")[[1]]),collapse=""))


## this is only an example of initialize function,
## note Generic is in fact virtual
setMethod("initialize", "GenericSeq",
          function(.Object, ..., id = "", sequence = ""){
            .Object@id <- id
            .Object@sequence <- toupper(sequence)
            callNextMethod(.Object, ...)
          })


setMethod("[", "GenericSeq",
```

```
          function(x, i, j="missing", drop="missing") {
            if (any(i > length(x)))
              stop("subscript out of bounds")
            s <- seq(x)
            s <- paste(strsplit(s,"")[[1]][i], collapse="")
            x@sequence <- s
            if (validObject(x))
              return(x)
          })
```

Specific methods for DNA and RNA

Source: https://github.com/cran/sequences/blob/master/R/methods-DnaSeq.R

```
setMethod("comp", "DnaSeq",
          function(object, ...) {
            chartr("ACGT","TGCA", seq(object))
          })


setMethod("transcribe", "DnaSeq",
          function(object, ...) {
            .sequence <- chartr("T","U", toupper(seq(object)))
            .id <-  paste(id(object), "-- transcribed")
            rna <- new("RnaSeq",
                       id = .id,
                       alphabet = c("A","C","G","U"),
                       sequence = .sequence)
            return(rna)
          })

setMethod("comp", "RnaSeq",
          function(object, ...) {
            chartr("ACGU","UGCA",seq(object))
          })
```

Function to read sequences

```
readFasta <- function(infile){
  lines <- readLines(infile)
  header <- grep("^>", lines)
  cat("Sequences found at lines: ", header, "\n")
  if (length(header) > 1) {
    warning("Reading first sequence only.\n")
    lines <- lines[header[1]:(header[2]-1)]
    header <- header[1]
  }
  .id <- sub("^> *","", lines[header], perl=TRUE)
  .sequence <- toupper(paste(lines[(header+1):length(lines)], collapse=""))
  .alphabet <- toupper(unique(strsplit(.sequence,"")[[1]]))
  if (all(.alphabet %in% c("A","C","G","T"))) {
```

```
      newseq <- DnaSeq(.id, .sequence)
  } else if (all(.alphabet %in% c("A","C","G","U"))) {
      newSeq <- RnaSeq(.id, .sequence)
  } else {
      stop("Alphabet ", .alphabet," is unknown.")
  }
  if (validObject(newseq))
    return(newseq)
}
```

```
fastafilename <- dir(path="./inst/extdata",
                     full.name=TRUE,
                     pattern="fasta$")
fastafilename
```

```
## [1] "./inst/extdata/aDnaSeq.fasta"     "./inst/extdata/moreDnaSeqs.fasta"
```

Read 1st sequence

```
myseq <- readFasta(fastafilename[1])
```

```
## Sequences found at lines:  1
```

```
myseq
```

```
## Object of class DnaSeq
##  Id: example dna sequence
##  Length: 132
##  Alphabet: A C G T
##  Sequence: AGCATACGACGACTACGACACTACGACATCAGACACTACAGACTACTACGACTACAGACATCAGACACTACATATTTACATCATCAGAG
```

```
transcribe(myseq)
```

```
## Object of class RnaSeq
##  Id: example dna sequence -- transcribed
##  Length: 132
##  Alphabet: A C G U
##  Sequence: AGCAUACGACGACUACGACACUACGACAUCAGACACUACAGACUACUACGACUACAGACAUCAGACACUACAUAUUUACAUCAUCAGAG
```

```
alphabet(myseq)
```

```
## [1] "A" "C" "G" "T"
```

```
seq(myseq)
```

```
## [1] "AGCATACGACGACTACGACACTACGACATCAGACACTACAGACTACTACGACTACAGACATCAGACACTACATATTTACATCATCAGAGATTATA
```

```
print(myseq)
```

```
## > example dna sequence
##   1   AGCATACGA
##  10   CGACTACGAC
##  20   ACTACGACAT
##  30   CAGACACTAC
##  40   AGACTACTAC
##  50   GACTACAGAC
##  60   ATCAGACACT
```

```
## 70   ACATATTTAC
## 80   ATCATCAGAG
## 90   ATTATATTAA
## 100   CATCAGACAT
## 110   CGACACATCA
## 120   TCATCAGCAT
## 130   CAT
```

```
rev(myseq)
```

```
## [1] "TACTACGACTACTACTACACAGCTACAGACTACAATTATATTAGAGACTACTACATTTATACATCACAGACTACAGACATCAGCATCATCAGACA
```

```
comp(myseq)
```

```
## [1] "TCGTATGCTGCTGATGCTGTGATGCTGTAGTCTGTGATGTCTGATGATGCTGATGTCTGTAGTCTGTGATGTATAAATGTAGTAGTCTCTAATAT
```

```
length(myseq)
```

```
## [1] 132
```

```
myseq[5:10]
```

```
## Object of class DnaSeq
##  Id: example dna sequence
##  Length: 6
##  Alphabet: A C G T
##  Sequence: TACGAC
```

```
myseq[5, 10]
```

```
## Object of class DnaSeq
##  Id: example dna sequence
##  Length: 1
##  Alphabet: A C G T
##  Sequence: T
```

```
# T
```

```
myseq[1, 10]
```

```
## Object of class DnaSeq
##  Id: example dna sequence
##  Length: 1
##  Alphabet: A C G T
##  Sequence: A
```

```
# A
```

Read the 2nd seqence

```
myseq2 <- readFasta(fastafilename[2])
```

```
## Sequences found at lines:  1 4 9
```

```
## Warning in readFasta(fastafilename[2]): Reading first sequence only.
```

```
myseq2
```

```
## Object of class DnaSeq
```

```
##  Id: sequence 1
##  Length: 79
##  Alphabet: A C G T
##  Sequence: AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCAAGGTGAGATACGACGTAGATGCTAGCTGACTCGATGC
```

```
# using setReplaceMethod
id(myseq2) <- "Sequence #1"
myseq2
```

```
## Object of class DnaSeq
##  Id: Sequence #1
##  Length: 79
##  Alphabet: A C G T
##  Sequence: AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCAAGGTGAGATACGACGTAGATGCTAGCTGACTCGATGC
```

```
# using setReplaceMethod
`id<-`(myseq2, "this is sequence no. 1")
```

```
## Object of class DnaSeq
##  Id: this is sequence no. 1
##  Length: 79
##  Alphabet: A C G T
##  Sequence: AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCAAGGTGAGATACGACGTAGATGCTAGCTGACTCGATGC
```

Let's start by loading the package and read a fasta sequence

```
read.csv("./inst/extdata/aDnaSeq.fasta", header = FALSE)
```

```
##                                                 V1
## 1                            > example dna sequence
## 2 agcatacgacgactacgacactacgacatcagacactacagactactac
## 3 gactacagacatcagacactacatatttacatcatcagagattatatta
## 4                acatcagacatcgacacatcatcatcagcatcat
```

```
read.csv("./inst/extdata/moreDnaSeqs.fasta", header = FALSE)
```

```
##                                                    V1
## 1                                        > sequence 1
## 2  AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCAAGGTGAGATA
## 3                           CGACGTAGATGCTAGCTGACTCGATGC
## 4                                        > sequence 2
## 5  CGATCGATCGTACGTCGACTGATCGTAGCTACGTCGTACGTAGGGAGTAGGG
## 6  AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCAAGGTGAGATA
## 7  GCGTACGAGAGACTGACATGGTAAGGTTTAAAGGGGTTACCCAAAGTGAGAT
## 8                    CATCGTCAGTTACTGCATGCTCGGAGACCAAGG
## 9                                        > sequence 3
## 10 CGATCGATCGTACGTCGACTGATCGTAGCTACGTCGTACGTAGGGAGTAGGG
## 11 GCGTACGAGAGACTGACATGGTAAGGTTTAAAGGGGTTACCCAAAGTGAGAT
## 12                             CATCGTCAGTTACTGCATGCTCG
```