Devoir 3 (solution): Programmation orientée objet et polynômes

Exercice 1

a)

```
plynR <- function(coefs) {
    is.numeric(coefs) || stop("`coefs` doit être un vecteur de nombres réels")

    structure(coefs, class = c("plynR", "numeric"))
}
```

b)

Avant de passer à la méthode demandée, nous allons implémenter quelques méthodes utilitaires. La méthode `deg` donne le degré d'un `plynR`. On défère l'appel à `deg.plynR` (fonction `NextMethod`) à la méthode `deg.default`.

```
deg <- function(p) UseMethod("deg")
deg.default <- function(p) {
    p_numeric <- as.numeric(p)

    for (k in rev(seq_along(p_numeric)))
        isTRUE(all.equal(p_numeric[k], 0L)) || return(k - 1L)

    0L
}
deg.plynR <- function(p) NextMethod()
```

La méthode `[.plynR` est appellée lorsqu'on utilise la syntaxe `p[i]` pour `p` $\in$ `plynR`. On l'implémente de manière à ce que `p[i]` retourne a_i, c'est-à-dire le coefficient de x^i. Notez que si a_i n'a pas été spécifié par l'utilisateur, la méthode retourne 0.

```
`[.plynR` <- function(x, i) {
    xN <- as.numeric(x)

    sapply(i, function(k) {
        k < 0L && stop("puissance invalide")
        k > deg(xN) && return(0.0)
        xN[k + 1L]
    })
}
```

On implémente à présent la méthode `print`.

```
char_coef <- function(x, i, first) UseMethod("char_coef")
char_coef.plynR <- function(x, i, first = FALSE) {
```

```
    signe <- ifelse(first,
             ifelse(x[i] < 0L, "-", ""),
             ifelse(x[i] < 0L, "- ", "+ "))

    va <- ifelse(isTRUE(all.equal(i, 0L)), "",
                 paste("x",
                       ifelse(isTRUE(all.equal(i, 1L)),
                              "",
                              paste("^", i, sep = "")),
                       sep = ""))

    cof <- ifelse(isTRUE(all.equal(abs(x[i]), 1L)) & !identical(va, ""),
                  "",
                  abs(x[i]))

    paste(signe, cof, va, sep = "")
}

print.plynR <- function(x) {
    inds <- Filter(function(i) !isTRUE(all.equal(x[i], 0L)),
                   seq_along(x) - 1L)
    if (identical(length(inds), 0L)) {
        cat("0\n")
        return()
    }

    cat(paste(char_coef(x, inds[[1L]], first = TRUE),
              do.call(paste, lapply(inds[-1L], char_coef, x = x))),
        "\n")
}
```

c)

La solution présentée ici est légèrement plus sophistiquée que ce qui était demandé. On commence par définir la méthode `+` pour la classe `plynR`. Par la suite, on définit la méthode `plus` (ainsi que sa générique) sur la base de l'opérateur `+`. `plus.plynR` permet de faire la somme d'un nombre arbitraire de polynômes. Finalement, on définit `moins` sur la base de `+` également. Remarquez que comme `plynR` hérite de `numeric`, si `p` $\in$ `plynR`, `-p` et `length(p)` se comportent comme attendu.

```
`+.plynR` <- function(e1, e2) {
    d_max <- max(deg(e1), deg(e2))
    plynR(e1[0L:d_max] + e2[0L:d_max])
}
```

```r
plus <- function(...) UseMethod("plus")
plus.plynR <- function(...) Reduce(`+`, list(...))

moins <- function(p, q) UseMethod("moins")
moins.plynR <- function(p, q) p + -q
```

d)

Même idée que pour `plus` ici.

```r
`*.plynR` <- function(e1, e2) {
    plynR(sapply(seq(0L, deg(e1) + deg(e2)), function(k) {
        inds <- 0L:k
        sum(e1[inds] * e2[rev(inds)])
    }))
}

fois <- function(...) UseMethod("fois")
fois.plynR <- function(...) Reduce(`*`, list(...))
```

e)

```r
derive <- function(p) UseMethod("derive")
derive.plynR <- function(p) {
    identical(deg(p), 0L) && return(plynR(0))

    plynR(sapply((seq_along(p) - 1L)[-1L], function(k) k * p[k]))
}
```

f)

```r
racines <- function(p) UseMethod("racines")
racines.plynR <- function(p) polyroot(p)
```

g)

```r
format_cplx <- Vectorize(function(z) {
    isTRUE(all.equal(z, 0.0 + 0.0i)) && return("")

    s_r <- ifelse(Re(z) > 0.0, "- ", "+ ")
    s_i <- ifelse(Im(z) > 0.0, "- ", "+ ")
    r <- abs(Re(z))
    i <- abs(Im(z))

    isTRUE(all.equal(i, 0.0)) && return(paste(s_r, r, sep = ""))
    isTRUE(all.equal(r, 0.0)) && return(paste(s_i, i, "i", sep = ""))
```

```
    paste(s_r, r, " ", s_i, i, "i", sep = "")
})

format_term <- function(r) {
    identical(r, "") && return("x")
    paste("(x ", r, ")", sep = "")
}

summary.plynR <- function(object) {
    print(object)
    identical(deg(object), 0L) && return()

    cat("=", object[deg(object)])

    roots <- format_cplx(round(racines(object), 2L))
    cat(paste(sapply(roots, format_term)), "\n", sep = "")
}
```

Exemples

```
p <- plynR(c(2, 3))
q <- plynR(c(0, 1, 4, 0, 0))
print(p)
print(q)
```

```
2 + 3x
x + 4x^2
```

```
print(p * q)
```

```
2x + 11x^2 + 12x^3
```

```
print(p + q)
```

```
2 + 4x + 4x^2
```

```
print(fois(p, p, q, q, q))
```

```
4x^3 + 60x^4 + 345x^5 + 940x^6 + 1200x^7 + 576x^8
```

```
print(derive(fois(p, p, q, q, q)))
```

```
12x^2 + 240x^3 + 1725x^4 + 5640x^5 + 8400x^6 + 4608x^7
```

```
racines(q)
```

0+0i

-0.25+0i

```
summary(q)
```

```
x + 4x^2
= 4x(x + 0.25)
```