Assignment 2

Pavlo Nikitenko(3049843), Chiara Sauren(3050345), Viktor Stoppel(3004345)

2022-07-06

Make sure to install and load following packages first:

```
library(profvis)
library(purrr)
library(tibble)
```

1)

a) A function to calculate the loglikelihood negative binomial distribution.

```
lprob_nbinomial<-function(p, x){
  (-sum(log(choose(x+10-1, x))+10*log(p)+x*log(1-p)))
}
```

Generate some example data to test and profile the function:

```
x <- rnbinom( n = 1e5, size = 5, prob = .2)
x <- rnbinom( n = 1e6, size = 5, prob = .2)
length(x)
```

```
## [1] 1000000
```

Profile collecting of example data and parts of the function in comparison

```
profvis({

  test <-  (-sum((log(choose(x+10-1, x)))+ 10 * log(0.5)+x * log(1-0.5)))

})
```

With increasing length of the example data x, the computation time increases accordingly. With n = 100000 (1e5) observations the lprob_nbinomial() function takes up to 10 ms, using up 2.3 mbit of memory space. With n = 1000000 (1e6) the computation time already takes 110 ms and takes up to 22.9 mbits of memory space.

b)

Ephemeral environments are meant to serve as a testing, temporary or dynamic environment that show a replica of the actual work-environment. They usually use replicated data and test whether a test or step is applicable before risking on damaging the real work environment. They help in accelerating the programming or software development cycle, limiting the need to rework broad parts of the written code. In comparison to a ephemeral environment, the function factory encloses the environment of the manufactured function. This serves as the execution environment of the function factory.

c) Function factory:

```
ll_nbinomial<-function(x){
  function(p){
    (-sum(log(choose(x+10-1, x))+10*log(p)+x*log(1-p)))
  }
}
```

d) Test of the function factory with pseudo-random numbers x1

```
set.seed(123)
x1<-rnbinom(n=1e3, size = 10, prob = .3)

# Test the function
ll<-ll_nbinomial(x1)
ll(0.2)
```

```
## [1] 4553.332
```

e) function that obtains the MLE:

```
nbin_mle<-function(x){
mle <-  list(optim(par = 0,fn=ll ,lower = 1e-8,
        upper = 1-1e-8))
return(mle)
}
#test function
nbin_mle(x1)
```

```
## [[1]]
## [[1]]$par
## [1] 0.3036198
##
## [[1]]$value
## [1] 3560.332
##
## [[1]]$counts
## function gradient
##       13       13
##
## [[1]]$convergence
```

```
## [1] 0
##
## [[1]]$message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
#alternative way

mle_a<-optimize(ll, interval = c(0,1))
mle_a
```

```
## $minimum
## [1] 0.3036122
##
## $objective
## [1] 3560.332
```

f) Implement a summary method for the output of nbin_mle

```
nbin_mle<-function(x){
  mle <-  list(optim(par = 0,fn=ll ,lower = 1e-8,
                     upper = 1-1e-8, method = "L-BFGS-B"))
  class(mle)<-"my_mle"
  return(mle)
}


nbin_mle(x1)
```

```
## [[1]]
## [[1]]$par
## [1] 0.3036198
##
## [[1]]$value
## [1] 3560.332
##
## [[1]]$counts
## function gradient
##       13       13
##
## [[1]]$convergence
## [1] 0
##
## [[1]]$message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
##
## attr(,"class")
## [1] "my_mle"
```

```
nbin_mle(x1)[[1]]$par
```

```
## [1] 0.3036198
```

```
summary.my_mle <- function(x) {
  if(class(x) == "my_mle") {
    x <- unlist(x)
    out <<- list(
      name = quote(x),
      min = min(x),
      tibble(
        prob=seq(0,1,0.01)),
      ll=my_nbin_mle[[1]]$par,
      MLE=my_nbin_mle[[1]]$value,
      counts=my_nbin_mle[[1]]$counts
    )
    class(out) <- "summary.my_mle"
    return(out)
  } else {
    message("Object not of class my_mle!")
  }
}

my_nbin_mle<-nbin_mle(x1)
summary(my_nbin_mle)
```

```
## $name
## x
##
## $min
## [1] "0"
##
## [[3]]
## # A tibble: 101 x 1
##     prob
##    <dbl>
##  1  0
##  2  0.01
##  3  0.02
##  4  0.03
##  5  0.04
##  6  0.05
##  7  0.06
##  8  0.07
##  9  0.08
## 10  0.09
## # ... with 91 more rows
##
## $ll
## [1] 0.3036198
```

```
##
## $MLE
## [1] 3560.332
##
## $counts
## function gradient
##       13       13
##
## attr(,"class")
## [1] "summary.my_mle"
```

g) print and plot method for objects of class summary.my_mle:

```
# print method
print.summary.my_mle <- function(x) {
  if(class(x) == "summary.my_mle") {
    cat(
      "Summary for", x$name, ":\n\n",
      "Probabilities for MLE", my_nbin_mle[[1]]$value, "for", my_nbin_mle[[1]]$par
    )
  } else {
    message("Object not of class summary.my_mle!")
  }

}

summary(my_nbin_mle)
```

```
## Summary for x :
##
##  Probabilities for MLE 3560.332 for 0.3036198
```

```
# plot method
plot.summary.my_mle <- function(x){
 pi <- (1:99)/100
loglikelihood <- sapply(pi, ll)
plot(pi, loglikelihood, type = "l",
     xlab = "Pi", ylab = "-log L(pi; x, k =10")
points(y =my_nbin_mle[[1]]$value,x=my_nbin_mle[[1]]$par,
      col="red", type = "p", lwd=2)


}
```

h) Demonstration with example data from d)

```
summary(my_nbin_mle)
```

```
## Summary for x :
##
##  Probabilities for MLE 3560.332 for 0.3036198
```

```
plot.summary.my_mle(x1)
```