The results below are generated from an R script.

```
#### capture des rentes ####
renteExtract <- function(text){
  regex <- "\\-"
  tirets <- grep(regex, text, value=FALSE)
  text <-  text[-tirets]
  regex <- "[0-9]{2,}\\."
  numRente <-  str_subset(text,regex)
  indexRente <- grep(regex,text,value=FALSE)
  result <- NULL
  if(!is.null(indexRente)){
    for (j in indexRente){
      sentence <- ""
      beg <- j+1
      end <- (indexRente[(which(indexRente==j))+1])-1
      if(is.na(end)) {
        end <- length(text)
      }
      for (i in text[beg:end]) {
        sentence <- str_c(sentence,i," ")
      }
      result <- c(result,sentence)
    }
    df = data.frame(numRente,result)
  }else { #cas ou il n y a pas de rente dans la connetablie
    numRente <- NA
    result <- NA
  }

#separation entre les num de rentes successives
  if(nrow(df)>0){
    for(i in 1:nrow(df)){
      df$numRente[i] <- str_replace(df$numRente[i],
                                    str_c(as.character(count_connetablie),"\\."),
                                    str_c("\\.",as.character(count_connetablie)))
      count_connetablie <<- count_connetablie + 1
    }
  }
  #df cumulant toutes les rentes. A affecter a un dataframe global
  df_rentes <<- rbind(df_rentes, df)
  return(df)
}

#### capture du rang des voies ####
rdvExtract <- function (text,rang){
  regex <- "^[AB]$"
  indexRdV <- grep(regex,text,value=FALSE)

  #suppression des faux indexes
  v_remove_A <- NULL
  for(i in indexRdV){
    if(text[i] ==  "A" && !str_detect(text[i+1],"[0-9]{2,}\\.")){
      v_remove_A <- c(v_remove_A,which(indexRdV==i))
```

```
    }
  }
  if(!is.null(v_remove_A)){
    indexRdV <- indexRdV[-v_remove_A]
  }

  RdV <- text[indexRdV]
  result <- NULL
  for (j in indexRdV){
    section <- NULL
    beg <- j+1
    end <- (indexRdV[(which(indexRdV==j))+1])-1
    if(is.na(end)) {
      end <- length(text)
    }
    for (i in text[beg:end]) {
      section <- str_c(section,i," ")
    }
    result <- c(result,section)
  }

  df = data.frame()
  #dataframe contenant la section pour chaque Rang de voie
  df_rdv = data.frame(RdV, result)
  for (i in 1:nrow(df_rdv)) {
    t  <- renteExtract(unlist(str_split(df_rdv$result[i], " ")))
    for (j in 1:nrow(t)) {
      df <- rbind(df, c(df_rdv$RdV[i],t$numRente[j], t$result[j]))
    }
  }
  return(df)
}

#### capture des connetablies ####
connetablieExtract <- function(text){
  regex <- "[0-9]+°"
  indexConnetablie <- grep(regex,text,value=FALSE)

  #fusion des elements "bis" du vecteur au numero de connetablie
  v_remove <- NULL
  for(i in indexConnetablie){
    if(text[i+1]=="bis"){
      text[i] <- str_c(text[i],"bis", sep=" ")
      v_remove <- c(v_remove,i+1)
    }
  }
  if(!is.null(v_remove)){
    text <- text[-v_remove]
  }
  indexConnetablie <- grep(regex,text,value=FALSE)

  numConnetablie <- grep(regex,text,value=TRUE)
  regex <- "^[AB]$"
```

```
indexRdV <- grep(regex,text,value=FALSE)
#suppression des faux indexes
v_remove <- NULL
for(i in indexRdV){
  if(text[i] ==  "A" && !str_detect(text[i+1],"[0-9]{2,}\\.")){
    v_remove <- c(v_remove,which(indexRdV==i))
  }
}
if(!is.null(v_remove)){
  indexRdV <- indexRdV[-v_remove]
}
v_connetablie <- NULL
v_section <- NULL

#capture de la definition de chaque connetablie
for (j in indexConnetablie){
  connetablie <- NULL
  RdVMark <- which(indexRdV >= j)[1]
  beg <- j+1
  end <- indexRdV[RdVMark]

  if(is.na(end)) {
    end <- length(text)
  }
  if(!is.na(end)) {
    i <- text[beg]
    while (beg != end && !str_detect(i, "[0-9]{2,}\\." )) {
      connetablie <- str_c(connetablie,i," ")
      beg <- beg +1
      i <- text[beg]
    }
    if(is.null(connetablie)){
      connetablie <- NA
    }
    v_connetablie <- c(v_connetablie,connetablie)
  }
}

# capture de la section de chaque connetablie
for (j in indexConnetablie){
  section <- NULL
  beg <- j+1
  end <- (indexConnetablie[which(indexConnetablie==j)+1])-1

  if(is.na(end)) {
    end <- length(text)
  }
  if(!is.na(end)) {
    for (i in text[beg:end]) {
      section <- str_c(section,i," ")
    }
    v_section <- c(v_section,section)
  }
```

```
  }

  ## donnees en sous forme de Dataframe #
  df_connetablie =  data.frame(numConnetablie, v_connetablie, v_section)
  names(df_connetablie)[1:3] <- c("numConnetablie", "connetablie", "section")
  #df cumulant toutes les connetablie. dataframe global
  df_connetablies <<- rbind(df_connetablies,df_connetablie[1:2])

  ## extraction des rangs de voie  pour chaque connetablie ##
  df = data.frame(numConnetablie <- NULL,
                  connetablie <- NULL,
                  rdv <- NULL,
                  numRente <- NULL
                  ,rente <- NULL)
  for (i in 1:nrow(df_connetablie)) {
    count_connetablie <<- 1

    #cas ou le numero de connetablie comporte une particule bis
    if(str_detect(df_connetablie$numConnetablie[i],"bis")){
      num <- str_extract(df_connetablies$numConnetablie[i],"\\d+°")
      if(nrow(df)>0){
        col <- df[,1]
        count_connetablie <<- length(str_subset(col,num))+1
      }#else print(df_connetablie$numConnetablie[i])
    }

    #cas ou  il n y a pas de rang de voie A
    if(!str_detect(df_connetablie$section[i],"A [0-9]{2,}\\.")){
      t  <- renteExtract(unlist(str_split(df_connetablie$section[i], " ")))
      if ((nrow(t)) == 0) {
        t <-c(NA, NA, NA)
      } else {
        rdvNA <- NA
        rdvNA[1:nrow(t)] <- NA
        t <- cbind(rdvNA,t)
      }

    }else { #cas ou un rang de voie est detecte dans la connetablie
      t  <- rdvExtract(unlist(str_split(df_connetablie$section[i], " ")))
    }
    if(!is.null(nrow(t))){
      for (j in 1:nrow(t)) {
        df <- rbind(df, c(df_connetablie$numConnetablie[i],
                          df_connetablie$connetablie[i],t[j,1], t[j,2],t[j,3]))
      }
    } else {
      df <- rbind(df, c(df_connetablie$numConnetablie[i],
                        df_connetablie$connetablie[i],t[1], t[2],t[3]))
    }
  }
  return(df)
}
```

```
#### capture des escroetes ####
escroeteExtract <- function(text){
  regex <- "^[IV]+([1-9])?( bis| ter)?$"
  indexEscroete <- grep(regex,text,value=FALSE)
  numEscroete <- grep(regex,text,value=TRUE)
  regex <- "[0-9]+°"
  indexConnetablie <- grep(regex,text,value=FALSE)
  v_escroete <- NULL
  v_section <- NULL

  #fusion des elements "bis" du vecteur au numero d'escroete
  v_remove <- NULL
  for(i in indexEscroete){
    if(text[i+1]=="bis"){
      text[i] <- str_c(text[i],"bis", sep=" ")
      v_remove <- c(v_remove,i+1)
    } else if(text[i+1]=="ter"){
      text[i] <- str_c(text[i],"ter", sep=" ")
      v_remove <- c(v_remove,i+1)
    }
  }
  if(!is.null(v_remove)){
    text <- text[-v_remove]
  }
  indexEscroete <- grep("^[IV]+([1-9])?( bis| ter)?$",text,value=FALSE)
  numEscroete <- grep("^[IV]+([1-9])?( bis| ter)?$",text,value=TRUE)


  #capture des definitions des escroetes
  for (j in indexEscroete){
    escroete <- NULL
    connetablieMark <- which(indexConnetablie >= j)[1]
    beg <- j+1
    end <- indexConnetablie[connetablieMark]-1

    if(length(text[beg:end])>3){
      for (i in text[beg:end]) {
        escroete <- str_c(escroete,i," ")
      }
    } else {
      escroete <- NA
    }
    v_escroete <- c(v_escroete,escroete)
  }

  #capture des sections des escroetes
  for (j in indexEscroete){
    section <- NULL
    beg <- j+1
    end <- (indexEscroete[which(indexEscroete==j)+1])-1
    if(is.na(end)) {
      end <- length(text)
    }
```

```
    for (i in text[beg:end]) {
      section <- str_c(section,i," ")
    }
    v_section <- c(v_section,section)
  }
  ## donnees en sous forme de dataframe ##
  df_escroete = data.frame(numEscroete,v_escroete,v_section)
  names(df_escroete)[1:3] <- c("numEscroete", "escroete", "section")
  #df cumulant toutes les connetablie. dataframe global
  df_escroetes <<- rbind(df_escroetes,df_escroete[1:2])

  ## extraction des connetablies pour chaque escroete##
  df = data.frame()
  for (i in 1:nrow(df_escroete)) {
    t  <- connetablieExtract(unlist(str_split(df_escroete$section[i], " ")))
    for (j in 1:nrow(t)) {
      df <- rbind(df, c(df_escroete$numEscroete[i],
                        df_escroete$escroete[i],
                        t[j,1], t[j,2],t[j,3],t[j,4],t[j,5]))
    }
  }
  return(df)
}

#### extraction complete ####
fullExtract <- function(text){
  return (escroeteExtract(text))
}
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS  12.3.1
##
## Matrix products: default
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] fr_BE.UTF-8/fr_BE.UTF-8/fr_BE.UTF-8/C/fr_BE.UTF-8/fr_BE.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices datasets  utils     methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-3 concaveman_1.1.0   ggforce_0.3.3      scales_1.2.0
##  [5] ggrepel_0.9.1      readxl_1.3.1       tidygeocoder_1.0.5 ggraph_2.0.5.9000
##  [9] ggmap_3.0.0        igraph_1.3.0       comparator_0.1.2   forcats_0.5.1
## [13] dplyr_1.0.9        purrr_0.3.4        readr_2.1.2        tidyr_1.2.0
## [17] tibble_3.1.8       ggplot2_3.3.6      tidyverse_1.3.1    stringr_1.4.0.9000
##
## loaded via a namespace (and not attached):
```

```
##  [1] bitops_1.0-7        fs_1.5.2            lubridate_1.8.0     httr_1.4.2
##  [5] tools_4.0.3         backports_1.4.1     utf8_1.2.2          R6_2.5.1
##  [9] DBI_1.1.2           colorspace_2.0-3    withr_2.5.0         sp_1.5-0
## [13] tidyselect_1.1.2    gridExtra_2.3       curl_4.3.2          compiler_4.0.3
## [17] cli_3.3.0           rvest_1.0.2         xml2_1.3.3          proxy_0.4-26
## [21] digest_0.6.29       jpeg_0.1-9          pkgconfig_2.0.3     highr_0.9
## [25] dbplyr_2.1.1        rlang_1.0.4         rstudioapi_0.13     farver_2.1.1
## [29] generics_0.1.3      jsonlite_1.8.0      magrittr_2.0.3      Rcpp_1.0.9
## [33] munsell_0.5.0       fansi_1.0.3         viridis_0.6.2       lifecycle_1.0.1
## [37] stringi_1.7.6       MASS_7.3-53         plyr_1.8.7          grid_4.0.3
## [41] crayon_1.5.0        lattice_0.20-41     graphlayouts_0.8.0  haven_2.4.3
## [45] hms_1.1.1           knitr_1.37          pillar_1.8.0        rjson_0.2.21
## [49] reprex_2.0.1        glue_1.6.2          evaluate_0.15       renv_0.15.4
## [53] modelr_0.1.8        png_0.1-7           vctrs_0.4.1         tzdb_0.2.0
## [57] tweenr_1.0.2        RgoogleMaps_1.4.5.3 cellranger_1.1.0    gtable_0.3.0
## [61] polyclip_1.10-0     clue_0.3-60         assertthat_0.2.1    xfun_0.30
## [65] broom_0.7.12        tidygraph_1.2.1     viridisLite_0.4.0   tinytex_0.37
## [69] cluster_2.1.0       ellipsis_0.3.2
```

```
Sys.time()
```

```
## [1] "2022-08-08 07:40:45 CEST"
```