new_functions.R

jcb

Sat Feb 14 17:05:31 2015

```
# SOMMAIRE
```

```
#
# complet
# sort.data.frame
# aligne.sur.calendrier
# mode.sortie

##============================
#
#   complet
#
#============================
```

@description crée un dataframe du pourcentage de complétion des items du RPU par les hôpitaux @name @param hop vecteur contenant le noms des hôpitaux @param d07 dataframe contenant les RPU à analyser @return un dataframe @example: analyse du fichier pour le mois de juillet 2014

a <- unique(d07$FINESS) # vecteur des noms d'hôpitaux q <- complet(a,d07)

```
complet <- function(hop, d07){
  j = 0;
  for(i in hop){
    x <- h(d07[d07$FINESS==i,]);
    x <- (1-x)*100;
    j = j+1
    if(j == 1)
      t = x
    else
      t = rbind(t,x)
  }
  t <- data.frame(t)
  row.names(t) <- hop
  return(t)
}


##============================
#
#   sort.data.frame
#
#============================
```

@description trie un dataframe @author Mark van der Loo @source http://www.markvanderloo.eu/yaRb/2014/08/15/sort-data-frame/ @example sort(q, by="DP") @example sort(iris, by="Sepal.Length") @example sort(iris, by=c("Species","Sepal.Length")) @example sort(iris, by=1:2) @example sort(iris, by="Sepal.Length",decreasing=TRUE)

```
sort.data.frame <- function(x, decreasing=FALSE, by=1, ... ){
  f <- function(...) order(...,decreasing=decreasing)
  i <- do.call(f,x[by])
  x[i,,drop=FALSE]
}

#=============================================================================
#
#   aligne.sur.calendrier
#
#=============================================================================
```

@description soit un vecteur x de dates non consécutives. Cette fonction insère les jours manquants de façon à former un vecteur de dates continu. @param date1 date de début au format Date ou ISO @param x vecteur de nd de RPU/jour => rownames = date de chaque jour @usage x <- seq("2015-01-10", "2015-01-20", 1) aligne.sur.calendrier("2015-01-01", "2015-01-31", x) sum(is.na(b)) # nb de jours sur la période sans passage > 6 heures mean(is.na(b)) # idem en %

```
aligne.sur.calendrier <- function(date1, date2, x){
  # creéer un calendrier
  calendrier <- seq(from = as.Date(date1), to = as.Date(date2), by = 1)
  a <- as.data.frame(calendrier)

  # transforme p6h.jour en dataframe pour merger ATTENTION: ne pas mettre as.data.frame
  # x <- dx[p14$DPAS > 6*60, "ENTREE"]
  z <- data.frame(x, as.Date(names(x)))
  names(z) <- c("rpu", "date")

  # merging
  b <- merge(a, z, by.y = "date", by.x = "calendrier", all.x = TRUE)

  return(b)
}

##============================
#
#   mode.sortie
#
#============================
```

@description crée un dataframe contenant le total de passages, d'hospitalisation, de mutation, de transferts par jour @author JcB @param dx dataframe contenant au minimum les colonnes ENTREE, MODE_SORTIE @return dataframe contenant une colonne date, passage.jour, hospit.jour, mutations.jour, transfert.jour, taux.hosp @details hospitalisation = mutation + transfert @details taux hospitalisation = hospitalisation / passages @usage dd <- mode.sortie(d14)

```
mode.sortie <- function(dx){
  date1 <- min(as.Date(dx$ENTREE))
  date2 <- max(as.Date(dx$ENTREE))
  passages.jour <- tapply(as.Date(dx$ENTREE), as.Date(dx$ENTREE), length)

  mut <- dx[dx$MODE_SORTIE == "Mutation", "ENTREE"]
  mutations.jour <- tapply(as.Date(mut),  as.Date(mut), length) # a ajuster sur calendrier
```

```
  mutations.jour <- aligne.sur.calendrier(date1, date1, mutations.jour)

  trans <- dx[dx$MODE_SORTIE == "Transfert", "ENTREE"]
  transfert.jour <- tapply(as.Date(trans),  as.Date(trans), length)
  transfert.jour <- aligne.sur.calendrier(date1, date1, transfert.jour)

  hospit.jour <- mutations.jour + transfert.jour

  date <- unique(sort(as.Date(dx$ENTREE)))
  taux.hosp <- round(hospit.jour * 100 / passages.jour, 2)
  sortie <- data.frame(date, passages.jour, hospit.jour, mutations.jour, transfert.jour, taux.hosp)
}
```