TP2 - Logiciel statistique R

Table des matières

I. Importation et mise en forme

Pour importer le jeu de données, nous faisons appel à la focntion `read_xlsx()` de la librairie **readxl** qui facilite l'extraction de données d'Excel vers R. Après avoir importé la base de données sur laquelle nous allons travailler, nous pouvons procéder à l'affichage de quelques les premières lignes.

```
library(readxl)

base_tp2 <- read_xlsx("..//Donnees//Base TP2.xlsx")
head(base_tp2)
```

```
## # A tibble: 6 x 30
##   region departement  sexe   age sit_mat si_chef_men ethnie occupation formation
##    <dbl>       <dbl> <dbl> <dbl>   <dbl>       <dbl>  <dbl>      <dbl>     <dbl>
## 1      5          53     2    35       1           1     10         22         4
## 2      5          53     1    50       1           2      1          1         1
## 3      2          22     2    35       1           1      1          1         3
## 4      1          12     1    25       1           3      3         22         5
## 5      5          52     2    60       1           1     77          1         3
## 6      3          31     1    36       1           1      3          1        99
## # i 21 more variables: niveau_alphabs <dbl>, types_varietes <chr>,
## #   types_varietes_1 <dbl>, types_varietes_2 <dbl>, criteres_var <chr>,
## #   criteres_var_1 <dbl>, criteres_var_2 <dbl>, criteres_var_3 <dbl>,
## #   criteres_var_4 <dbl>, criteres_var_5 <dbl>, criteres_var_6 <dbl>,
## #   criteres_var_7 <dbl>, criteres_var_8 <dbl>, criteres_var_9 <dbl>,
## #   criteres_var_10 <dbl>, criteres_var_11 <dbl>, criteres_var_12 <dbl>,
## #   criteres_var_13 <dbl>, criteres_var_14 <dbl>, criteres_var_15 <dbl>, ...
```

Pour obtenir des informations détaillées sur une base de données, nous pouvons tiliser la fonction `str()`.

```
str(base_tp2)
```

```
## tibble [53 x 30] (S3: tbl_df/tbl/data.frame)
##  $ region          : num [1:53] 5 5 2 1 5 3 3 2 2 4 ...
##  $ departement     : num [1:53] 53 53 22 12 52 31 32 22 22 41 ...
##  $ sexe            : num [1:53] 2 1 2 1 2 1 2 1 1 1 ...
##  $ age             : num [1:53] 35 50 35 25 60 36 25 56 55 80 ...
```

```
##  $ sit_mat         : num [1:53] 1 1 1 1 1 1 1 1 1 1 ...
##  $ si_chef_men     : num [1:53] 1 2 1 3 1 1 3 2 2 2 ...
##  $ ethnie          : num [1:53] 10 1 1 3 77 3 3 1 1 2 ...
##  $ occupation      : num [1:53] 22 1 1 22 1 1 15 1 1 1 ...
##  $ formation       : num [1:53] 4 1 3 5 3 99 1 1 1 1 ...
##  $ niveau_alphabs  : num [1:53] NA 0 NA NA NA NA 1 0 0 0 ...
##  $ types_varietes  : chr [1:53] "2" "1" "1" "1" ...
##  $ types_varietes_1: num [1:53] 0 1 1 1 1 1 1 1 0 1 ...
##  $ types_varietes_2: num [1:53] 1 0 0 0 0 1 0 0 1 0 ...
##  $ criteres_var    : chr [1:53] "1 12 13" "1 4 6 7 11 13" "1 5 6" "1 4 6 7 14 15" ...
##  $ criteres_var_1  : num [1:53] 1 1 1 1 1 1 1 0 1 1 ...
##  $ criteres_var_2  : num [1:53] 0 0 0 0 1 1 0 0 1 0 ...
##  $ criteres_var_3  : num [1:53] 0 0 0 0 1 0 0 0 0 0 ...
##  $ criteres_var_4  : num [1:53] 0 1 0 1 1 0 1 0 0 0 ...
##  $ criteres_var_5  : num [1:53] 0 0 1 0 0 0 0 0 0 0 ...
##  $ criteres_var_6  : num [1:53] 0 1 1 1 1 0 0 0 0 0 ...
##  $ criteres_var_7  : num [1:53] 0 1 0 1 1 0 0 0 0 0 ...
##  $ criteres_var_8  : num [1:53] 0 0 0 0 1 0 0 0 0 0 ...
##  $ criteres_var_9  : num [1:53] 0 0 0 0 1 0 0 0 0 0 ...
##  $ criteres_var_10 : num [1:53] 0 0 0 0 0 0 0 0 0 0 ...
##  $ criteres_var_11 : num [1:53] 0 1 0 0 1 0 0 1 1 1 ...
##  $ criteres_var_12 : num [1:53] 1 0 0 0 0 0 0 0 0 0 ...
##  $ criteres_var_13 : num [1:53] 1 1 0 0 1 1 0 0 0 0 ...
##  $ criteres_var_14 : num [1:53] 0 0 0 1 1 1 0 0 0 0 ...
##  $ criteres_var_15 : num [1:53] 0 0 0 1 0 0 0 0 0 0 ...
##  $ criteres_var_16 : num [1:53] 0 0 0 0 0 0 0 0 0 0 ...
```

La base de données est composée de 53 observations et de 30 variables comme nous montre la ligne ci-après grâce à la fonction `dim()`

```
dim(base_tp2)
```

```
## [1] 53 30
```

II. Recodage et labélisation

II.1. Recodage

L'objectif de cette partie est de données des valeurs descriptives aux modalités. Pour ce faire, nous utilisons la fonction `recode()` de **dplyr**. La description des modalités de chaque variable est disposée dans le tableau ci-après: :::table

:::

```
base_tp2$sexe = dplyr::recode_factor(base_tp2$sexe,
                      '1'="Homme",
                      '2'="Femme")
```

```
class(base_tp2$sexe)
```

```
## [1] "factor"
```

```
typeof(base_tp2$sexe)
```

```
## [1] "integer"
```

```
base_tp2$sit_mat = dplyr::recode_factor(base_tp2$sit_mat,
                      '1'="Marie(e)",
                      '3'="Veuf(ve)",
                      '4'="Divorce(e)",
                      '5'="Separe(e)",
                      '6'="Celibataire")
```

```
base_tp2$si_chef_men = dplyr::recode_factor(base_tp2$si_chef_men,
                      '1'="femme du chef de menage",
                      '3'="chef de menage",
                      '4'="fils-fille du chef de menage",
                      '99'="Autres")
```

```
base_tp2$ethnie = dplyr::recode_factor(base_tp2$ethnie,
                      '1'="Wolof",
                      '2'="Pulaar/Toucouleur",
                      '3'="Serere",
                      '4'="Mandika/Bambara",
                      '5'="Soninke",
```

```
                      '6'="Diola",
                      '7'="Manjack",
                      '8'="Bainouk",
                      '9'="Maures",
                      '10'="Balante",
                      '77'="Autre")
```

```
base_tp2$occupation =
  dplyr::recode_factor(base_tp2$occupation,
                      '1'="Agriculture, Elevage, Sylvicuture, Peche",
                      '2'="Activites extractives",
                      '3'="Activites de fabrications (Artisanat)",
                      '4'="Activite de transformation",
                      '5'="Production et distribution d'electricite\
                      et de gaz",
                      '6'="Production et distribution d'eau, assainissement,\
                      traitement des dechets et depollution")
```

```
base_tp2$formation = dplyr::recode_factor(base_tp2$formation,
                      '1'="Non scolarise",
                      '2'="Elementaire",
                      '3'="Moyen",
                      '4'="Secondaire",
                      '5'="Licence",
                      '6'="Master",
                      '7'="Doctorat",
                      '99'="Ne sait pas")
```

```
base_tp2$niveau_alphabs = dplyr::recode_factor(base_tp2$niveau_alphabs,
                      '0'="Sans niveau",
                      '1'="Sait lire dans une langue",
                      '2'="Sait lire et ecrire dans une langue")
```

```
YesNo = function(data, var){
  data[[var]] = dplyr::recode_factor(data[[var]],
                      '0'="Non",
                      '1'="Oui",
```

```
                      .default = "Missing")
}
```

```
## Types de variétés
YesNo(base_tp2, "types_varietes_1")

YesNo(base_tp2, "types_varietes_2")



## Critères de choix de variétés
vars2 = paste0("criteres_var_",1:16)

for (var in vars2){
  base_tp2[[var]] = YesNo(base_tp2, var)
}
```

II.2. Labélisation

Labéliser une variable consiste à lui attribuer une description explicite afin de mieux comprendre sa signification. Pour ce faire, utilisons la fonction `apply_labels()` de la librairie **expss**.

```
library(expss)

base_tp2 <- base_tp2|> expss::apply_labels(
  region = "Region", departement = "Departement",
  sexe = "Sexe", age = "Age",
  sit_mat = "Situation matrimoniale",
  si_chef_men = "Statut dans le menage",
  ethnie = "Ethnie", occupation = "Occupation",
  formation = "Formation",
  niveau_alphabs = "Niveau d'alphabetisation",
  types_varietes = "Quelles sont les varietes que vous utilisez\
  pour la production de sesame ?",
  types_varietes_1 = "Traditionnelles",
  types_varietes_2 = "Améliorées",
  criteres_var = "Quels sont les criteres de choix des varietes de sesame ?",
```

```
  criteres_var_1 = "Rendements eleves",
  criteres_var_2 = "Taille des graines",
  criteres_var_3 = "Resistantes aux maladies/ravageurs",
  criteres_var_4 = "Tolerantes aux secheresses",
  criteres_var_5 = "Tolerantes aux inondations",
  criteres_var_6 = "Faible charge de travail",
  criteres_var_7 = "Faibles quantites d'intrants",
  criteres_var_8 = "Facile à transformer",
  criteres_var_9 = "Haute teneur en huile",
  criteres_var_10 = "Haut rendement apres transformation",
  criteres_var_11 = "Demande sur le marché",
  criteres_var_12 = "Bon gout",
  criteres_var_13 = "Belle couleur",
  criteres_var_14 = "Haut rendement en fourrages",
  criteres_var_15 = "Qualité du fourrage",
  criteres_var_16 = "Autres a specifier"
)
```

III. Analyse descriptive

Dans cette partie, Pour automatiser les affichages de tableaux et de graphes dans le cadre de l'analyse descriptive, nous allons développer deux fonctions : univarie() et bivarie().

III.1. La fonction univarie

Elle est conçue pour les analyses univariées. Elle prend en entrée la dataframe (**data**) et la variable à étudier (**var**). En fonction du type de cette variable, qu'elle soit quantitative ou qualitative, la fonction affiche soit un graphique (uniquement pour les variables qualitatives) soit un tableau. Pour les variables de type discrète, elle génère un tableau présentant les statistiques de tendance centrale telles que la moyenne, la médiane et les quartiles.
Pour les variables qualitatives, l'utilisateur peut choisir entre une sortie graphique ou tabulaire (valeur par défaut) ou graphique en spécifiant **plot=TRUE et tab=FALSE**. Par défaut, la sortie graphique affiche un diagramme circulaire, mais l'utilisateur peut opter pour un diagramme en barres en spécifiant **type="bar"**. Il est également possible d'afficher les graphiques en fonction des fréquences (**props=TRUE**) ou des effectifs (**props=FALSE**).

```r
univarie <- function(data, var, tab = TRUE,
                     plot=FALSE,type="",props = TRUE)
  {

  ## Librairies
  library(gtsummary)
  library(ggplot2)
  library(gt)
  library(dplyr)
  attach(data)

  ## Variable qualitative
    if (is.character(var)==TRUE | is.factor(var)==TRUE){

      # Tableau des effectifs
      df = data.frame(table(var))
      names(df) = c("var","eff")

      # Tableau des fréquences
      dfprop = data.frame(round(proportions(table(var))*100,2))
      names(dfprop) = c("var","freq")

      # Tableau de resultat
      table = data.frame(var) |> tbl_summary()

      # Diagramme en secteurs
      plot1 = ggplot(dfprop,
                aes(x = "", y = freq, fill =var)) +
              geom_bar(stat = "identity", width=1) +
              coord_polar(theta = "y", start = 0) +
              geom_text(aes(label = paste(freq,"%")),color="white",
                        position = position_stack(vjust=0.5))+
              labs(fill = attr(var,"label")) +
              theme_void()
```

```
    # Diagramme en barres avec les effectifs
    plot2_eff = ggplot(df,
              aes(x = var, y = eff, fill = var)) +
            geom_bar(stat = "identity", width=1) +
            labs(fill = attr(var,"label")) +
            theme(legend.position = "none")

    # Diagramme en barres avec les fréquences
    plot2_freq = ggplot(dfprop,
              aes(x = var, y = freq, fill = var)) +
            geom_bar(stat = "identity", width=1) +
            labs(fill = attr(var,"label")) +
            theme(legend.position = "none")
  }

## Variable discrète
else if (is.numeric(var)==TRUE){

    # Tri à plat
    dftri = data.frame(var)
    names(dftri) = "var"

    # Tableau des statistiques descriptives (Tendance centrale)
    min = min(var, na.rm = TRUE)
    q1 = quantile(var, na.rm = TRUE, 0.25)
    median = median(var, na.rm = TRUE)
    mean = mean(var, na.rm = TRUE)
    sd = sd(var, na.rm = TRUE)
    variance = sd^2
    q3 = quantile(var, na.rm = TRUE, 0.75)
    max = max(var, na.rm = TRUE)
    stats =  c(min,q1,median,mean,sd,variance,q3,max)
    statsLab = c("Minimum", "1er quartile","Médiane",
                 "Moyenne", "Ecart-type", "Variance",
                 "3e quartile", "Maximum")
```

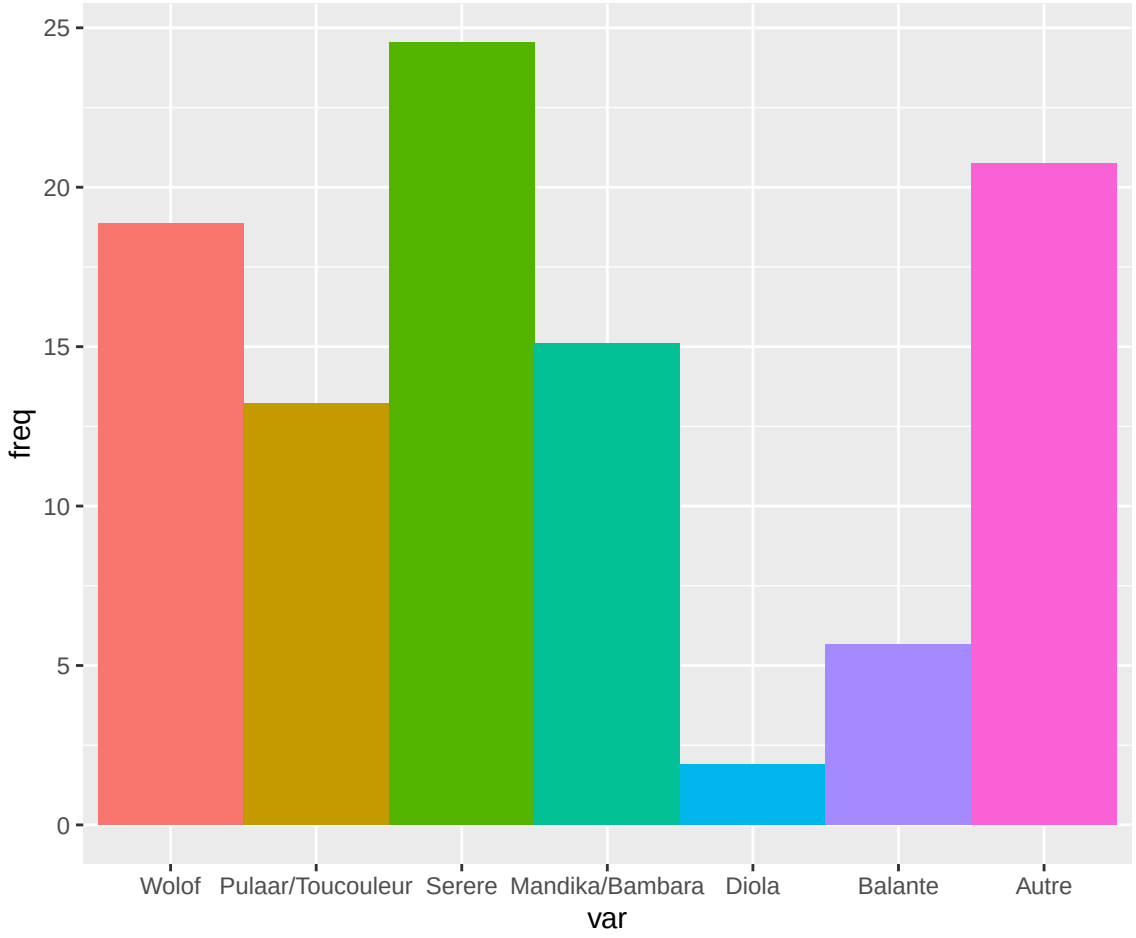
```
      table = data.frame(
        Statistiques = statsLab,
        Valeurs = stats
      )
  }

  ## Sorties : graphiques ou tableaux
  if (tab == TRUE){result = table}

  else if(plot == TRUE){
    tab = FALSE
    if (type == "pie"){result = plot1}
    else{
      if(props == FALSE){result = plot2_eff}
      else{result = plot2_freq}
      }
  }

  else if (plot == TRUE){
    tab = FALSE
    result = plot2
  }
  else {result = print("Spécifier les arguments")}

  return(result)
}
```

```
univarie(base_tp2, age)
```

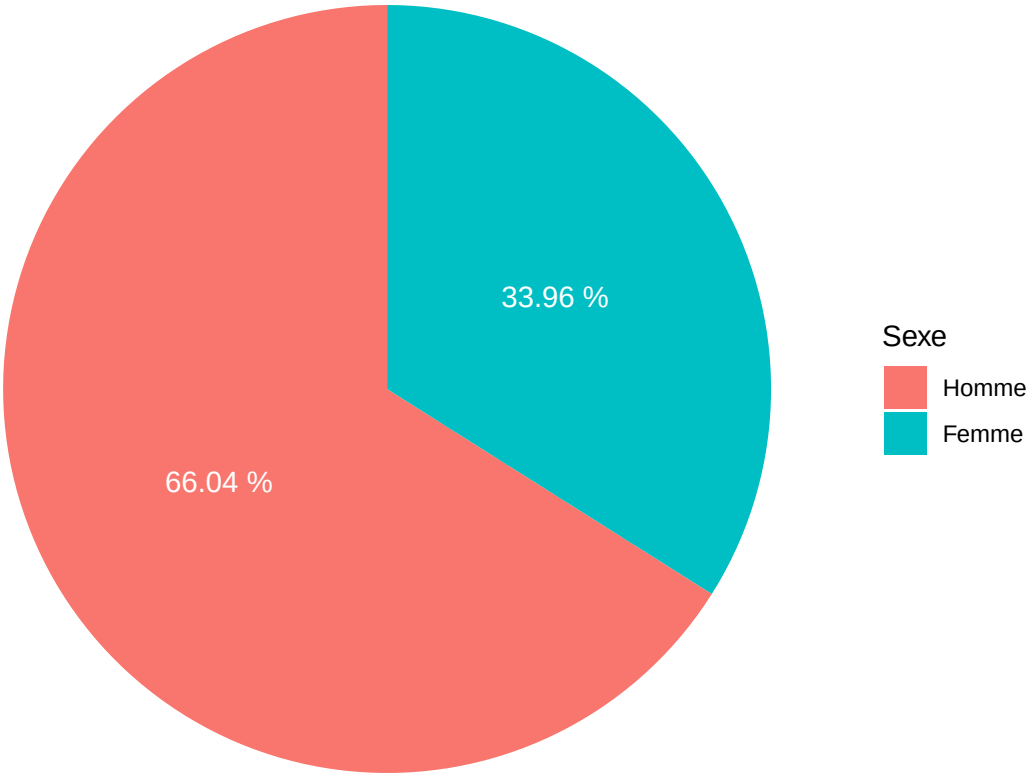
```
##   Statistiques   Valeurs
## 1      Minimum  24.00000
## 2 1er quartile  39.00000
## 3      Médiane  49.00000
## 4      Moyenne  48.60377
## 5   Ecart-type  12.91293
## 6     Variance 166.74383
```

```
## 7  3e quartile  58.00000
## 8       Maximum  80.00000
```

```
univarie(base_tp2, ethnie, plot = TRUE, tab = FALSE, props = TRUE)
```



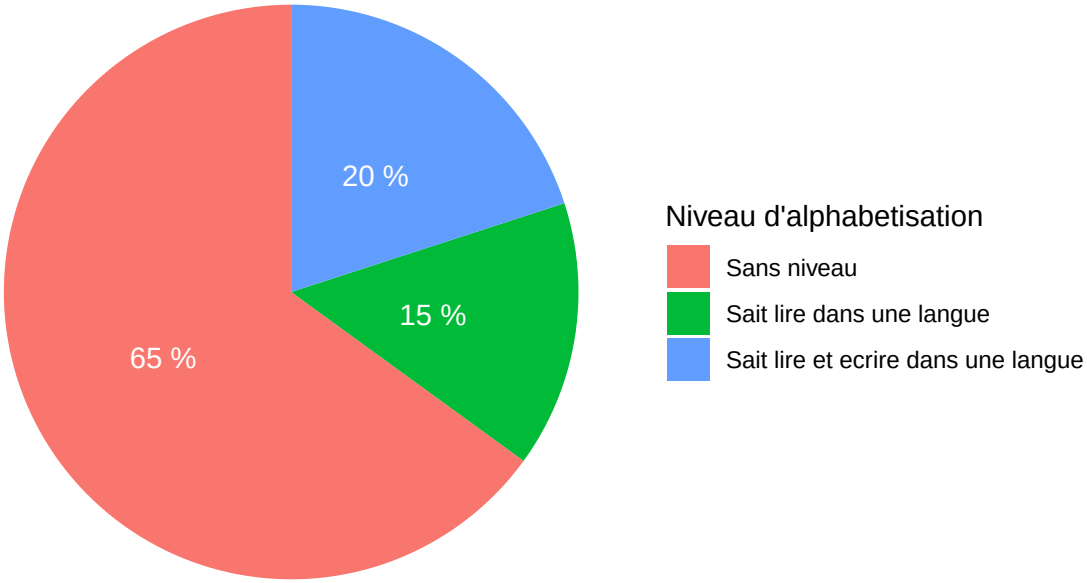

```
univarie(base_tp2, sexe, plot = TRUE, tab = FALSE, props = TRUE, type="pie")
```




```
univarie(base_tp2, formation, tab = TRUE)
```

Characteristic	**N = 53**
Formation	
Non scolarise	30 (57%)
Elementaire	10 (19%)
Moyen	7 (13%)
Secondaire	3 (5.7%)
Licence	2 (3.8%)
Ne sait pas	1 (1.9%)

```
univarie(base_tp2, niveau_alphabs, plot = TRUE,
         tab = FALSE, props = TRUE, type="pie")
```



III.2. La fonction bivarie

Elle prend en argument le jeu de données (**data**) et les deux variables (var1 , var2). Il est à noter que cette fonction ne prend en charge que des variables qualitatives. Avec son option par défaut **all=TRUE** , elle affiche le tableau croisé ainsi que le diagramme des deux variables. Tout comme la fonction **univarie()**, elle peut afficher uniquement le graphique ou le tableau avec les mêmes options. Ces fonctions nous permettront de généraliser les statistiques descriptives en fonction du type de variable.

```
bivarie <- function(df, var1, var2, all=TRUE){
  library(gtsummary)
  library(ggplot2)
  attach(df)

  ## Variable qualitative
  if (is.character(var1)==TRUE){
```

```
    df_plot = data.frame(table(var1, var2))
    names(df_plot) = c("Var1","Var2","Freq")

    plot = ggplot(df_plot,
                aes(x = Var1, y = Freq, fill = Var2)) +
             geom_bar(stat = "identity", position = "fill")

    table = tbl_cross(df, df$var1, df$var2)
  }

  ## Sorties : graphiques ou tableaux
  if (all==TRUE){res = list(table,plot)}
  else if(all=="table"){res = table}
  else if(all=="plot"){res = plot}
  else {res = print("Choisir l'option table ou plot")}

  return(res)
}
```

```
bivarie(base_tp2, sexe, ethnie)[[1]]
```

```
## function (..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no",
##     "ifany", "always"), dnn = list.names(...), deparse.level = 1)
## {
##     list.names <- function(...) {
##         l <- as.list(substitute(list(...)))[-1L]
##         if (length(l) == 1L && is.list(..1) && !is.null(nm <- names(..1)))
##             return(nm)
##         nm <- names(l)
##         fixup <- if (is.null(nm))
##             seq_along(l)
##         else nm == ""
##         dep <- vapply(l[fixup], function(x) switch(deparse.level +
##             1, "", if (is.symbol(x)) as.character(x) else "",
##             deparse(x, nlines = 1)[1L]), "")
##         if (is.null(nm))
```

```
##             dep
##         else {
##             nm[fixup] <- dep
##             nm
##         }
##     }
##     miss.use <- missing(useNA)
##     miss.exc <- missing(exclude)
##     useNA <- if (miss.use && !miss.exc && !match(NA, exclude,
##         nomatch = 0L))
##         "ifany"
##     else match.arg(useNA)
##     doNA <- useNA != "no"
##     if (!miss.use && !miss.exc && doNA && match(NA, exclude,
##         nomatch = 0L))
##         warning("'exclude' containing NA and 'useNA' != \"no\"' are a bit contradicti
##     args <- list(...)
##     if (length(args) == 1L && is.list(args[[1L]])) {
##         args <- args[[1L]]
##         if (length(dnn) != length(args))
##             dnn <- paste(dnn[1L], seq_along(args), sep = ".")
##     }
##     if (!length(args))
##         stop("nothing to tabulate")
##     bin <- 0L
##     lens <- NULL
##     dims <- integer()
##     pd <- 1L
##     dn <- NULL
##     for (a in args) {
##         if (is.null(lens))
##             lens <- length(a)
##         else if (length(a) != lens)
##             stop("all arguments must have the same length")
##         fact.a <- is.factor(a)
##         if (doNA)
```

```
##             aNA <- anyNA(a)
##         if (!fact.a) {
##             a0 <- a
##             op <- options(warn = 2)
##             on.exit(options(op))
##             a <- factor(a, exclude = exclude)
##             options(op)
##         }
##         add.na <- doNA
##         if (add.na) {
##             ifany <- (useNA == "ifany")
##             anNAc <- anyNA(a)
##             add.na <- if (!ifany || anNAc) {
##                 ll <- levels(a)
##                 if (add.ll <- !anyNA(ll)) {
##                   ll <- c(ll, NA)
##                   TRUE
##                 }
##                 else if (!ifany && !anNAc)
##                   FALSE
##                 else TRUE
##             }
##             else FALSE
##         }
##         if (add.na)
##             a <- factor(a, levels = ll, exclude = NULL)
##         else ll <- levels(a)
##         a <- as.integer(a)
##         if (fact.a && !miss.exc) {
##             ll <- ll[keep <- which(match(ll, exclude, nomatch = 0L) ==
##                 0L)]
##             a <- match(a, keep)
##         }
##         else if (!fact.a && add.na) {
##             if (ifany && !aNA && add.ll) {
##                 ll <- ll[!is.na(ll)]
```

```
##                 is.na(a) <- match(a0, c(exclude, NA), nomatch = 0L) >
##                   0L
##             }
##             else {
##                 is.na(a) <- match(a0, exclude, nomatch = 0L) >
##                   0L
##             }
##         }
##         nl <- length(ll)
##         dims <- c(dims, nl)
##         if (prod(dims) > .Machine$integer.max)
##             stop("attempt to make a table with >= 2^31 elements")
##         dn <- c(dn, list(ll))
##         bin <- bin + pd * (a - 1L)
##         pd <- pd * nl
##     }
##     names(dn) <- dnn
##     bin <- bin[!is.na(bin)]
##     if (length(bin))
##         bin <- bin + 1L
##     y <- array(tabulate(bin, pd), dims, dimnames = dn)
##     class(y) <- "table"
##     y
## }
## <bytecode: 0x0000022ad9b15260>
## <environment: namespace:base>
```