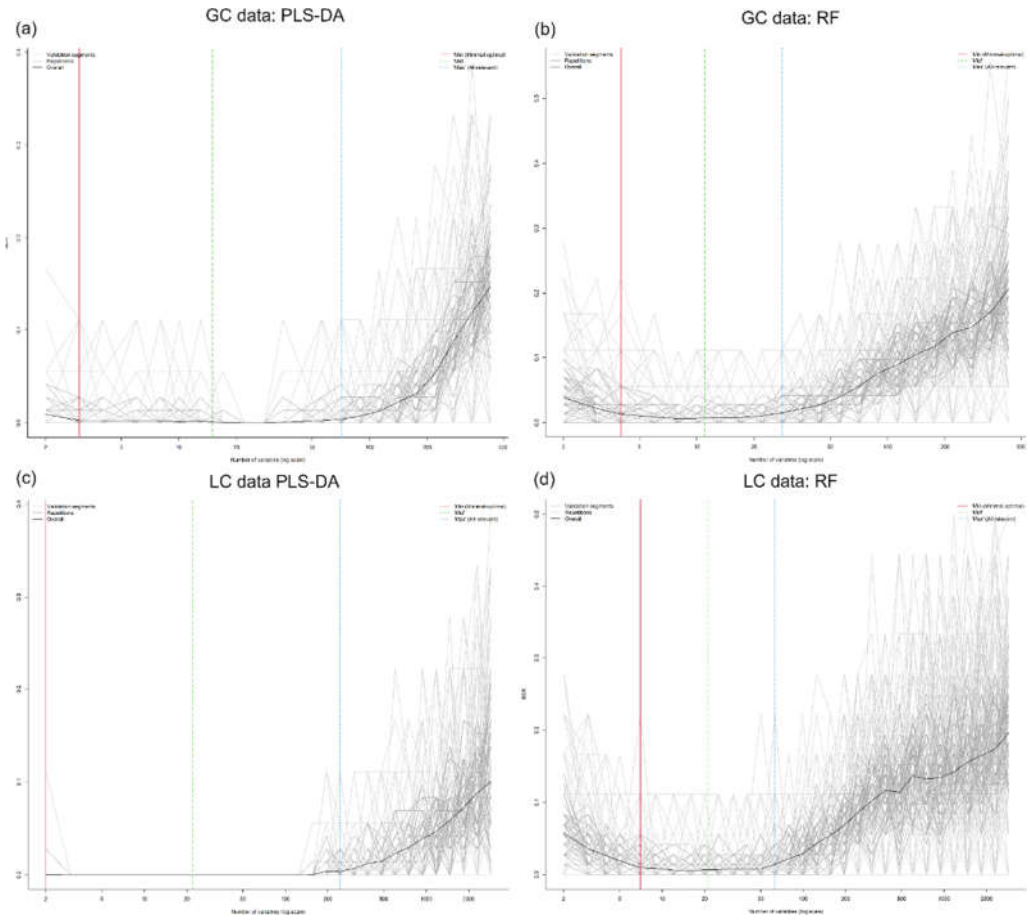
Supporting Information Material: A Statistical Methodology to Assesses the Modulation of Wine Metabolome and its contribution to the Sensory Attributes

José Manuel Muñoz-Redondo[1*], Belén Puertas[2], Gema Pereira-Caro[1], José Luis Ordoñez-Diaz[1], María José Ruiz-Moreno, Emma Cantos-Villar[2], José Manuel Moreno-Rojas[1*]

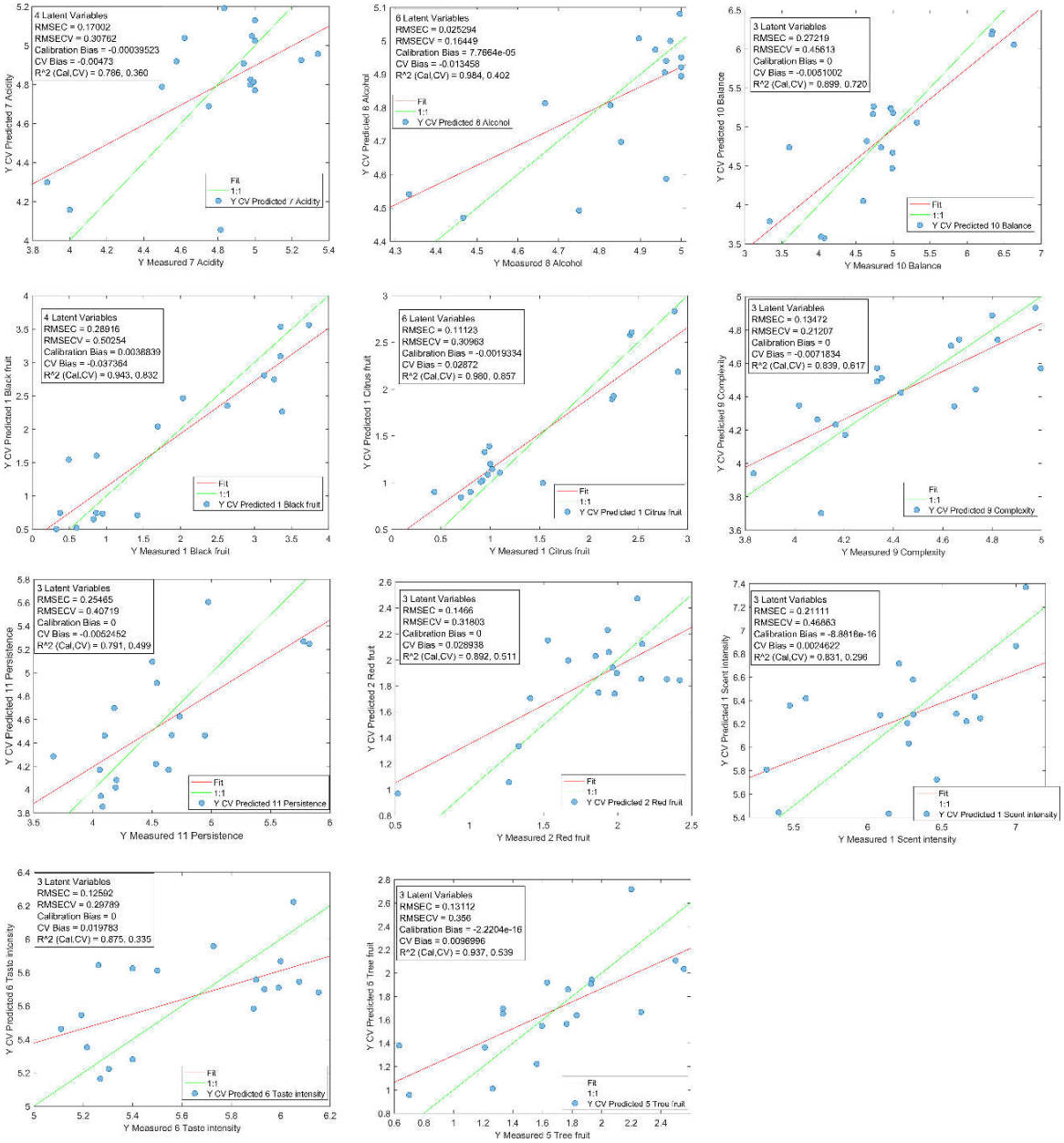
1 Department of food Science and Health. Andalusian Institute of Agricultural and Fisheries Research and Training (IFAPA). Alameda del Obispo Avda, Menéndez Pidal, s/n. 14004, Córdoba, Spain. GPC: mariag.pereira@juntadeandalucia.es; JLOD: josel.ordonez@juntadeandalucia.es; MJRM: mariaj.ruiz.moreno@juntadeandalucia.es.

2 Department of food Science and Health. Andalusian Institute of Agricultural and Fisheries Research and Training (IFAPA). Cañada de la Loba, 11471, Jerez de la Frontera, Cádiz, Spain. BP: mariab.puertas@juntadeandalucia.es; ECV: emma.cantos@juntadeandalucia.es.

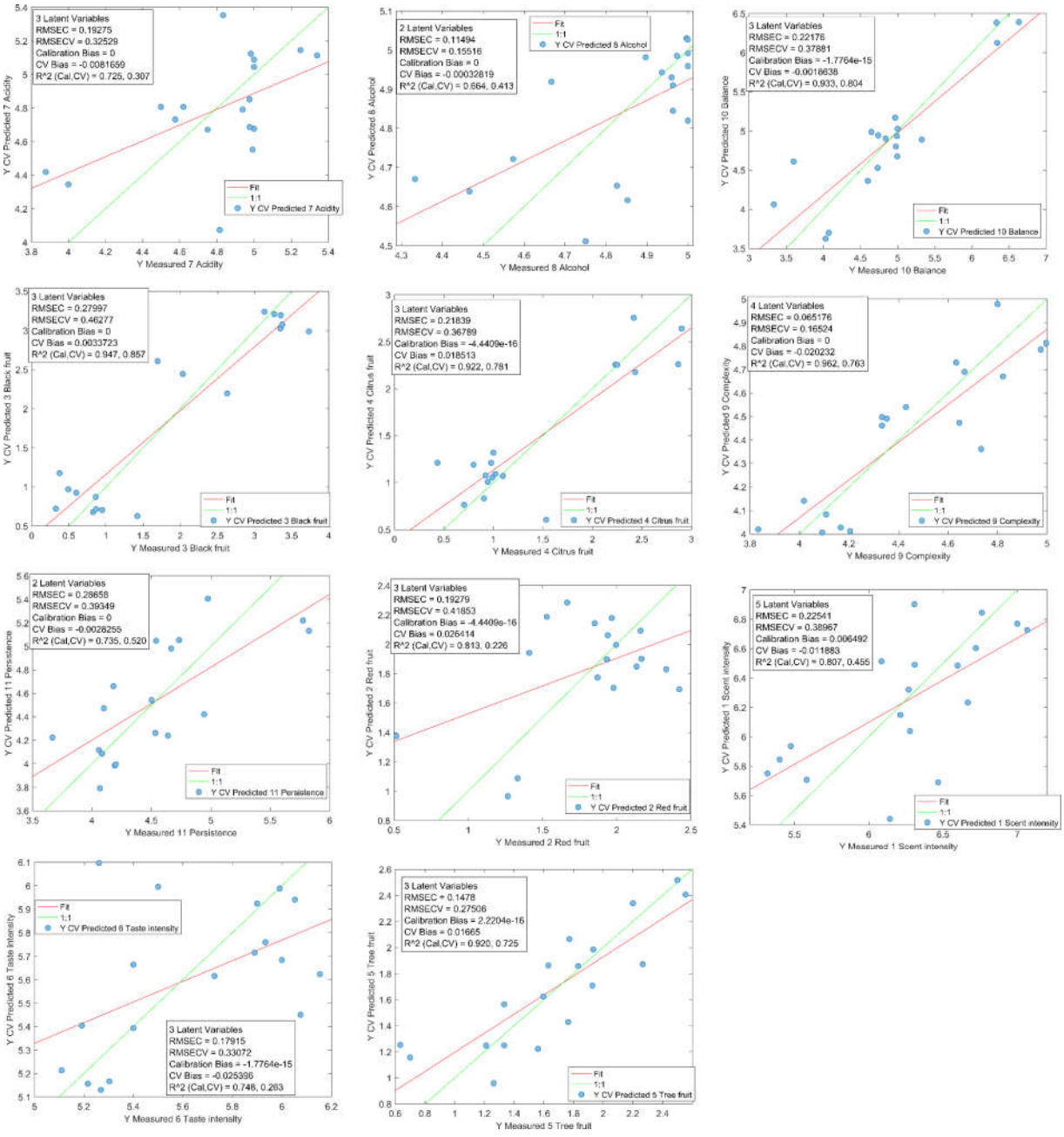
* Correspondence: JMMRe: josem.munoz.redondo@juntadeandalucia.es; +34 666 306 651; JMMRo: josem.moreno.rojas@juntadeandalucia.es; 34 671 532 758.



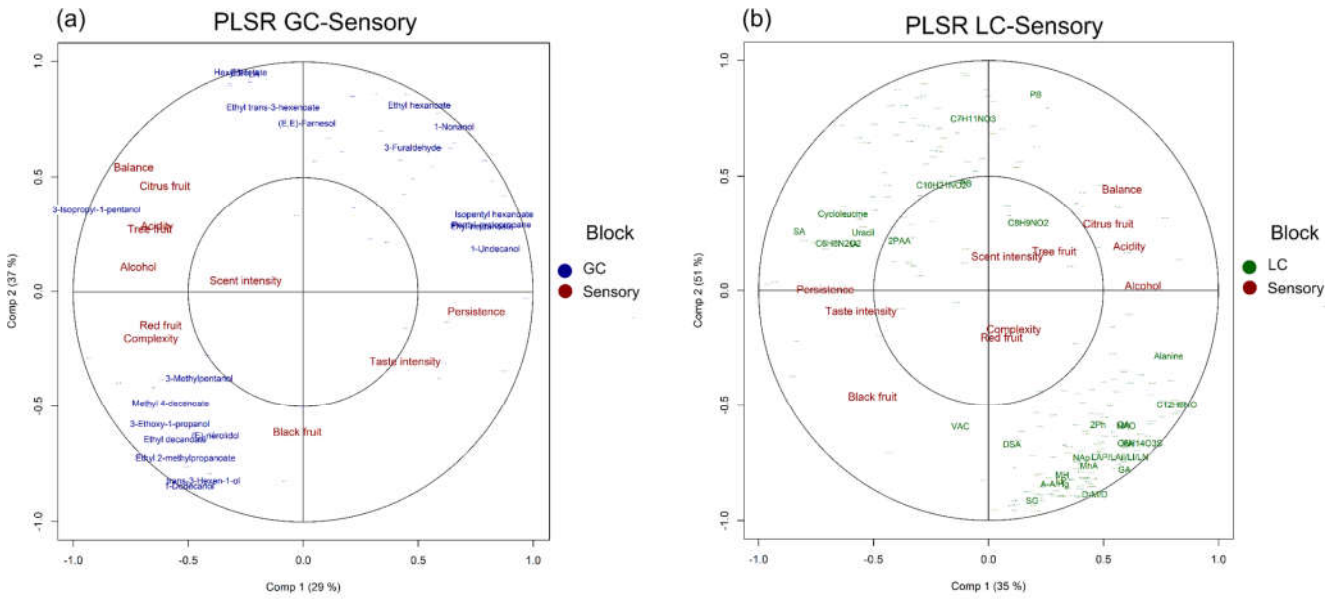
Supporting Figure S1. Balanced error rates (BER) obtained for the partial least squares discriminant analysis (PLS-DA) and random forest (RF) double-cross validated models used to select for the maximum number of relevant variables in GC data (a and b) and LC data (c and d). The blue dotted line indicates the all-relevant variables selected for each model.



Supporting Figure S2. Partial least squares regression (PLSR) performed to predict the sensory descriptors from the volatile profile.



Supporting Figure S3. Partial least squares regression (PLSR) performed to predict the sensory descriptors from the non-volatile profile.



Supporting Figure S4. Correlation loadings plot from the partial least squares regression (PLS2) performed on the GC-sensory and LC-sensory data sets.

Supporting Material R script S1. XCMS optimised parameter settings with the IPO package for Positive and Negative ESI modes.

```
#####################################################

## XCMS optimised parameter settings and CAMERA

## Positive ESI mode.

#####################################################


library(xcms)

library(Rmpi)


xset <- xcmsSet(
```

```
  method = "centWave",
  peakwidth      = c(21.088, 86),
  ppm            = 54.5,
  noise          = 0,
  snthresh       = 10,
  mzdiff         = 0.015,
  prefilter      = c(3, 100),
  mzCenterFun    = "wMean",
  integrate      = 1,
  fitgauss       = FALSE,
  verbose.columns = FALSE)
xset <- retcor(
  xset,
  method        = "obiwarp",
  plottype      = "none",
  distFunc      = "cor_opt",
  profStep      = 1,
  center        = 3,
  response      = 1,
  gapInit       = 0,
```

```
  gapExtend      = 2.1,
  factorDiag     = 2,
  factorGap      = 1,
  localAlignment = 0)
xset <- group(
  xset,
  method  = "density",
  bw      = 22,
  mzwid   = 0.035,
  minfrac = 0.7,
  minsamp = 1,
  max     = 50)

xset <- fillPeaks(xset)
```

Supporting Material R script S2. Automatic annotation of metabolites from METLIN present in a database.

```
#' Title WinMetID
#' This function selects the metabolites annotated from METLIN which are present in a database (excel file)
```

```
#' @param nameFeature vector containing a ID label of each metabolite for the subsequent identification

#' @param massMetQ vector containing the neutral mass of each metabolite

#' @param ppm vector containing the ppm error (if available, else set missing value) from METLIN search

#' @param error vector containing the max error allowed to match putative identification

#' @param adduct vector containing the adduct of each metabolite from METLIN search

#' @param rt vector containing the retention times of each metabolite

#' @param formula vector containing the formula of each metabolite (if available, else set missing value)

#' @param excel.name string variable with the name of the excel file follwed by .xlsx

#'

#'

#' @return a matrix containing the metabolites annotated from METLIN which are present in a database (excel file)

#' @author José Manuel Muñoz-Redondo. josem.munoz.redondo@juntadeandalucia.es. IFAPA Alameda del Obispo (Spain)

#'

#' #'################################################

#'

WinMetID = function(nameFeature, massMetQ, error, ppm, adduct, rt, formula, excel.name){
```

```
library(readxl)

#' Load Winmet data (in the same folder than the function)

dataWinmet = read_excel("/Users/josem/Google Drive/Functions/3.
WinmetSelect/Winmet.xlsx",sheet = 1)

massWinmet = dataWinmet$Mass



massMatch = compMatch = famMatch = cassMatch = list()

n.massMatch = all.massMatch = compMetQ = massSave = ppmSave = adductSave =

  rtSave = formulaSave = c()

count = 1

for (i in 1:length(nameFeature)) {

  mass.Selected = massWinmet[which(massWinmet > c(massMetQ[i]) - error/10000 &

                                   massWinmet < c(massMetQ[i]) + error/10000)]

  comp.Selected = dataWinmet$Metabolite[which(massWinmet > c(massMetQ[i]) - error/10000
&

                                              massWinmet < c(massMetQ[i]) + error/10000)]

  fam.Selected = dataWinmet$Familty[which(massWinmet > c(massMetQ[i]) - error/10000 &

                                          massWinmet < c(massMetQ[i]) + error/10000)]

  cass.Selected = dataWinmet$CAS[which(massWinmet > c(massMetQ[i]) - error/10000 &

                                       massWinmet < c(massMetQ[i]) + error/10000)]

  if (length(mass.Selected) > 0) {
```

```
    compMetQ = c(compMetQ, nameFeature[i])
    massSave = c(massSave, massMetQ[i])
    ppmSave = c(ppmSave, ppm[i])
    adductSave = c(adductSave, adduct[i])
    rtSave = c(rtSave, rt[i])
    formulaSave = c(formulaSave, formula[i])
  }
  massMatch[[count]] = mass.Selected
  compMatch[[count]] = comp.Selected
  famMatch[[count]] = fam.Selected
  cassMatch[[count]] = cass.Selected
  n.massMatch[i] = ifelse(length(mass.Selected) > 0, 1, 0)
  all.massMatch[i] = length(mass.Selected)
  count = c(count + n.massMatch[i]) # It allows to order the data
}

total.massMatch = sum(n.massMatch) # Number of mass matches (Rows)
max.massMatch = max(all.massMatch) # Maximum mass match for one mass (Columns)
results = matrix(nrow = total.massMatch, ncol = c(max.massMatch * 3 + 6))
colnames(results) = c("Features MetaboQuest", "QueryMass", "ppm", "Adduct", "rt", "formula",
```

```
                rep(c("Metabolites WinMet", "Family", "CAS"), max.massMatch))




# Saving results in a data matrix to export

for (j in 1:total.massMatch) {

  results[j,1] = compMetQ[j] # First column: name of MetaboQuest Features

  results[j,2] = massSave[j]

  results[j,3] = ppmSave[j]

  results[j,4] = adductSave[j]

  results[j,5] = round(rtSave[j],2)

  results[j,6] = formulaSave[j]

  for (k in 1:length(compMatch[[j]])) {

    results[j,c(3*k+4)] = compMatch[[j]][k] # Names from Winmet

    results[j,c(3*k+5)] = famMatch[[j]][k]

    results[j,c(3*k+6)] = cassMatch[[j]][k]

  }

}



marked.feature = c()

levels = levels(as.data.frame(results)$`Features MetaboQuest`)
```

```
  for (i in 1:length(levels)) {
    features = results[,1]
    marked.feature[i] = which(features == levels[i])[1]
  }
  results[is.na(results)] = ""

  library(xlsx)
  write.xlsx(results[marked.feature,], excel.name)

  return(list("results" = results[marked.feature,]))
}

#############################################################################################
#################################################################
#############################################################################################
#################################################################

################################################
## XCMS optimised parameter settings and CAMERA
## Negative ESI mode.
```

```
################################################


library(xcms)

library(Rmpi)



xset <- xcmsSet(

  method = "centWave",

  peakwidth       = c(23.2, 77),

  ppm             = 25,

  noise           = 0,

  snthresh        = 10,

  mzdiff          = -0.00705,

  prefilter       = c(3, 100),

  mzCenterFun     = "wMean",

  integrate       = 1,

  fitgauss        = FALSE,

  verbose.columns = FALSE)

xset <- retcor(

  xset,

  method        = "obiwarp",
```

```
  plottype       = "none",
  distFunc       = "cor_opt",
  profStep       = 0.8695,
  center         = 1,
  response       = 1,
  gapInit        = 0.496,
  gapExtend      = 2.7,
  factorDiag     = 2,
  factorGap      = 1,
  localAlignment = 0)
xset <- group(
  xset,
  method  = "density",
  bw      = 12.4,
  mzwid   = 0.003,
  minfrac = 0.94,
  minsamp = 1,
  max     = 50)

xset <- fillPeaks(xset)
```