Notebook

This is the source code used in the manuscript:

'Machine learning analysis of mineral chemistry in pyrite grains from the Jacobina gold deposits, São Francisco Craton, Brazil: geochemical patterns and implications to mineral exploration'

authored by: "Guilherme Ferreira" date: "02/03/2022"

The following code was written in R (4.1.2)

Abstract

We applied machine learning (ML) to process LA-ICP-MS data (45 elements) with 441 samples of pyrite from gold-bearing quartz-pebble-metaconglomerate from the Serra de Jacobina deposits in the São Francisco Craton, Brazil. First, the pyrite samples were described by optical and scanning electron microscopy to gather information about the texture differences. Then, the pyrite grains were classified according to their source and stratigraphical level: detrital and epigenetic pyrite from the mineralized Jacobina Group and pyrite from the basement or intrusive rocks. We used Agglomerative Clustering methods to evaluate the trace elements patterns according to pyrite group, mineral source, and stratigraphic levels. Then, we implemented the Uniform Manifold Approximation and Projection technique (UMAP) to reduce the dimensionality of data into a two-dimensional projection to inspect the inner structure of the data. This result was confirmed by the analysis of the dendrograms, which show different associations of elements among detrital and epigenetic pyrites. Elements such as Cu, Zn, Ag, Sb, Te, Au, Pb, and Bi are mobilized during mineral alteration and was crystallized in newly formed minerals, such as chalcopyrite, pyrrhotite, and sphalerite, which are spatially associated with epigenetic pyrite and free gold. These findings could explain the differences in the mineral assemblage in portions of the deposits that prevail sedimentary minerals or the others that were strongly modified by later alterations. In conclusion, ML is recommended in the processing of mineral chemistry data because it helps to process data without discarding significant variables, and the method allows to evidence the multivariate structure of data.

Data wrangling

Opening libraries

```
library(tidyverse) # ggplot2, tidyr, dplyr
library(readxl) # open XLSX data
library(geoquimica) # Data wrangling
library(umap) # Dimensionality reduction
library(pheatmap) # Distance matrices
library(dendextend) # Dendrograms
library(ggpubr) # Plot adjusts
```

Data preparation

```
set.seed(0)

df <- data.table::fread("~/GitHub/jacobina/data/minchem/piritas_jacobina_editada_v2.csv")
```

```r
# Defining variable class
df[,12:142] <- lapply(X = df[,12:142],FUN = as.double)

# Creating variable of imputation control
df1 <- df %>%
  drop_na(`Pyrite Type`) %>%
  mutate(impute_ni60 = ifelse(test = is.na(Ni60),yes = 'True',no = 'False'),
         impute_co = ifelse(test = is.na(Co59),yes = 'True',no = 'False'),
         impute_ti = ifelse(test = is.na(Ti49),yes = 'True',no = 'False'),
         impute_v = ifelse(test = is.na(V51),yes = 'True',no = 'False'))


index <- df1 %>%
  select(`Source file`:`Reef`, impute_ni60:impute_v)

statistics <- df1 %>%
  select(-names(index)) %>%
  select(matches('LOD$|2SE$'))

lod <- df1 %>%
  dplyr::select(matches('LOD$'))

elems <- df1 %>%
  dplyr::select(-names(index),-V1)
```

Data selection

```r
geoquimica::elem_fillrate( data.table::fread(
  "~/GitHub/jacobina/data/minchem/piritas_jacobina_editada_v2.csv",
  verbose = FALSE) %>%
    mutate_at(.vars = 12:143,.funs = as.double) %>%
    drop_na(`Pyrite Type`) %>% # Drop wrong analysis
    select(-(V1:`Source file`),-(DateTime:Comments)) %>%
    janitor::clean_names()
  ) %>%
  filter(!str_detect(string = Column.Name, pattern = 'lod$|2se$'),
         !Column.Name %in% c('datetime','generation','pyrite_type','texture','reef')) %>%
  arrange(Fill.Rate) %>%
  mutate(Column.Name = fct_inorder(Column.Name)) %>%
  # mutate()
  ggplot(aes(x = Column.Name, y = Fill.Rate)) +
  geom_col(aes(fill = ifelse(test = Fill.Rate < 50,'Non-selected','Selected')), col = 'gray') +
  geom_hline(yintercept = 50, lty = 5, col = 'red', size = .7) +
  scale_y_continuous(breaks = seq(0,100,10))  +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90,vjust = 0.5, hjust = 1),
        legend.position = c(.15,.8)
        ) +
  labs(x = 'Elements',
       y = '% of non-missing values',
       fill = '')
```

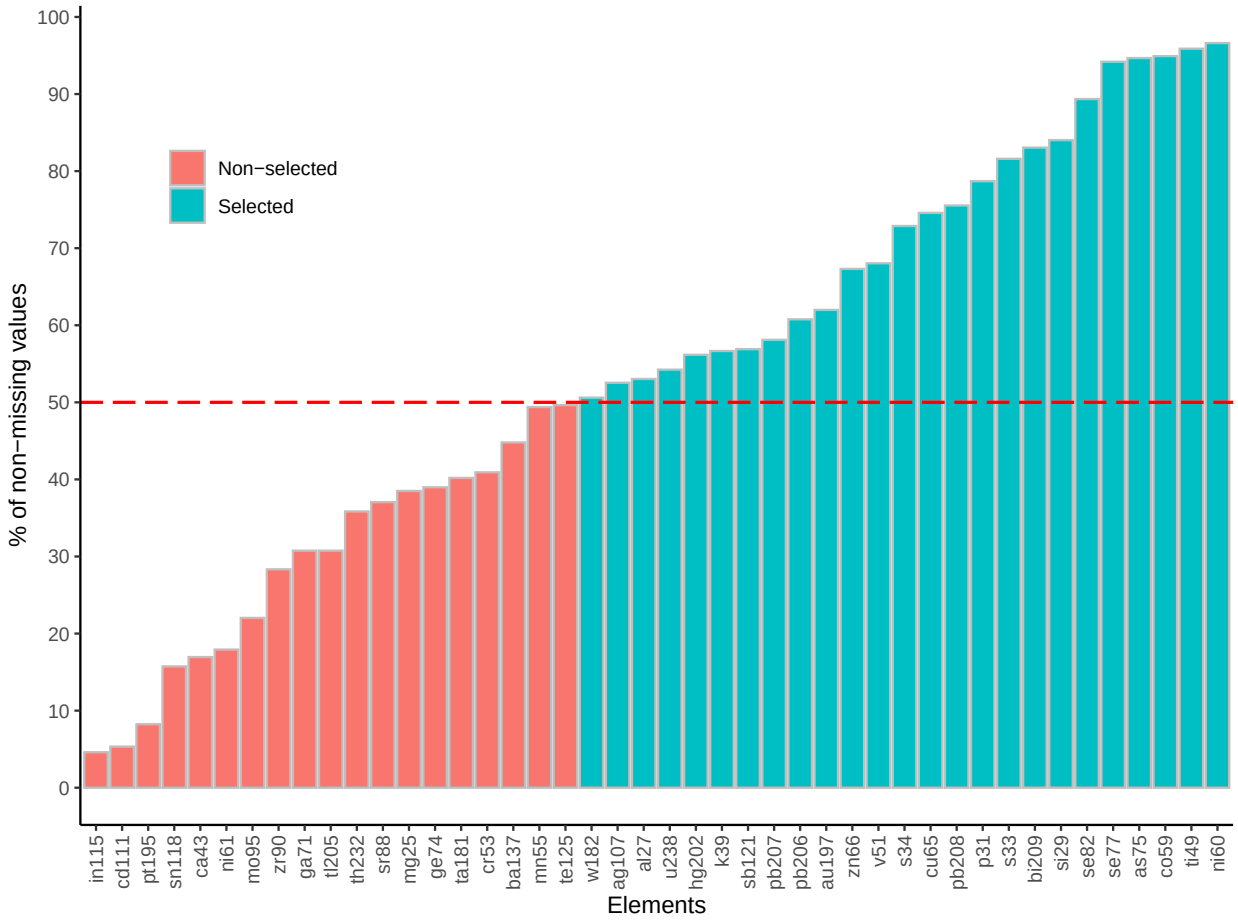


Fig.1: list of elements ordered by the percentage of non-missing data. The 50% threshold (horizontal dashed line) was used to determine if a variable could be selected for multivariate analysis. The elements Al, P, and Si were not selected based on the small variability in the dataset.

Imputation

LDL and Missing Value Imputation

```
# Imputation of LDL elements
half_ldl <- elems %>%
  mutate(Al27 = ifelse(test = is.na(Al27),yes = `Al27 LOD`/sqrt(2),no = Al27),
         Si29 = ifelse(test = is.na(Si29),yes = `Si29 LOD`/sqrt(2),no = Si29),
         P31 = ifelse(test = is.na(P31),yes = `P31 LOD`/sqrt(2),no = P31),
         S33 = ifelse(test = is.na(S33),yes = `S33 LOD`/sqrt(2),no = S33),
         S34 = ifelse(test = is.na(S34),yes = `S34 LOD`/sqrt(2),no = S34),
         K39 = ifelse(test = is.na(K39),yes = `K39 LOD`/sqrt(2),no = K39),
         Ti49 = ifelse(test = is.na(Ti49),yes = `Ti49 LOD`/sqrt(2),no = Ti49),
         V51 = ifelse(test = is.na(V51),yes = `V51 LOD`/sqrt(2),no = V51),
         Co59 = ifelse(test = is.na(Co59),yes = `Co59 LOD`/sqrt(2),no = Co59),
         Ni60 = ifelse(test = is.na(Ni60),yes = `Ni60 LOD`/sqrt(2),no = Ni60),
         Cu65 = ifelse(test = is.na(Cu65),yes = `Cu65 LOD`/sqrt(2),no = Cu65),
```

```
         Zn66 = ifelse(test = is.na(Zn66),yes = `Zn66 LOD`/sqrt(2),no = Zn66),
         As75 = ifelse(test = is.na(As75),yes = `As75 LOD`/sqrt(2),no = As75),
         Se77 = ifelse(test = is.na(Se77),yes = `Se77 LOD`/sqrt(2),no = Se77),
         Se82 = ifelse(test = is.na(Se82),yes = `Se82 LOD`/sqrt(2),no = Se82),
         Ag107 = ifelse(test = is.na(Ag107),yes = `Ag107 LOD`/sqrt(2),no = Ag107),
         Sb121 = ifelse(test = is.na(Sb121),yes = `Sb121 LOD`/sqrt(2),no = Sb121),
         W182 = ifelse(test = is.na(W182),yes = `W182 LOD`/sqrt(2),no = W182),
         Au197 = ifelse(test = is.na(Au197),yes = `Au197 LOD`/sqrt(2),no = Au197),
         Hg202 = ifelse(test = is.na(Hg202),yes = `Hg202 LOD`/sqrt(2),no = Hg202),
         Pb206 = ifelse(test = is.na(Pb206),yes = `Pb206 LOD`/sqrt(2),no = Pb206),
         Pb207 = ifelse(test = is.na(Pb207),yes = `Pb207 LOD`/sqrt(2),no = Pb207),
         Pb208 = ifelse(test = is.na(Pb208),yes = `Pb208 LOD`/sqrt(2),no = Pb208),
         Bi209 = ifelse(test = is.na(Bi209),yes = `Bi209 LOD`/sqrt(2),no = Bi209),
         U238 = ifelse(test = is.na(U238),yes = `U238 LOD`/sqrt(2),no = U238)) %>%
  mutate(impute_pb206 = ifelse(test = is.na(Pb206),yes = 'True',no = 'False'),
         impute_pb207 = ifelse(test = is.na(Pb207),yes = 'True',no = 'False'),
         impute_s33 = ifelse(test = is.na(S33),yes = 'True',no = 'False'),
         impute_s34 = ifelse(test = is.na(S34),yes = 'True',no = 'False')) %>%
  select(-matches('LOD$|2SE$')) %>%
  geoquimica::elem_select(cut = .5)

# Imputation of missing values based on a multivariate non-parametric regression
imputed <- missRanger::missRanger(data = half_ldl,
                                  pmm.k = 3,
                                  maxiter = 10,
                                  seed = 0,
                                  verbose = 2)
```

```
##
## Missing value imputation by random forests
##
##   Variables to impute:       P31, S33, S34, K39, Pb206, Pb207
##   Variables used to impute:  Al27, Si29, P31, S33, S34, K39, Ti49, V51, Co59, Ni60, Cu65, Zn66, As75
##  P31 S33 K39 Pb206   Pb207   S34
## iter 1:  0.9855  0.4544  0.5600  0.5095  0.2164  0.1844
## iter 2:  0.5228  0.0639  0.2971  0.3116  0.1766  0.0708
## iter 3:  0.5385  0.0658  0.3022  0.3387  0.1589  0.0599
```

Data Recode

```
# Recoding variables Generation, Reef, Reef_label and Unit
df2 <- index %>%
  bind_cols(imputed) %>%
  mutate(Unit = case_when(Reef == 'Basal Reef' ~ 'Serra do Córrego',
                          Reef == 'Main Reef' ~ 'Serra do Córrego',
                          Reef == 'SPC' ~ 'Serra do Córrego',
                          Reef == 'LU' ~ 'Serra do Córrego',
                          Reef == 'LVLPC' ~ 'Serra do Córrego',
                          Reef == 'MSPC' ~ 'Serra do Córrego',
                          Reef == 'MPC' ~ 'Serra do Córrego',
                          Reef == 'SPC' ~ 'Serra do Córrego',
```

```
                          Reef == 'MU' ~ 'Serra do Córrego',
                          Reef == 'Holandez' ~ 'Serra do Córrego',
                          Reef == 'Maneira' ~ 'Serra do Córrego',
                          Reef == 'ITV' ~ 'Intrusive',
                          Reef == 'UMF' ~ 'Intrusive',
                          Reef == 'IQL' ~ 'Serra do Córrego',
                          Reef == 'Basement' ~ 'Basement',
                          Reef == 'CAF' ~ 'Cruz das Almas'),
       Reef_label = case_when(Reef == 'Basal Reef' ~ 'LC',
                              Reef == 'Main Reef' ~ 'LC',
                              Reef == 'SPC' ~ 'UC1',
                              Reef == 'LU' ~ 'UC1',
                              Reef == 'LVLPC' ~ 'UC1',
                              Reef == 'MSPC' ~ 'UC1',
                              Reef == 'MPC' ~ 'UC1',
                              Reef == 'SPC' ~ 'UC1',
                              Reef == 'MU' ~ 'UC1',
                              Reef == 'Holandez' ~ 'UC2',
                              Reef == 'Maneira' ~ 'UC2',
                              Reef == 'ITV' ~ 'Intr.',
                              Reef == 'UMF' ~ 'Intr.',
                              Reef == 'IQL' ~ 'IQL',
                              Reef == 'Basement' ~ 'Base.',
                              Reef == 'CAF' ~ 'CdA'),
       Reef = case_when(Reef == 'Basal Reef' ~ 'Lower Conglomerate',
                        Reef == 'Main Reef' ~ 'Lower Conglomerate',
                        Reef == 'SPC' ~ 'Upper Conglomerate 1',
                        Reef == 'LU' ~ 'Upper Conglomerate 1',
                        Reef == 'LVLPC' ~ 'Upper Conglomerate 1',
                        Reef == 'MSPC' ~ 'Upper Conglomerate 1',
                        Reef == 'MPC' ~ 'Upper Conglomerate 1',
                        Reef == 'SPC' ~ 'Upper Conglomerate 1',
                        Reef == 'MU' ~ 'Upper Conglomerate 1',
                        Reef == 'Holandez' ~ 'Upper Conglomerate 2',
                        Reef == 'Maneira' ~ 'Upper Conglomerate 2',
                        Reef == 'ITV' ~ 'Intrusive',
                        Reef == 'UMF' ~ 'Intrusive',
                        Reef == 'IQL' ~ 'Intermediate Quartzite',
                        Reef == 'Basement' ~ 'Basement',
                        Reef == 'CAF' ~ 'Cruz das Almas'),
) %>%
dplyr::select(-c(`Source file`:Comments, P31,Si29,Al27)) %>%
mutate(Generation = case_when(Reef == 'Intrusive' ~ 'Intrusive',
                              Reef == 'Basement' ~ 'Basement',
                              TRUE ~ as.character(Generation)),
       Texture = case_when(Reef == 'Intrusive' | Reef == 'Basement' ~ 'Subhedral',
                           TRUE ~ as.character(Texture)))
```

Dimensionality Reduction

Uniform Manifold Approximation and Projection - UMAP

```
# UMAP processing

dfumap <-
  df2 %>%
  select_if(is.numeric) %>%
  geoquimica::elem_norm(method = 'clr') %>%
  umap::umap()

# Data merging
df3 <-
  df2 %>%
  bind_cols(as_tibble(dfumap$layout))

# Making Plot a)
umap1 <-
  df3 %>%
  ggplot(aes(x = V1, y = V2,
             fill = Generation,
             shape = Generation,
             group = Generation)) +
  geom_vline(xintercept = 0, col = 'grey', size = .7) +
  geom_hline(yintercept = 0, col = 'grey', size = .7) +
  geom_point(inherit.aes = FALSE,
             data = df3 %>%
               group_by(Texture) %>%
               summarize(U1mean = mean(V1),
                         U2mean = mean(V2)),
             mapping = aes(x = U1mean, y = U2mean),
             cex = 3, shape = 3, col = 'black') +
  ggforce::geom_ellipse(inherit.aes = FALSE,
                        data = df3 %>%
                          group_by(Texture) %>%
                          summarize(U1mean = mean(V1),
                                    U1sd = sd(V1),
                                    U2mean = mean(V2),
                                    U2sd = sd(V2)),
                        mapping = aes(x0 = U1mean, y0 = U2mean, a = 2*U1sd, b = 2*U2sd, angle = 0),
                        fill = 'grey', alpha = .05) +
  geom_point(alpha = .6, col = 'black') +
  scale_shape_manual(values = c(22,23,24, 21)) +
  facet_wrap(. ~ Texture,nrow = 2) +
  labs(x = 'UMAP1', y = 'UMAP2') +
  theme_bw()

# Making plot b)
umap2 <-
  df3 %>%
  ggplot(aes(x = V1, y = V2,
             fill = Generation,
```

```
            shape = Generation,
            group = Generation)) +
  geom_vline(xintercept = 0, col = 'grey', size = .7) +
  geom_hline(yintercept = 0, col = 'grey', size = .7) +
  geom_point(inherit.aes = FALSE,
            data = df3 %>%
              group_by(Reef) %>%
              summarize(U1mean = mean(V1),
                        U2mean = mean(V2)),
            mapping = aes(x = U1mean, y = U2mean),
            cex = 3, shape = 3, col = 'black') +
  ggforce::geom_ellipse(inherit.aes = FALSE,
                        data = df3 %>%
                          group_by(Reef) %>%
                          summarize(U1mean = mean(V1),
                                    U1sd = sd(V1),
                                    U2mean = mean(V2),
                                    U2sd = sd(V2)),
                        mapping = aes(x0 = U1mean, y0 = U2mean, a = 2*U1sd, b = 2*U2sd, angle = 0),
                        fill = 'grey', alpha = .05) +
  geom_point(alpha = .6, col = 'black') +
  scale_shape_manual(values = c(22,23,24, 21,
                                22,23,24, 21)) +
  facet_wrap(. ~ Reef,ncol = 3) +
  labs(x = 'UMAP1', y = 'UMAP2') +
  theme_bw() #+

# Arrange plots
ggarrange(umap1, umap2,nrow = 2,
          labels = c('a)','b)'),
          align = 'hv',
          heights = c(2,3),font.label = list(size = 16, face = 'bold'))
```

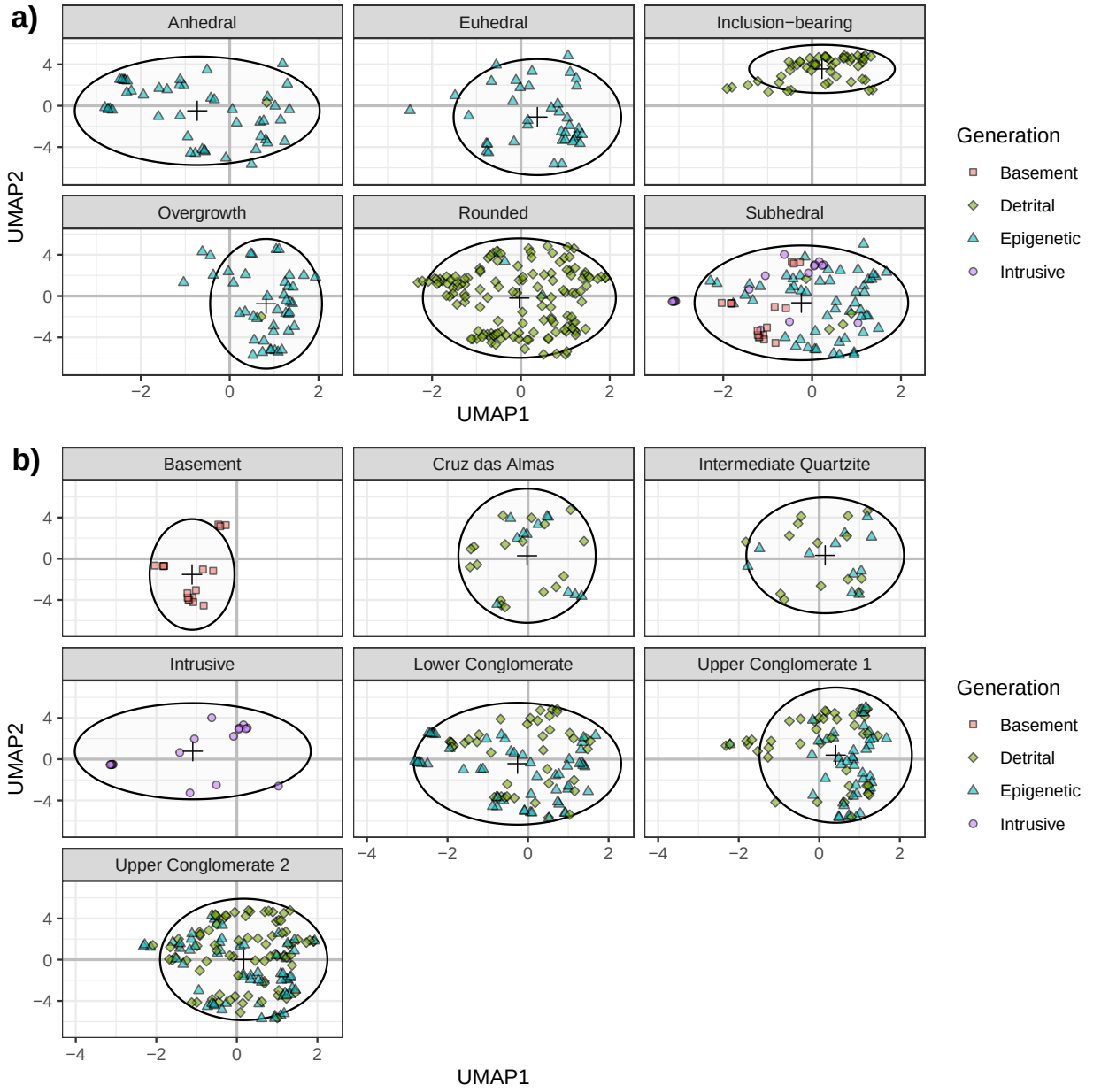


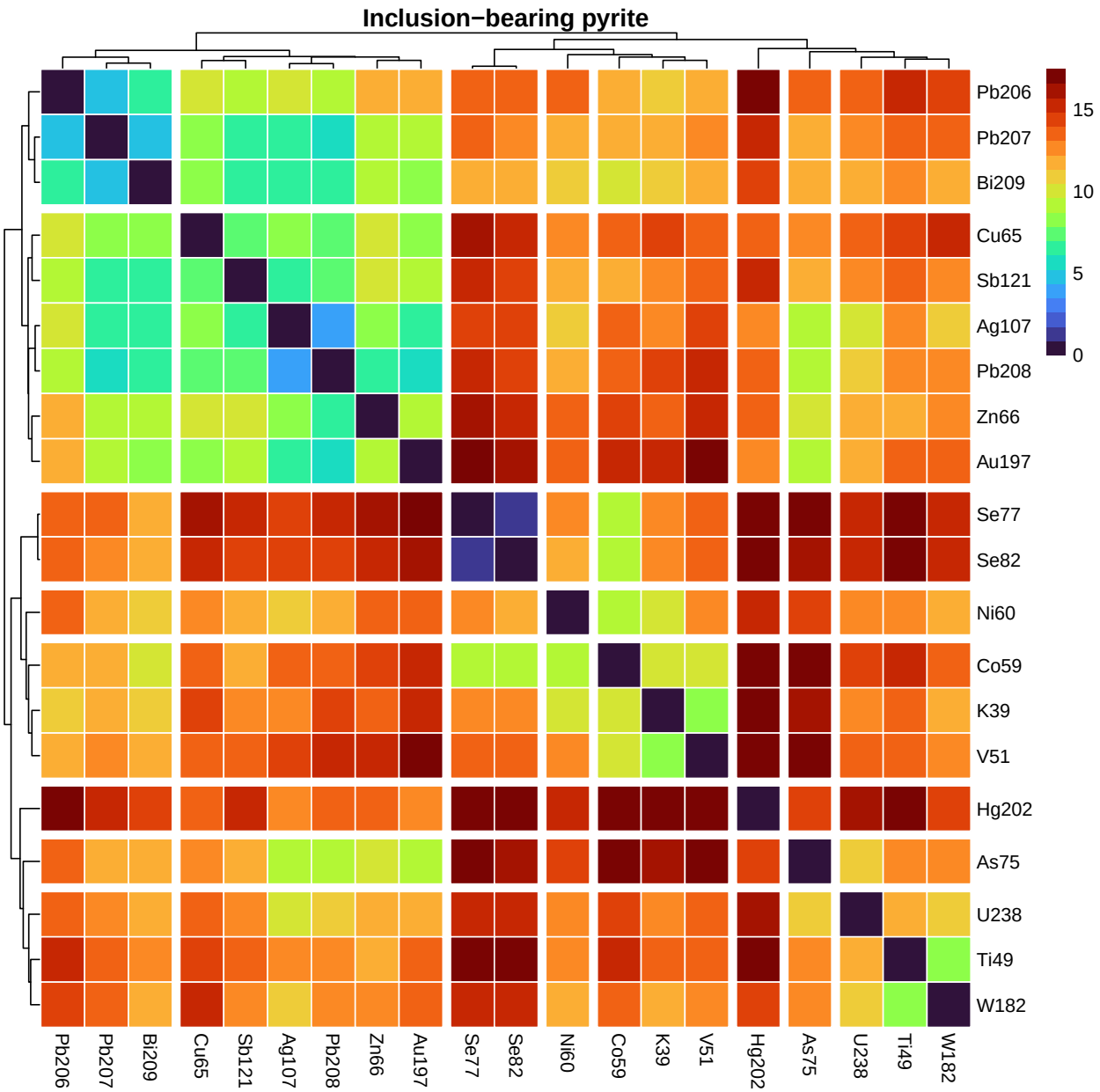
Fig.2: UMAP configuration for data with samples classified according to a) pyrite texture and b) stratigraphic level. The ellipses drawn in each box are centered based on the mean of the samples, and its axes are calculated according to two times the standard deviation for each UMAP coordinate.

Distance matrices

```
d <-
  df2 %>%
  filter(Texture == 'Inclusion-bearing') %>%
  geoquimica::elem_norm(method = 'clr') %>%
  geoquimica::elem_norm() %>%
  select(K39:U238) %>%
```

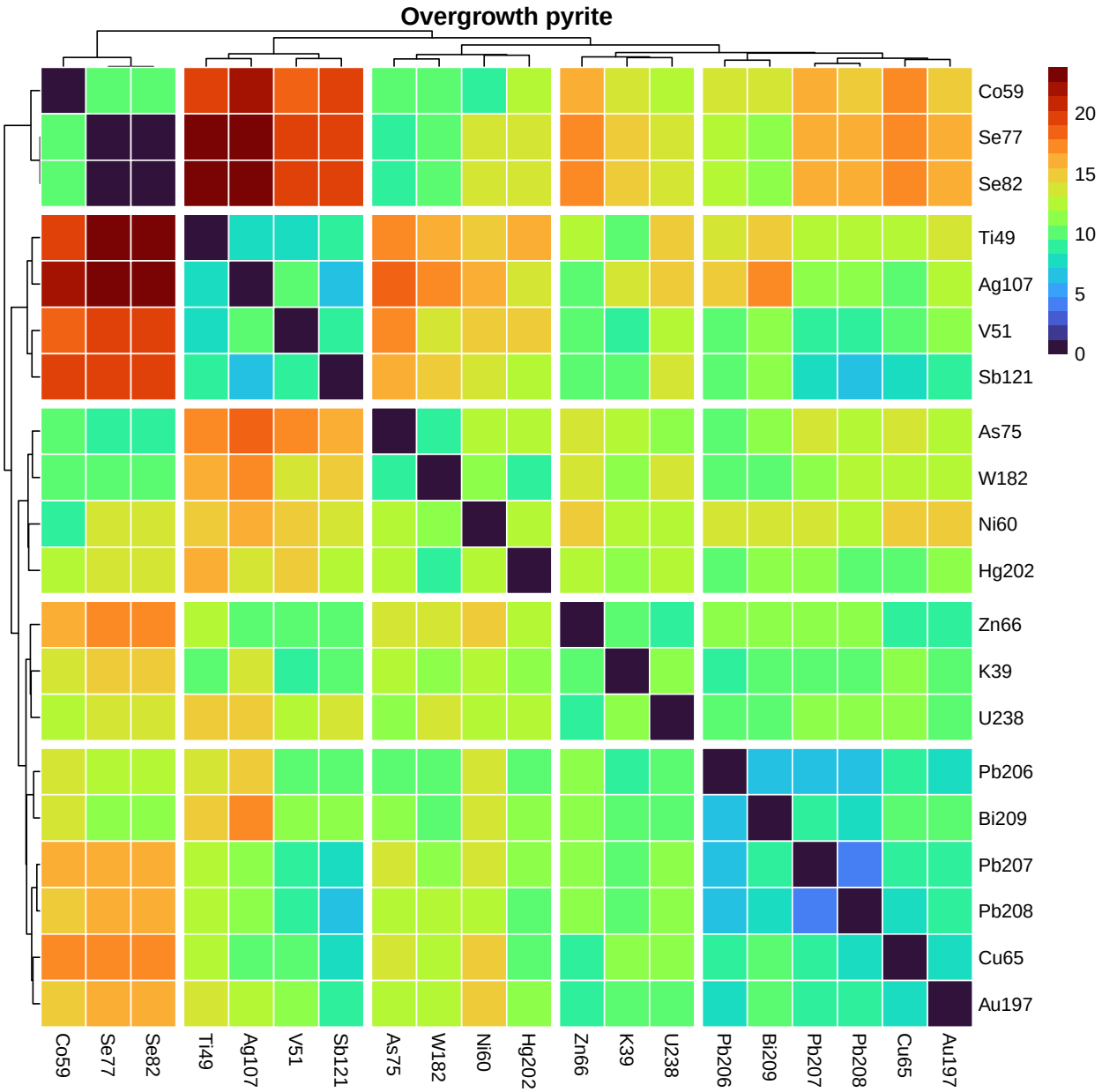
```
  t()

# Making Fig. A, Inclusion-bearing Pyrite
inbear <-
  ggplotify::as.ggplot(pheatmap(mat = df2 %>%
              filter(Texture == 'Inclusion-bearing') %>%
              geoquimica::elem_norm(method = 'clr') %>%
              geoquimica::elem_norm() %>%
              select(K39:U238) %>%
              t() %>%
              dist(method = 'manhattan'),
            labels_row = rownames(d),
            labels_col = rownames(d),
            border_color = 'white',
            clustering_distance_rows = 'manhattan',
            clustering_distance_cols = 'manhattan',
            color = pals::turbo(20),
            cutree_rows = 8,
            cutree_cols = 8,
            main = "Inclusion-bearing pyrite",
            treeheight_row = 15,
            treeheight_col = 15,))
```



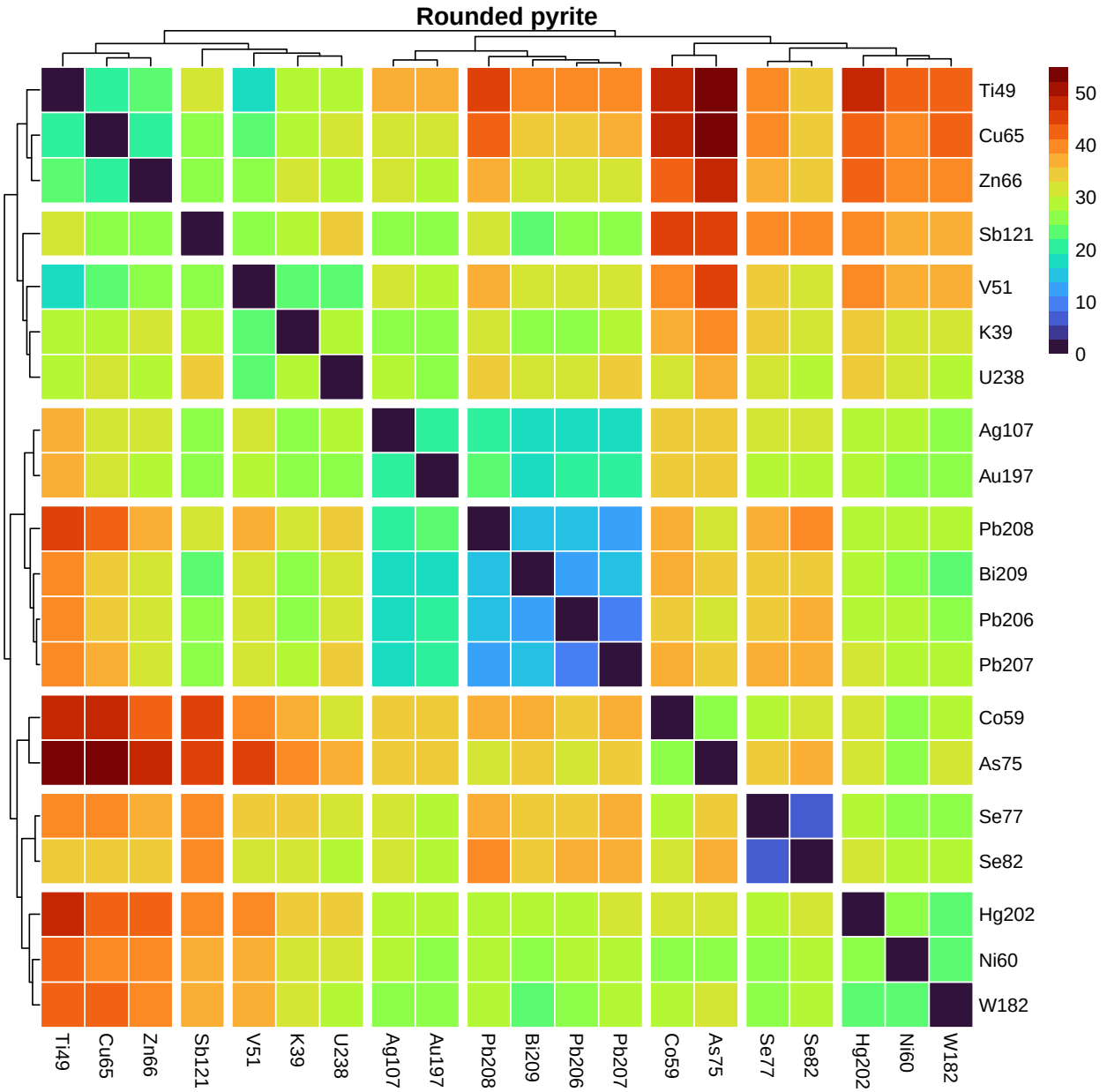

```
# Making Fig. B, Overgrowth Pyrite
overg <- ggplotify::as.ggplot(pheatmap(mat = df2 %>%
              filter(Texture == 'Overgrowth') %>%
              geoquimica::elem_norm(method = 'clr') %>%
              geoquimica::elem_norm() %>%
              select(K39:U238) %>%
              t() %>%
              dist(method = 'manhattan'),
            labels_row = rownames(d),
            border_color = 'white',
            labels_col = rownames(d),
            color = pals::turbo(20),
            clustering_distance_rows = 'manhattan',
            clustering_distance_cols = 'manhattan',
```

```
            cutree_rows = 5,
            cutree_cols = 5,
            main = "Overgrowth pyrite",
            treeheight_row = 15,
            treeheight_col = 15))
```

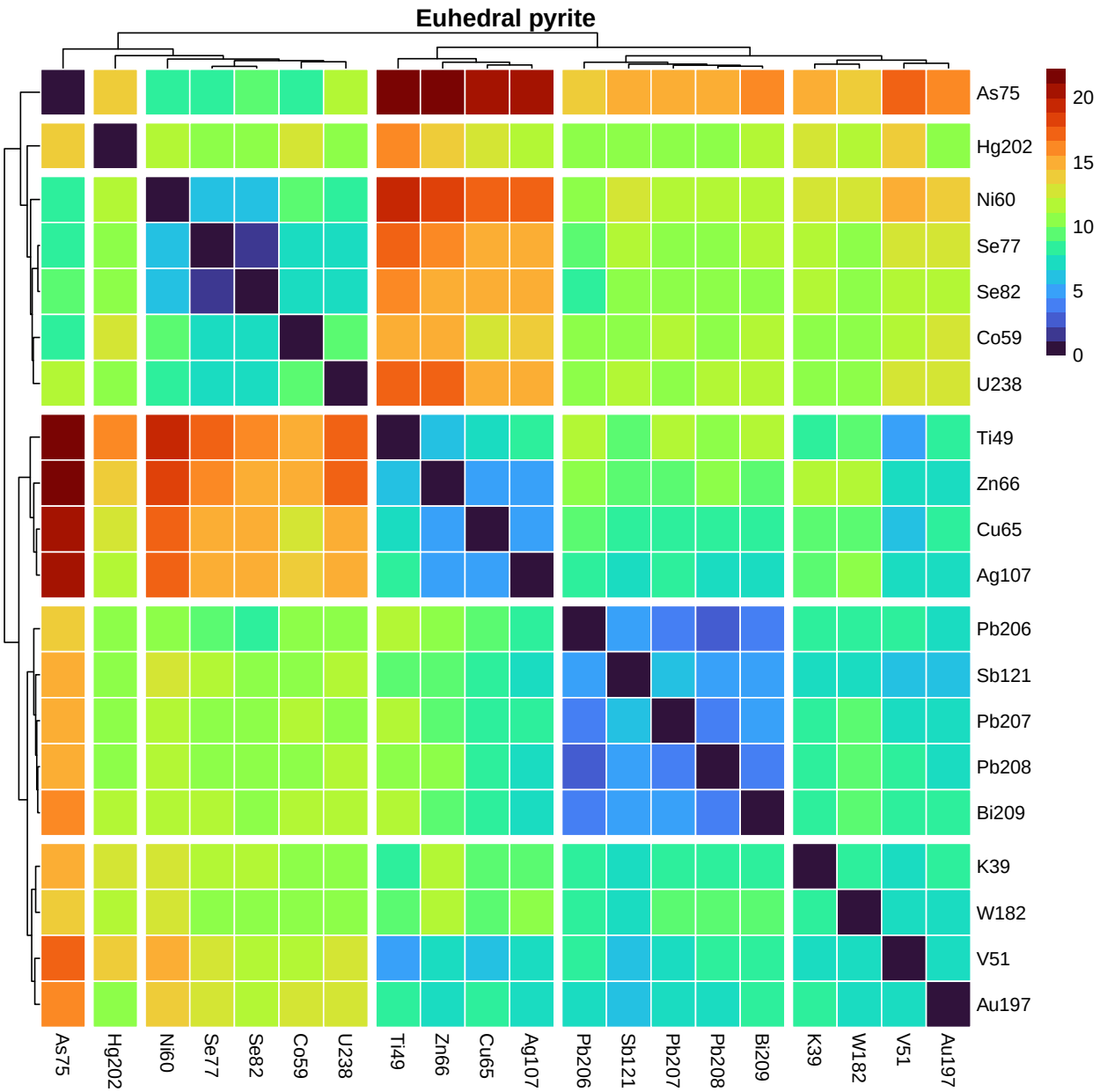



```
# Making Fig. C, Rounded Pyrite
rounded <- ggplotify::as.ggplot(pheatmap(mat = df2 %>%
              filter(Texture == 'Rounded') %>%
              geoquimica::elem_norm(method = 'clr') %>%
              geoquimica::elem_norm() %>%
              select(K39:U238) %>%
              t() %>%
```

```
  dist(method = 'manhattan'),
labels_row = rownames(d),
labels_col = rownames(d),
border_color = 'white',
clustering_distance_rows = 'manhattan',
clustering_distance_cols = 'manhattan',
color = pals::turbo(20),
cutree_rows = 8,
cutree_cols = 8,
main = "Rounded pyrite",
treeheight_row = 15,
treeheight_col = 15))
```



```r
# Making Fig. D, Euhedral Pyrite
euhedral <- ggplotify::as.ggplot(pheatmap(mat = df2 %>%
             filter(Texture == 'Euhedral') %>%
             geoquimica::elem_norm(method = 'clr') %>%
             geoquimica::elem_norm() %>%
             select(K39:U238) %>%
             t() %>%
             dist(method = 'manhattan'),
           labels_row = rownames(d),
           labels_col = rownames(d),
           border_color = 'white',
           clustering_distance_rows = 'manhattan',
           clustering_distance_cols = 'manhattan',
           color = pals::turbo(20),
           cutree_rows = 6,
           cutree_cols = 6,
           main = "Euhedral pyrite",
           treeheight_row = 15,
           treeheight_col = 15))
```




```
# Arrange plots
ggarrange(inbear,overg,rounded,euhedral,
          ncol = 2,nrow = 2,
          labels = c('a)','b)','c)','d)'),align = 'hv',hjust = 0)
```

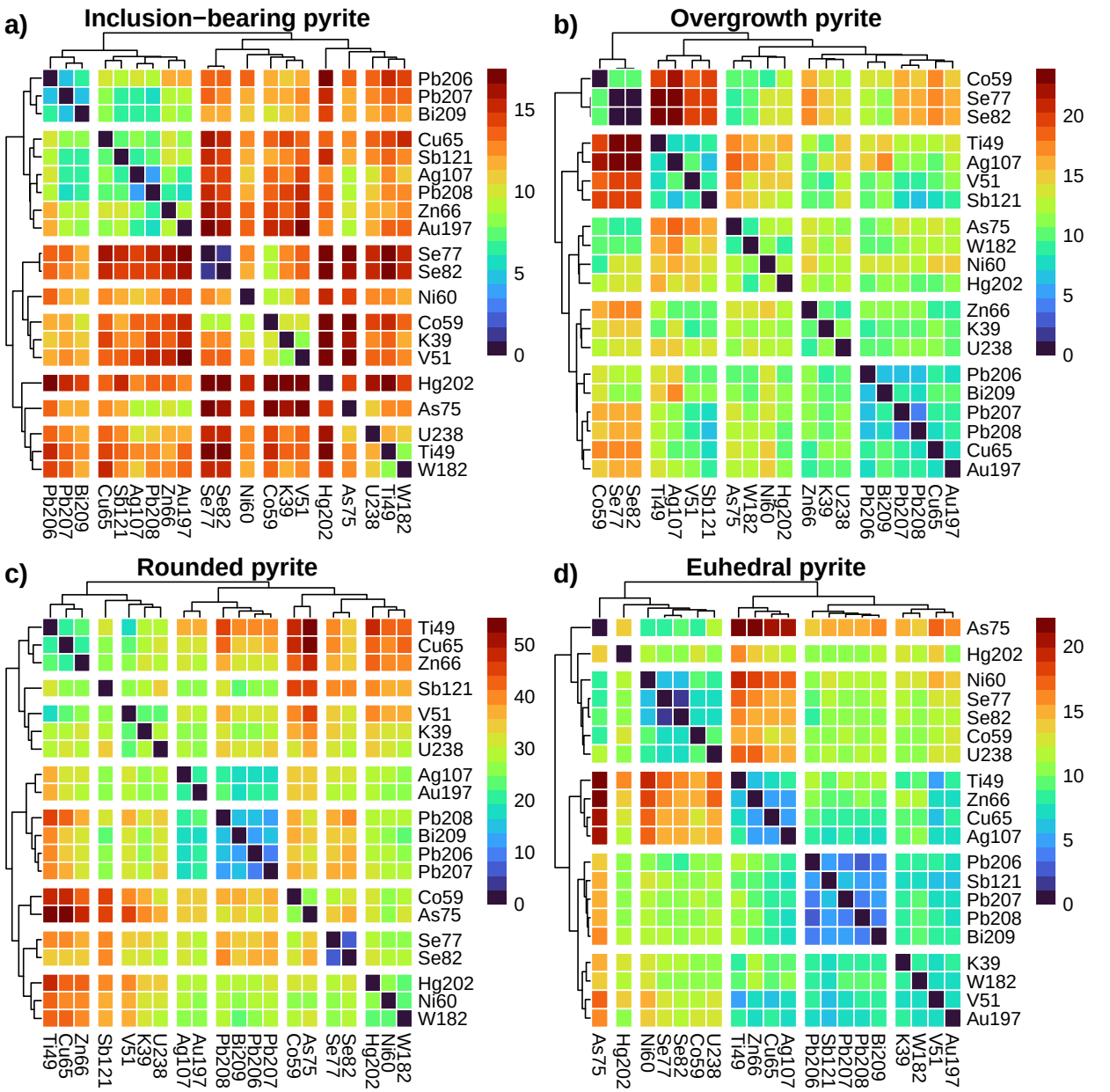


Fig.3: Dendrograms and distance matrices for pyrite grains according to grain texture: a) Inclusion-bearing, b) Overgrowth, c) Anhedral, and d) Euhedral. Rows and columns are ordered according to the agglomerative clustering, calculated by the Manhattan distance.

Compared dendrograms

```
d1 <-
  df3 %>%
  filter(Texture == 'Inclusion-bearing') %>%
  select(K39:U238) %>%
  geoquimica::elem_norm(method = 'clr') %>%
  t() %>%
```

```
  dist(method = 'manhattan') %>%
  hclust(method = 'average') %>%
  as.dendrogram()

d2 <-
  df3 %>%
  filter(Texture == 'Overgrowth') %>%
  select(K39:U238) %>%
  geoquimica::elem_norm(method = 'clr') %>%
  t() %>%
  dist(method = 'manhattan') %>%
  hclust(method = 'average') %>%
  as.dendrogram()


# Custom these kendo, and place them in a list
dl1 <-
  dendextend::dendlist(
    d1 %>%
      set("branches_lty", 1) %>%
      set("labels_col",
          h = 40) %>%
      set("branches_lty", 1) %>%
      set("branches_k_color",
          h = 40),
    d2 %>%
      set("branches_lty", 1) %>%
      set("labels_col",
          h = 60) %>%
      set("branches_lty", 1) %>%
      set("branches_k_color",
          h = 60)
  )

# Plot them together
dendextend::tanglegram(dl1,
             common_subtrees_color_lines = FALSE,
             highlight_distinct_edges  = FALSE,
             highlight_branches_lwd = FALSE,
             margin_inner=4, intersecting = TRUE,
             lwd=1, k_labels = NULL, k_branches = NULL)
```

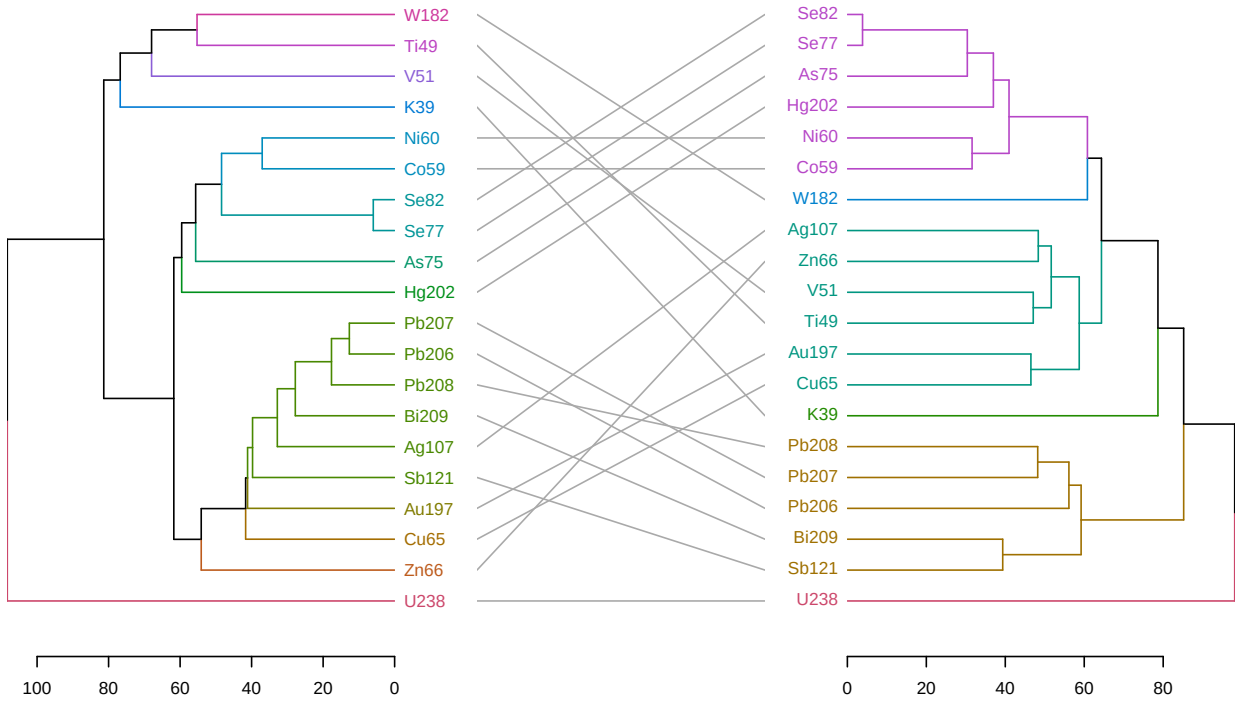


Fig.4: Linked dendrograms for Detrital (left) and Epigenetic pyrites (right) mapping the relation of elements.

Appendix A - Imputation of LDL and Missing values

```
# Function to make qq plots coded by categorical variables
make_qq <- function(dd, x) {
  dd<-dd[order(dd[[x]]), ]
  dd$qq <- qnorm(ppoints(nrow(dd)))
  dd
}

# Data without imputation
p_original <-
  df3 %>%
  filter(impute_v == 'False') %>%
  make_qq(dd = .,x = 'V51') %>%
  ggplot(aes(x = qq, y = log(V51))) +
  geom_qq_line(aes(sample = qq),
               lty = 1, col = 'gray', size = 1,
               alpha = .7) +
  geom_point(aes(col = impute_v,
                 shape = impute_v),
             # col = 'white',
             alpha = .3) +
  labs(x = 'Theoretical distribution',
       y = 'log(V51)',
```

```
       col = 'Imputed?',
       shape = 'Imputed?') +
  scale_shape_manual(values = c(16,17)) +
  theme_classic() +
  theme(legend.justification = 'center',
        legend.background = element_rect(
          fill = 'grey98'))

# Data with half of LDL imputed values
p_ldl <-
  df3 %>%
  make_qq(dd = .,x = 'V51') %>%
  ggplot(aes(x = qq, y = log(V51))) +
  geom_qq_line(aes(sample = qq),
               lty = 1, col = 'gray', size = 1,
               alpha = .7) +
  geom_point(aes(col = impute_v,
                 shape = impute_v),
             # col = 'white',
             alpha = .4) +
  labs(x = 'Theoretical distribution',
       y = 'log(V51)',
       col = 'Imputed?',
       shape = 'Imputed?') +
  scale_shape_manual(values = c(16,17)) +
  theme_classic() +
  theme(legend.justification = 'center',
        legend.background = element_rect(
          fill = 'grey98'))

# Random Forest Regression without categorical variables
p_rf1 <-
  df %>%
  drop_na(`Pyrite Type`) %>%
  janitor::clean_names() %>%
  select(-matches('lod$|2se$')) %>%
  select(-`pyrite_type`,-generation,-texture,-reef,-c(source_file:comments)) %>%
  missRanger::missRanger(data = .,
                         pmm.k = 5,
                         # formula = Ni61 ~ Ni60,
                         maxiter = 10,
                         seed = 321321,
                         verbose = 2,
                         num.trees = 1000) %>%
  bind_cols(df1$impute_ti,
            df1$impute_v,
            df1$impute_co,
            df1$impute_ni60) %>%
  rename(impute_ti = `...46`,
         impute_v = `...47`,
         impute_co = `...48`,
         impute_ni60 = `...49`) %>%
  make_qq(dd = .,x = 'v51') %>%
```

```
  ggplot(aes(x = qq, y = log(v51))) +
  geom_qq_line(aes(sample = qq),
               lty = 1, col = 'gray', size = 1,
               alpha = .7) +
  geom_point(aes(col = impute_v,
                 shape = impute_v),
             alpha = .4,
             # col = 'white'
  ) +
  labs(x = 'Theoretical distribution',
       y = 'log(V51)',
       col = 'Imputed?',
       shape = 'Imputed?') +
  scale_shape_manual(values = c(16,17)) +
  theme_classic() +
  theme(legend.justification = 'center',
        legend.background = element_rect(
          fill = 'grey98'))
```

```
##
## Missing value imputation by random forests
##
##   Variables to impute:       mg25, al27, si29, p31, s33, s34, k39, ca43, ti49, v51, cr53, mn55, co59
##   Variables used to impute:  v1, mg25, al27, si29, p31, s33, s34, k39, ca43, ti49, v51, cr53, mn55, c
##  ni60    ti49    co59    as75    se77    se82    si29    bi209   s33 p31 pb208   cu65    s34 v51 zn6
## iter 1:  1.0984  0.9778  0.9965  0.8076  0.8318  0.4159  0.9323  1.0150  0.4100  0.9744  0.8928  0.08
## iter 2:  0.3723  0.4508  0.4851  0.4009  0.3997  0.4136  0.6449  0.6256  0.1106  0.4962  0.6807  0.09
## iter 3:  0.3508  0.4495  0.4654  0.4452  0.4003  0.3987  0.6406  0.6372  0.1052  0.5104  0.6879  0.11
```

```
# Random Forests regression with caterogical variables
p_rf2 <-
  df %>%
  drop_na(`Pyrite Type`) %>%
  janitor::clean_names() %>%
  select(-matches('lod$|2se$')) %>%
  select(-`pyrite_type`,-c(source_file:comments)) %>%
  missRanger::missRanger(data = .,
                         pmm.k = 5,
                         maxiter = 10,
                         seed = 321321,
                         verbose = 2,
                         num.trees = 1000) %>%
  bind_cols(df1$impute_ti,
            df1$impute_v,
            df1$impute_co,
            df1$impute_ni60) %>%
  rename(impute_ti = `...49`,
         impute_v = `...50`,
         impute_co = `...51`,
         impute_ni60 = `...52`) %>%
  make_qq(dd = .,x = 'v51') %>%
  ggplot(aes(x = qq, y = log(v51))) +
  geom_qq_line(aes(sample = qq),
```

```
              lty = 1, col = 'gray', size = 1,
              alpha = .7) +
  geom_point(aes(col = impute_v,
                 shape = impute_v),
             alpha = .4,
             # col = 'white'
  ) +
  labs(x = 'Theoretical distribution',
       y = 'log(V51)',
       col = 'Imputed?',
       shape = 'Imputed?') +
  scale_shape_manual(values = c(16,17)) +
  theme_classic() +
  theme(legend.justification = 'center',
        legend.background = element_rect(
          fill = 'grey98'))
```

```
##
## Missing value imputation by random forests
##
##   Variables to impute:       mg25, al27, si29, p31, s33, s34, k39, ca43, ti49, v51, cr53, mn55, co59
##   Variables used to impute:  v1, texture, generation, reef, mg25, al27, si29, p31, s33, s34, k39, ca4
##  ni60    ti49    co59    as75    se77    se82    si29    bi209   s33 p31 pb208   cu65    s34 v51 zn60
## iter 1:  0.9735  0.9630  0.9643  0.7628  0.7654  0.4052  0.9534  1.0158  0.3920  0.9571  0.9016  0.11
## iter 2:  0.3507  0.4615  0.5201  0.4309  0.3673  0.3720  0.6420  0.6604  0.1075  0.5100  0.6912  0.08
## iter 3:  0.3873  0.4338  0.5158  0.4469  0.3552  0.3658  0.6578  0.6430  0.1054  0.5194  0.7051  0.08
## iter 4:  0.3945  0.4520  0.4900  0.4283  0.3523  0.3624  0.6280  0.6349  0.0998  0.4927  0.6870  0.09
## iter 5:  0.3826  0.4671  0.4934  0.4429  0.3308  0.3410  0.6585  0.6543  0.0978  0.5082  0.7082  0.09
```

```
# Adjusting the plots
ggpubr::ggarrange(p_ldl, p_rf1, p_rf2,p_original,
                  ncol = 2,nrow = 2,align = 'hv',
                  labels = c('a)','b)','c)','d)'),
                  common.legend = TRUE,legend = 'right')
```

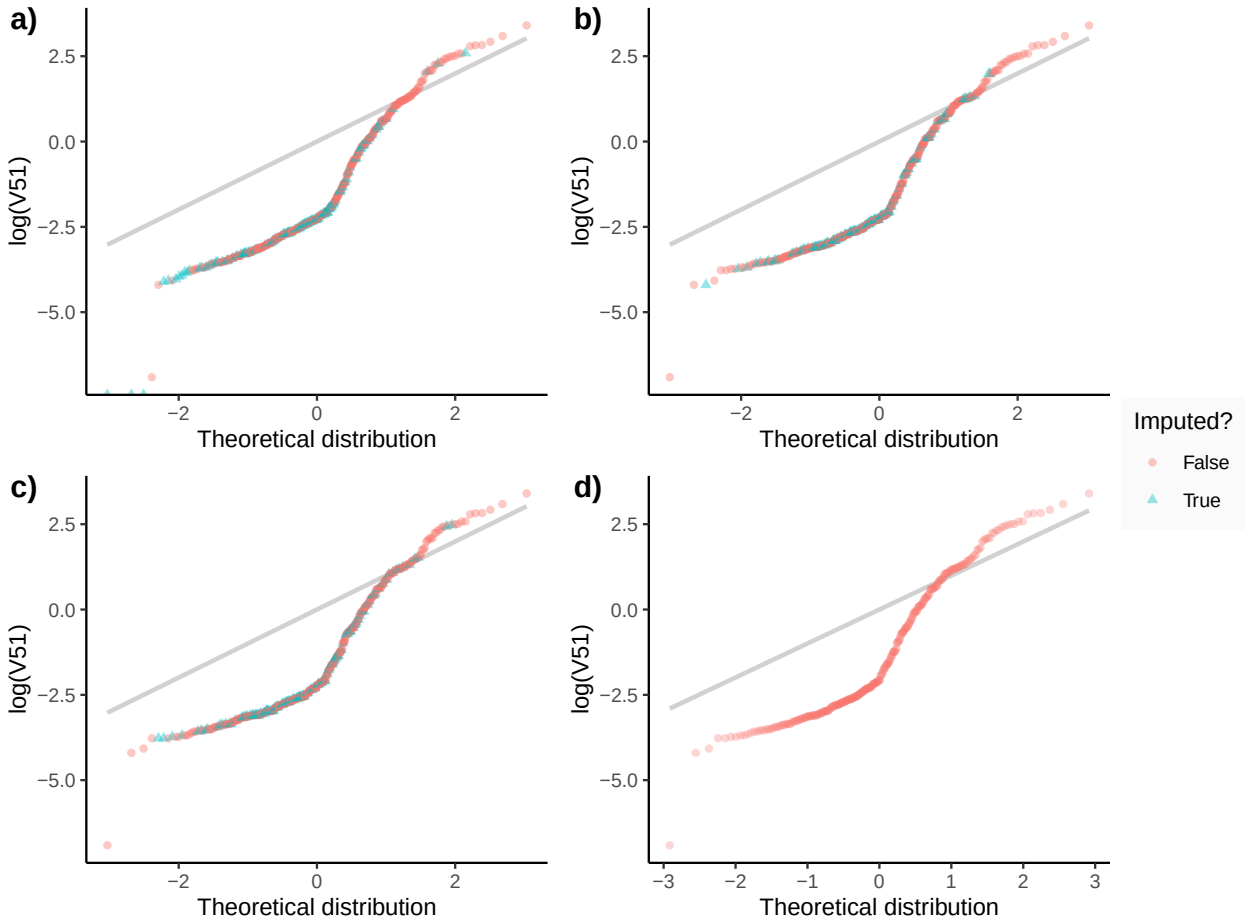


Fig.5: Quantile-plot for the concentration of V51 on a logarithmic scale according to different distributions: a) imputation based on a fraction of the detection limit, b) Random Forests imputation based only in quantitative variables, c) Random Forests imputation based on quantitative and categorical variables (e.g., grain texture) and d) original distribution (without imputed values).