Alignment Evaluation Documentation: Code accompaniment to the report entitled 'Visualization, Quantification and Alignment of Spectral Drift in Population Scale Untargeted Metabolomics Data'

Mir Henglin et al.

December 23, 2016

citation: TBA

```
# Data manipulation
library(magrittr)
library(purrr)
library(dplyr)
library(tidyr)

# Plotting
library(ggplot2)

# Shape-based clustering
library(dtwclust)
```

We create some sample data here. While we will be using real data (loaded in the background) for the rest of this documentation, the data generated below provides an example of the structure of our data.

```
plate <-
  1:10

well <-
  1:10

metaboliteID <-
  runif(10, 100, 900)

while (length(unique(metaboliteID)) != length(metaboliteID)) {
  metaboliteID <-
    runif(10, 100, 900)
}

dat <-
  expand.grid(plate, well, metaboliteID) %>%
  setNames(c('plate', 'well', 'metaboliteID'))


dat %<>%
  mutate(plateWellID = paste0(plate, '_', well),
         value = runif(n(), 1000, 1000000)) %>%
```

```
  tbl_df()

dat
```

```
## # A tibble: 1,000 × 5
##    plate  well metaboliteID plateWellID     value
##    <int> <int>        <dbl>       <chr>     <dbl>
## 1      1     1     136.9584         1_1 143762.39
## 2      2     1     136.9584         2_1  24007.76
## 3      3     1     136.9584         3_1 332234.40
## 4      4     1     136.9584         4_1 610857.48
## 5      5     1     136.9584         5_1 166541.97
## 6      6     1     136.9584         6_1 521470.79
## 7      7     1     136.9584         7_1 828335.42
## 8      8     1     136.9584         8_1 294054.12
## 9      9     1     136.9584         9_1 844857.60
## 10    10     1     136.9584        10_1 289540.63
## # ... with 990 more rows
```

For our sample data, the columns are:

- metaboliteID
  - numeric ID representing a unique measured metabolite.
- plateWellID
  - numeric ID representing a sample in which metabolites were measured.
- plate, well
  - the plate and well that a sample was measured in.
- value
  - The peak abundance measured in a sample.

Once again, this is data generated only to give an idea of what our data looks like. It is not used beyond this point.

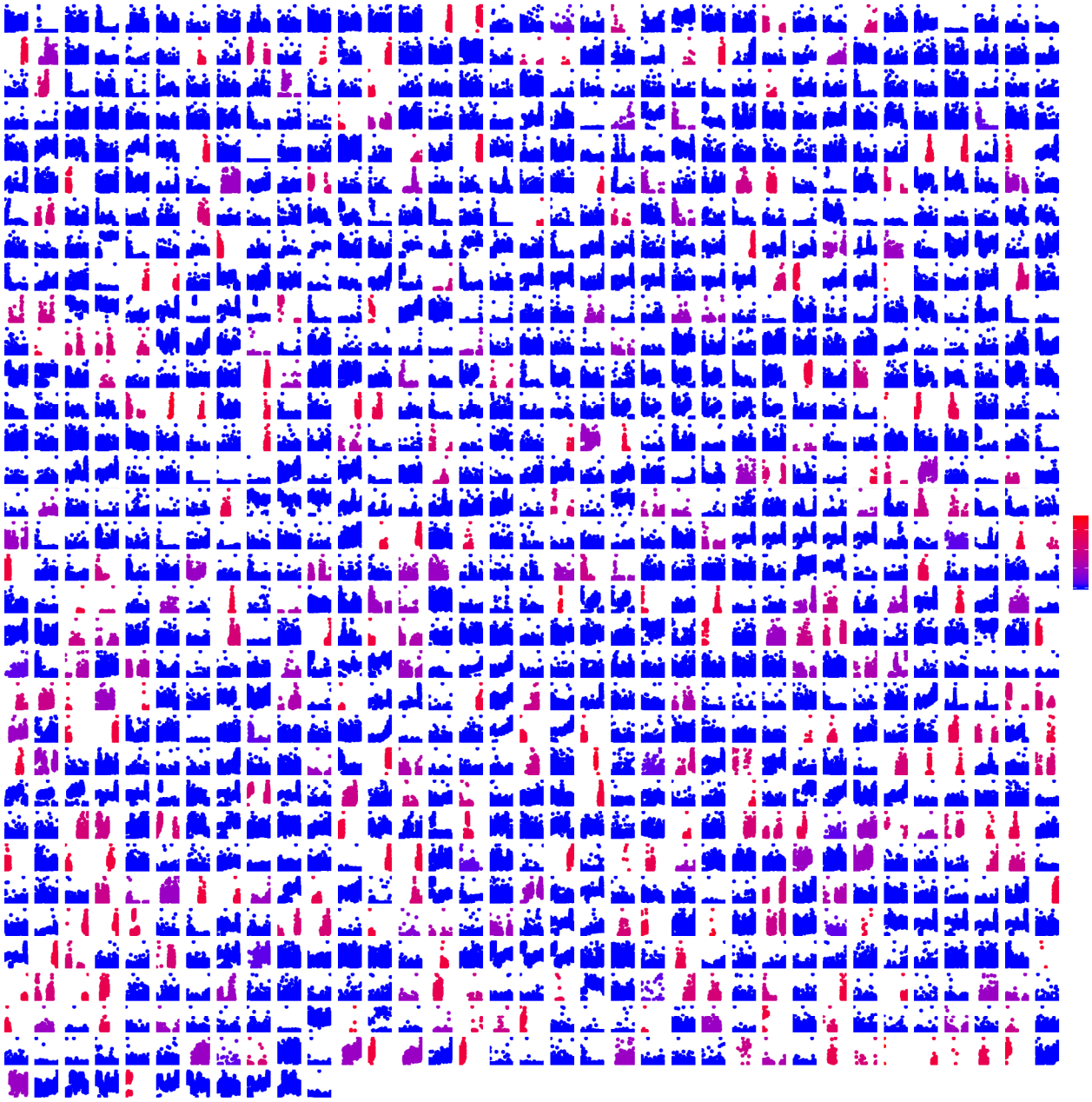
We measure samples on 96-well plates. For a given plate, wells are run one at a time, before moving onto the next plate. If a metabolite is unmeasured across an entire plate, we label that plate as misaligned. We then aggregate those results to calculate the mean misalignment score for a metabolite across the entire dataset.

```
misalign <-
  dat %>%
  group_by(metaboliteID, plate) %>%
  summarise(misaligned = all(is.na(value))) %>%
  ungroup() %>%
  group_by(metaboliteID) %>%
  mutate(misaligned = mean(misaligned)) %>%
  ungroup()


dat %<>% left_join(misalign, by = c('metaboliteID', 'plate'))
```

Here we show two plots, one a smaller subset of the other to show greater detail. Text has been removed to mask data identifiers. Here we can evaluate misalignment by plotting the measured values versus the chronological order by which individual samples were run (the plateWellID is structured to be plotted in chronological order in this case.) We can look for signs of excess or unusual signal variation over chronological time. Note, an expected warning will appear to signify that points with `NA`, representing a metabolite that was not measured in a particular sample, are being omitted from the plot.

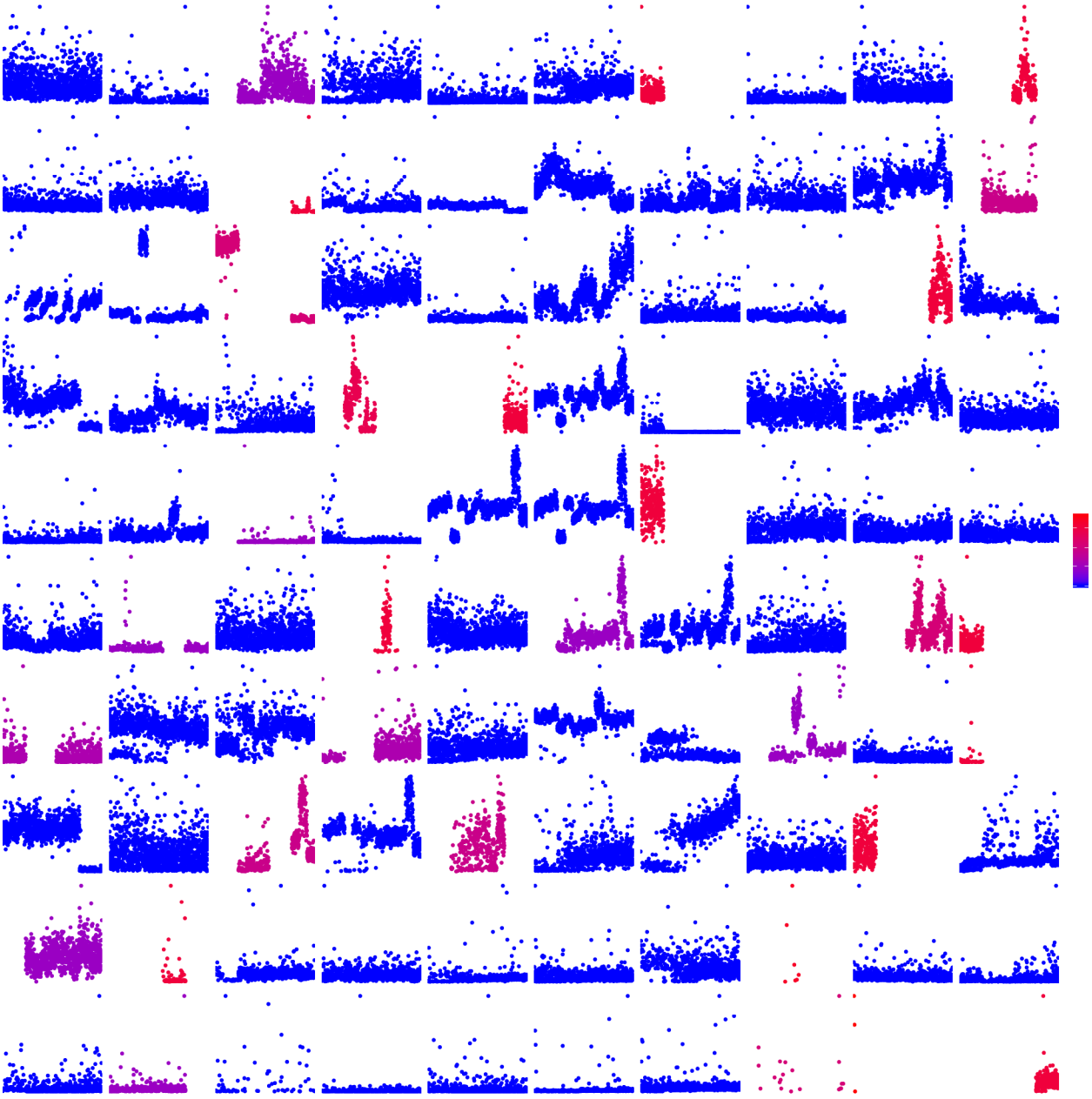
```r
metabsToPlot <-
  dat %>%
  distinct(metaboliteID) %>%
  sample_n(100)

blankThm <-
  theme(
    text = element_blank(),
    strip.background = element_blank(),
    strip.text = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank()
  )

dat %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  blankThm
```

```
## Warning: Removed 359384 rows containing missing values (geom_point).
```



```
dat %>%
  inner_join(metabsToPlot, by = 'metaboliteID') %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  blankThm
```

```
## Warning: Removed 34172 rows containing missing values (geom_point).
```



One option that can be applied to make these plots more useful is to cluster the plots based on the 'shape' of each graph. This can be useful because we expect certain types of sources of 'drift' (i.e. signal variation over time) to generate certain types of shapes of the data when plotted as a time series. The main caveat, when considering this option, is that calculating the clusters can be time-intensive with respect to computation. For example, for the dataset generating these plots, the computational processing time is roughly 10 minutes. Below, we cluster using the `dtwclust` package. This package offers a variety of clustering methods and parameters. Here, we will use hierarchical clustering and generate 10 clusters.

```
# Format the data for the clustering function
tsdat <-
  dat %>%
  select(metaboliteID, value) %>%
  split(.$metaboliteID) %>%
  map(function(x) {
```

```
    x[, 'value']
  })

nms <- names(tsdat)

tsdat %<>%
  bind_cols %>%
  set_names(nms) %>%
  as.data.frame %>%
  t()

# Here, we replace NA's with zero. Recognizing this step can be important for understanding why certain
tsdat[is.na(tsdat)] <- 0


# Start the clock!
ptm <- proc.time()

# Compute Hierarchical clusters
clusts <-
  dtwclust(tsdat,
           k = 10, # Number of clusters
           type = 'hierarchical', # Clustering technique
           method = 'ward.D2', # linkage method
           distance = 'sbd', # Shape based distance
           preproc = zscore)

# Get the cluster labels
clusterLabels <-
  data.frame(metaboliteID = nms,
             label =  cutree(clusts, k = 10))


# Stop the clock
ptmf <- proc.time() - ptm
```

```
# All these factors influence cluster time.
# Clustering Time (Elapsed, Seconds)
ptmf
```

```
##    user  system elapsed
## 267.199  64.957 333.162
```

```
# Dimensions of ouor data (row, column)
dim(dat)
```

```
## [1] 1409694      11
```

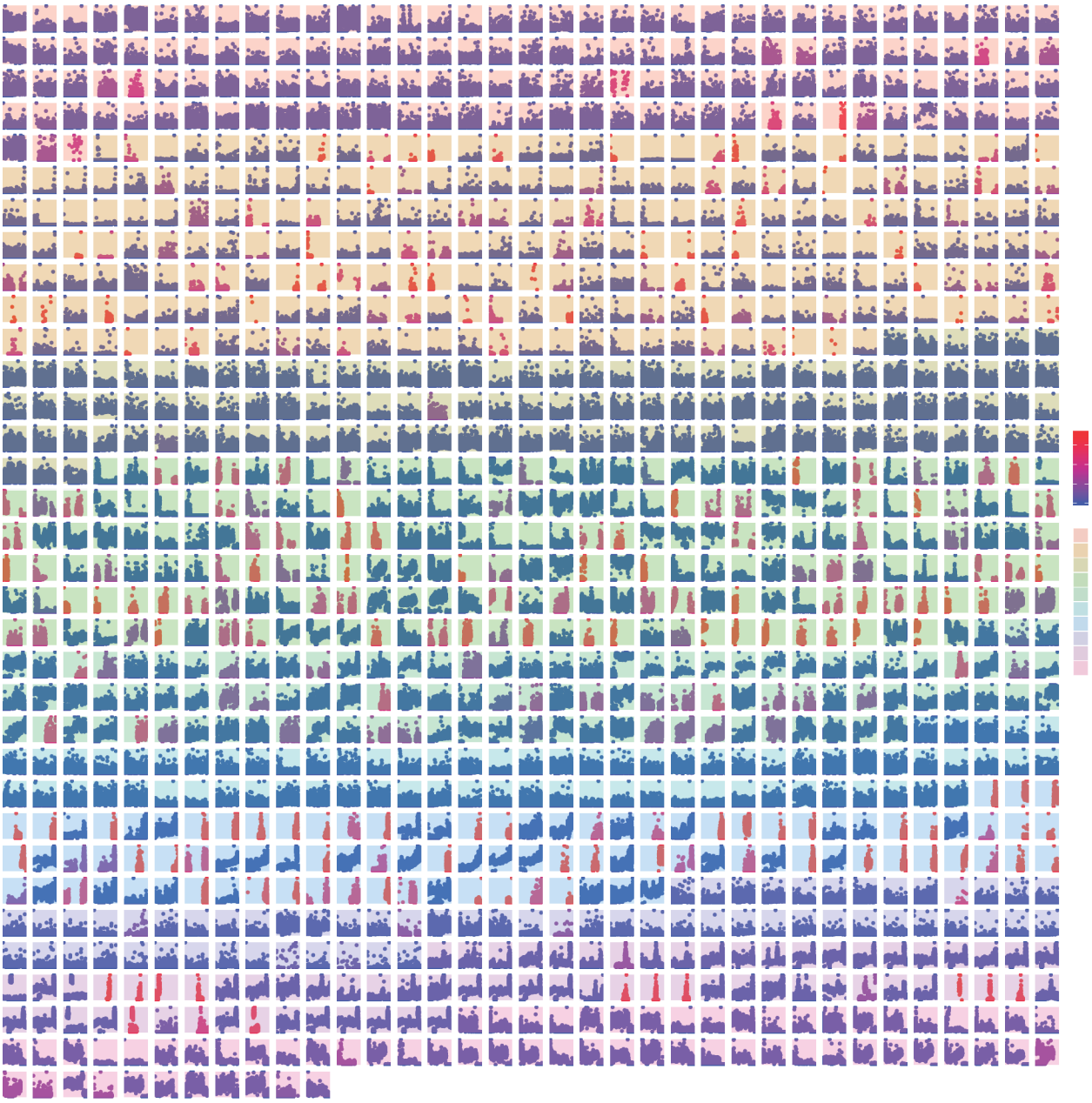
```
# Number of unique metabolites
length(unique(dat$metaboliteID))
```

```
## [1] 1166
```

When we recreate the plots using these labels, we see that like metabolites are now adjacent to one another. While not perfect, clusters of similar shapes can be identified. Here, coloured backgrounds are used to indicate which metabolites share the same cluster.

```r
# For coloured backgrounds
rectangles <-
  dat %>%
  group_by(metaboliteID) %>%
  arrange(plateWellID) %>%
  summarise(xmin = plateWellID[1],
            xmax = plateWellID[n()],
            ymin = min(value, na.rm = T),
            ymax = max(value, na.rm = T)) %>%
  left_join(clusterLabels) %>%
  mutate(label = as.factor(label))
```

```r
dat %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = rectangles) +
  blankThm
```



```
dat %>%
  inner_join(metabsToPlot, by = 'metaboliteID') %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  # mutate(plateWellID = as.numeric(as.factor(plateWellID))) %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = inner_join(rectangles, metabsToPlot, by = 'metaboliteID')) +
  blankThm
```

It may also be of interest to transform the data before clutering and plotting it, to see if other relationships can be revealed. Here, we reorder the data so that plates with the smallest mean value will be plotted first, while plates with the largest mean value will be plotted last. Within plates, points are plotted chronologically. This step can be helpful for shape-based clustering, when applied, to more specifically distinguish between metabolites more or less affected by certain types of sources of signal variation over time. For instance, plate re-ordered data may in some datasets allow for easier distinctions to be made between misalignment or batch effect (wherein step up patterns are more common with or without alterations in plate-specific dispersion or variance) versus more subtle instrument drift or contaminant; such re-ordering may also facilite shape based recognition of chemical instability patterns.

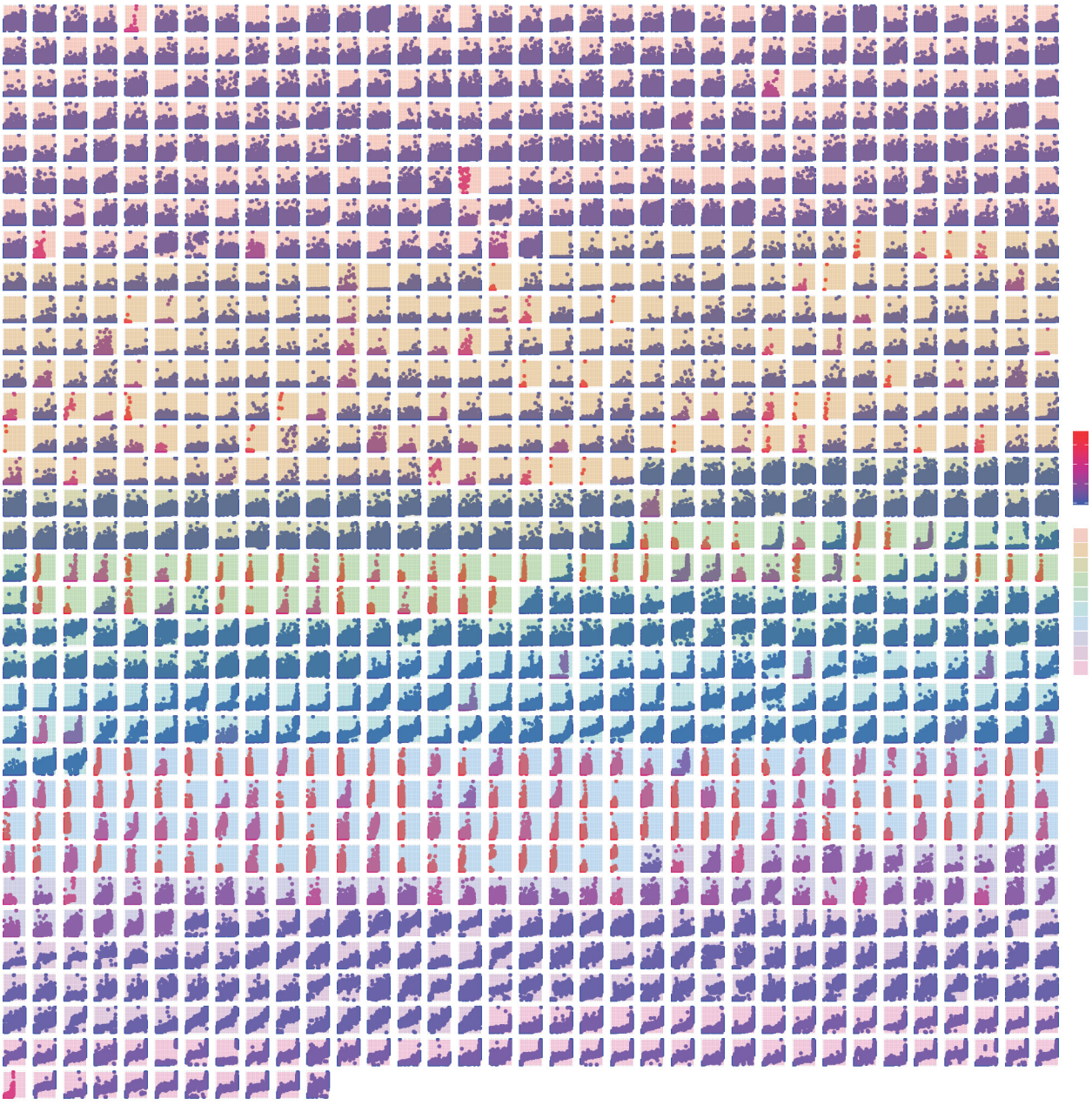
```
# Format the data for the clustering function
tsdat <-
  dat %>%
  select(plate, well, metaboliteID, value) %>%
```

```
  split(.$metaboliteID) %>%
  map(function(x) {
    x %<>%
      group_by(plate) %>%
      mutate(plateMean = mean(value, na.rm = T)) %>%
      ungroup() %>%
      arrange(plateMean, well) %>%
      select(value)

    x[, 'value']
  })

nms <- names(tsdat)

tsdat %<>%
  bind_cols %>%
  set_names(nms) %>%
  as.data.frame %>%
  t()

# Here, we replace NA's with zero. Recognizing this step can be important for understanding why certain
tsdat[is.na(tsdat)] <- 0

# Compute Hierarchical clusters
clusts <-
  dtwclust(tsdat,
           k = 10, # Number of clusters
           type = 'hierarchical', # Clustering technique
           method = 'ward.D2', # linkage method
           distance = 'sbd', # Shape based distance
           preproc = zscore)

# Get the cluster labels
clusterLabels <-
  data.frame(metaboliteID = nms,
             label =  cutree(clusts, k = 10))
```

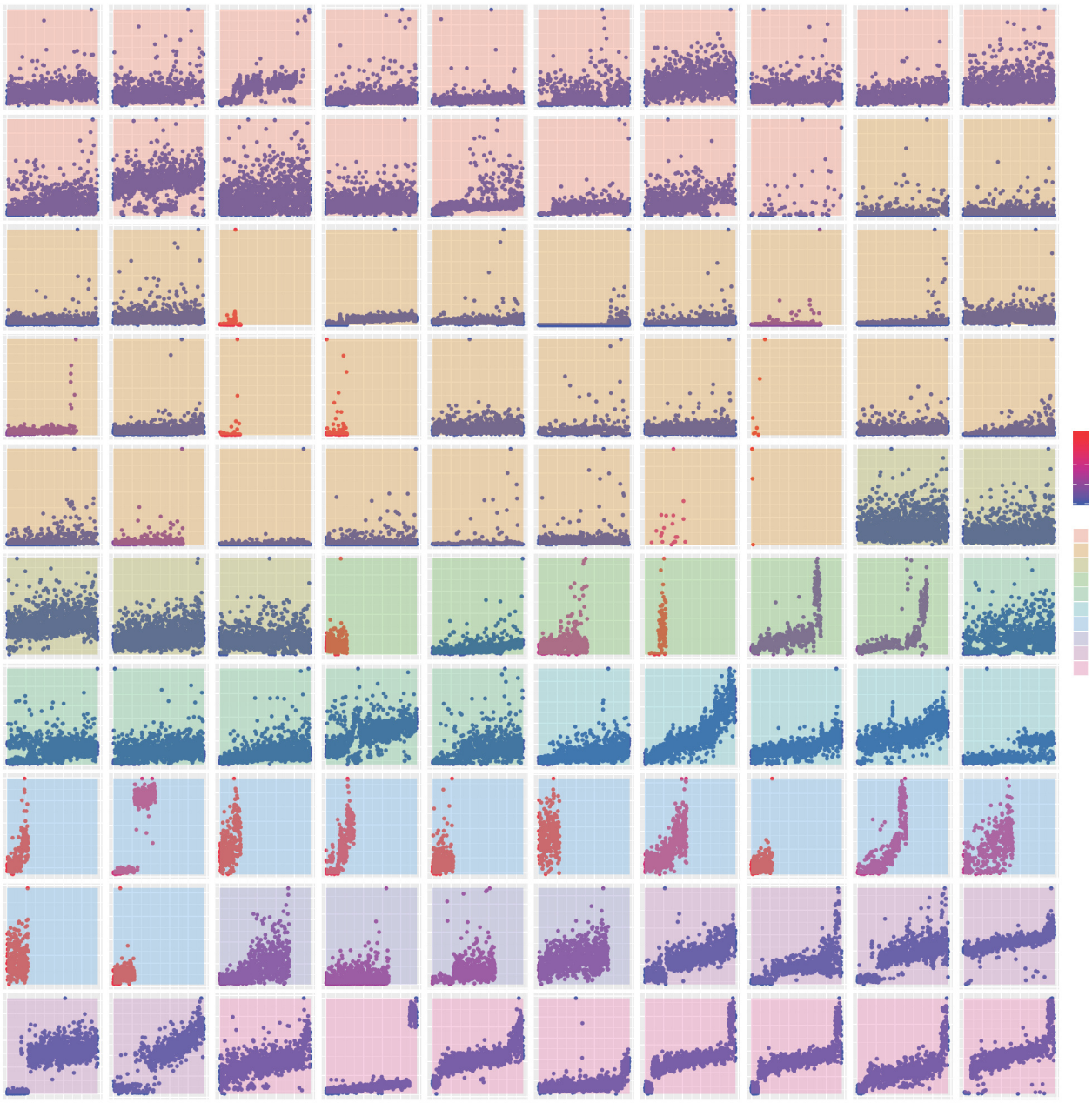
```
# For coloured backgrounds
pmoDat <-
  dat %>%
  group_by(metaboliteID, plate) %>%
  mutate(plateMean = mean(value, na.rm = T)) %>%
  ungroup() %>%
  group_by(metaboliteID) %>%
  arrange(plateMean, well) %>%
  mutate(plateWellID = 1:n())

rectangles <-
  pmoDat %>%
  group_by(metaboliteID) %>%
  arrange(plateWellID) %>%
  summarise(xmin = plateWellID[1],
            xmax = plateWellID[n()],
            ymin = min(value, na.rm = T),
```

```
            ymax = max(value, na.rm = T)) %>%
  left_join(clusterLabels) %>%
  mutate(label = as.factor(label))
```

```
pmoDat %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = rectangles) +
  blankThm
```



```
pmoDat %>%
  inner_join(metabsToPlot, by = 'metaboliteID') %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  # mutate(plateWellID = as.numeric(as.factor(plateWellID))) %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = inner_join(rectangles, metabsToPlot, by = 'metaboliteID')) +
  blankThm
```



Another transform would involve removing the top 5% of values from each metabolite before plotting. We

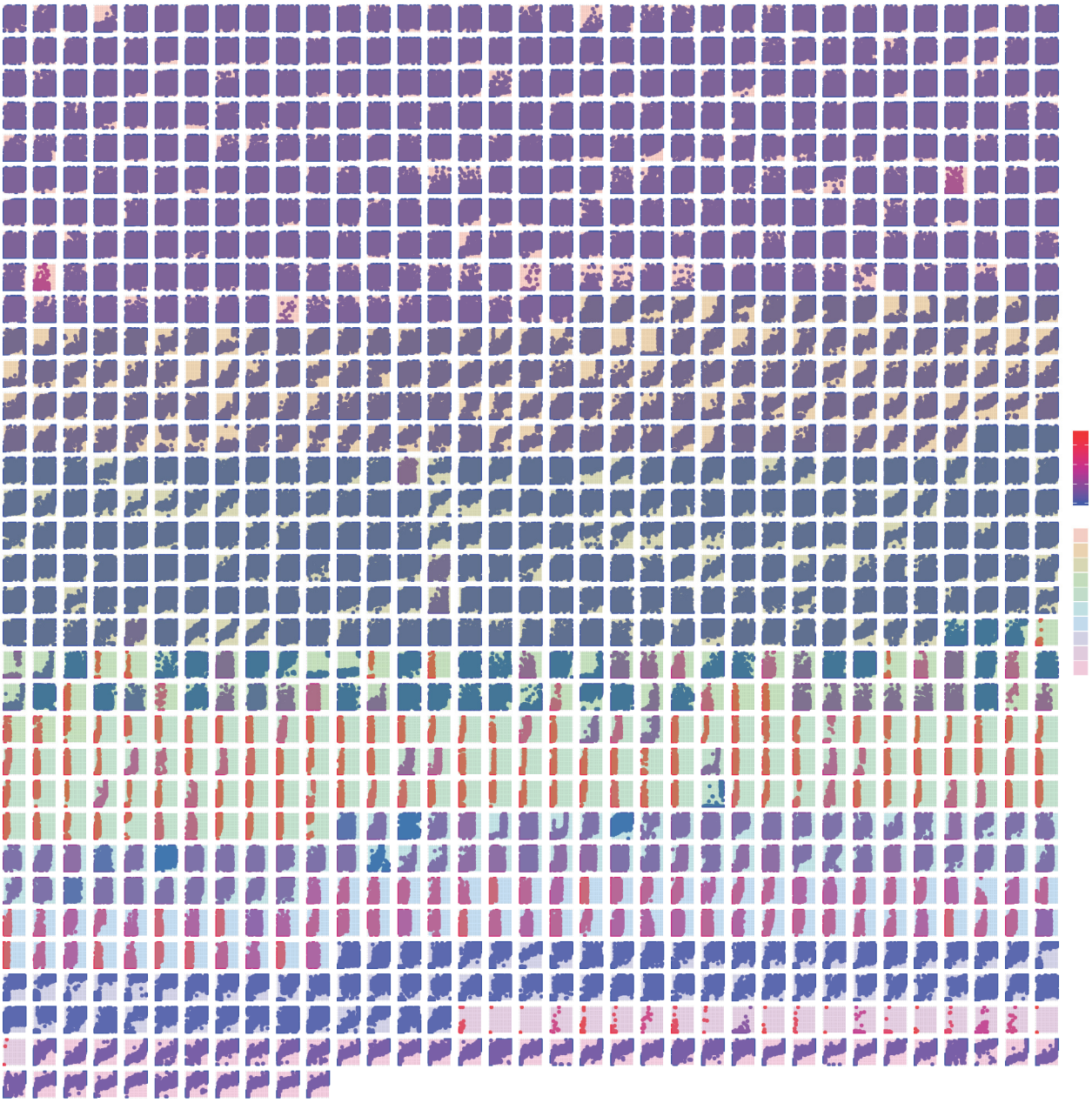
observed very high values in our data, and including these values obscures the detail contained in the majority of the data points. Thus, filtering out these values off can reveal more useful information about the structure of our data.

```
# Format the data for the clustering function
tsdat <-
  dat %>%
  select(plate, well, metaboliteID, value) %>%
  split(.$metaboliteID) %>%
  map(function(x) {
    x %<>%
      group_by(plate) %>%
      mutate(plateMean = mean(value, na.rm = T)) %>%
      ungroup() %>%
      arrange(plateMean, well) %>%
      mutate(value = ifelse(value <= quantile(value, probs = 0.95, na.rm = T), value, NA)) %>%
      select(value)

    x[, 'value']
  })

nms <- names(tsdat)

tsdat %<>%
  bind_cols %>%
  set_names(nms) %>%
  as.data.frame %>%
  t()

# Here, we replace NA's with zeros. Recognizing this step can be important for understanding why certai
tsdat[is.na(tsdat)] <- 0

# Compute Hierarchical clusters
clusts <-
  dtwclust(tsdat,
           k = 10, # Number of clusters
           type = 'hierarchical', # Clustering technique
           method = 'ward.D2', # linkage method
           distance = 'sbd', # Shape based distance
           preproc = zscore)

# Get the cluster labels
clusterLabels <-
  data.frame(metaboliteID = nms,
             label =  cutree(clusts, k = 10))
```

```
# For coloured backgrounds
pmotDat <-
  dat %>%
  group_by(metaboliteID, plate) %>%
  mutate(plateMean = mean(value, na.rm = T)) %>%
  ungroup() %>%
  group_by(metaboliteID) %>%
  mutate(value = ifelse(value <= quantile(value, probs = 0.95, na.rm = T), value, NA)) %>%
  arrange(plateMean, well) %>%
```

```
  mutate(plateWellID = 1:n())

rectangles <-
  pmotDat %>%
  group_by(metaboliteID) %>%
  arrange(plateWellID) %>%
  summarise(xmin = plateWellID[1],
            xmax = plateWellID[n()],
            ymin = min(value, na.rm = T),
            ymax = max(value, na.rm = T)) %>%
  left_join(clusterLabels) %>%
  mutate(label = as.factor(label))
```

```
pmotDat %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = rectangles) +
  blankThm
```



```
pmotDat %>%
  inner_join(metabsToPlot, by = 'metaboliteID') %>%
  left_join(clusterLabels, by  = 'metaboliteID') %>%
  # mutate(plateWellID = as.numeric(as.factor(plateWellID))) %>%
  ggplot() +
  geom_point(aes(x = plateWellID, y = value, colour = misaligned)) +
  facet_wrap( ~ label + metaboliteID, scales = 'free_y') +
  scale_color_gradient(low = 'blue', high = 'red') +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax, fill = label),
            alpha = 0.3,
            data = inner_join(rectangles, metabsToPlot, by = 'metaboliteID')) +
  blankThm
```