SOM3 - 3D Mesh distance analysis

2025-12-10 14:50:17.304111

Content

This script reads, measures and analysis 3D mesh data comparison between experimental samples.

Load required libraries

Define directories and file paths

```
# data folders
raw_dir <- "analysis/data/raw_data"  # .ply and .pp
out_csv  <- "analysis/data/derived_data" # stats and summarised data analysis
out_plots <- "analysis/plots/" # all plots and figures
snap_dir <- "analysis/plots/comparisons" # plot and figures for sample comparison
```

Parse filename to extract info about samples

```
parse_filename <- function(path) {
  fname <- fs::path_file(path)

  tibble(
    path  = path,
    file  = fname,

    # ID is everything before the first `_cycle`
    id = str_extract(fname, "^[^_]+(?=_cycle)") %||%
         str_extract(fname, "^[^_]+") %||%
         NA_character_,

    # Extract cycle number safely
    cycle = str_extract(fname, "cycle[0-9]+") %>%
            str_remove("cycle") %>%
            as.integer()
  )
}
```

Define a new variable not in the filename: raw material

```
raw_material_lookup <- tribble(
  ~id,   ~raw_material,
  "DWG1S1",  "Basalt",
  "DWG1S2",  "Basalt",
  "DWG1S3",  "Basalt",
  "DWG3S1",  "Scoria",
  "DWG3S2",  "Scoria",
  "DWG3S3",  "Scoria",
  "DWG4S1",  "Pumiceous",
  "DWG4S2",  "Pumiceous",
  "DWG4S3",  "Pumiceous",
  "KWG1S1",  "Glassy ignimbrite",
  "KWG1S2",  "Glassy ignimbrite",
  "KWG1S3",  "Glassy ignimbrite",
  "MW6G2S1", "Regular ignimbrite",
  "MW6G2S2", "Regular ignimbrite",
  "MW6G2S3", "Regular ignimbrite"
)
```

3D meshes and landmarks

Function to read landmarks (.pp)

```
read_pp <- function(file) {
  txt <- readLines(file, warn = FALSE)

  point_lines <- grep("<point", txt, value = TRUE)
  if (length(point_lines) == 0)
    stop("No <point ...> lines found in ", file)

  coords <- stringr::str_match(
    point_lines,
    'x="([0-9eE+\\.-]+)".*y="([0-9eE+\\.-]+)".*z="([0-9eE+\\.-]+)"'
  )[, 2:4]

  coords <- apply(coords, 2, as.numeric)

    rownames(coords) <- as.character(seq_len(nrow(coords)) - 1L)
  colnames(coords) <- c("x", "y", "z")

  coords
}
```

Load meshes and landmarks

```r
# List files
mesh_files     <- fs::dir_ls(raw_dir, glob = "*.ply")
landmark_files <- fs::dir_ls(raw_dir, glob = "*.pp")

# Parse filenames into metadata tables
mesh_info <- mesh_files %>% map_dfr(parse_filename)
land_info <- landmark_files %>% map_dfr(parse_filename)

# Ensure cycles exist for both meshes and landmarks
mesh_info  <- mesh_info  %>% filter(!is.na(cycle))
land_info  <- land_info  %>% filter(!is.na(cycle))

# Identify IDs with meshes and landmarks
common_ids <- intersect(mesh_info$id, land_info$id)

mesh_info <- mesh_info %>% filter(id %in% common_ids)
land_info <- land_info %>% filter(id %in% common_ids)

# Create unique names for lists: id_cycle, e.g., DWG4S3_cycle1
mesh_keys <- paste0(mesh_info$id, "_cycle", mesh_info$cycle)
land_keys <- paste0(land_info$id, "_cycle", land_info$cycle)

# Read meshes
meshes <- map(mesh_info$path, ~ {
  Rvcg::vcgPlyRead(.x, updateNormals = TRUE, clean = TRUE)
})
```

```
## Removed 2 duplicate 0 unreferenced vertices and 2 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 2 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 12 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 1 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 1 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 1 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 3 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 3 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 13 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 11 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 15 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 5 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 16 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 5 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 6 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 10 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 3 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 4 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 3 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 14 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 7 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 2 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 4 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 2 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 2 duplicate faces
## Removed 0 duplicate 0 unreferenced vertices and 3 duplicate faces
```

```r
names(meshes) <- mesh_keys

# Read landmarks
lands <- map(land_info$path, read_pp)
names(lands) <- land_keys

head(mesh_info, 5)
```

```
## # A tibble: 5 x 4
##   path                                      file              id     cycle
##   <fs::path>                                <chr>             <chr>  <int>
## 1 analysis/data/raw_data/DWG1S1_cycle0.ply DWG1S1_cycle0.ply DWG1S1     0
## 2 analysis/data/raw_data/DWG1S1_cycle2.ply DWG1S1_cycle2.ply DWG1S1     2
## 3 analysis/data/raw_data/DWG1S2_cycle0.ply DWG1S2_cycle0.ply DWG1S2     0
## 4 analysis/data/raw_data/DWG1S2_cycle2.ply DWG1S2_cycle2.ply DWG1S2     2
## 5 analysis/data/raw_data/DWG1S3_cycle0.ply DWG1S3_cycle0.ply DWG1S3     0
```

```r
head(land_info, 5)
```

```
## # A tibble: 5 x 4
##   path                                     file              id     cycle
##   <fs::path>                               <chr>             <chr>  <int>
## 1 analysis/data/raw_data/DWG1S1_cycle0.pp DWG1S1_cycle0.pp DWG1S1     0
## 2 analysis/data/raw_data/DWG1S1_cycle2.pp DWG1S1_cycle2.pp DWG1S1     2
## 3 analysis/data/raw_data/DWG1S2_cycle0.pp DWG1S2_cycle0.pp DWG1S2     0
## 4 analysis/data/raw_data/DWG1S2_cycle2.pp DWG1S2_cycle2.pp DWG1S2     2
## 5 analysis/data/raw_data/DWG1S3_cycle0.pp DWG1S3_cycle0.pp DWG1S3     0
```

Landmark checking in a table (including z, x, y)

```r
summarize_landmarks_id_coords <- function(id, land_info, lands) {

  # All cycles available for this ID
  cycles <- land_info |>
    dplyr::filter(id == !!id) |>
    arrange(cycle) |>
    pull(cycle)

  # Load all cycles' landmarks for this ID
  lms <- lapply(cycles, function(cyc) {
    key <- paste0(id, "_cycle", cyc)
    lm  <- lands[[key]]

    if (is.null(lm)) {
      warning("Landmarks missing for: ", key)
      return(NULL)
    }

    list(cycle = cyc, lm = lm)
  })
```

```r
  # Remove NULL entries
  lms <- lms[!sapply(lms, is.null)]

  # If nothing left, skip
  if (length(lms) == 0) return(NULL)

  # Determine reference landmark IDs
  if (0 %in% cycles) {
    ref_ids <- rownames(lands[[paste0(id, "_cycle0")]])
  } else {
    ref_ids <- rownames(lms[[1]]$lm)
  }

  # Build long table
  long_tbl <- dplyr::bind_rows(lapply(lms, function(entry) {
    cyc <- entry$cycle
    lm  <- entry$lm

    tibble::tibble(
      id = id,
      cycle = cyc,
      landmark_id = as.integer(rownames(lm)),
      x = lm[,1],
      y = lm[,2],
      z = lm[,3],
      missing_vs_ref = paste(setdiff(ref_ids, rownames(lm)), collapse = ", ")
    )
  }))

  return(long_tbl)
}


# Run for all IDs
landmark_comparison_table_coords <- purrr::map_dfr(
  unique(land_info$id),
  summarize_landmarks_id_coords,
  land_info = land_info,
  lands     = lands
)

readr::write_csv(
  landmark_comparison_table_coords,
  file.path(out_csv, "landmark_summary_all_samples.csv")
)

head(landmark_comparison_table_coords, 5)
```

```
## # A tibble: 5 x 7
##   id     cycle landmark_id     x     y     z missing_vs_ref
##   <chr>  <int>       <int> <dbl> <dbl> <dbl> <chr>
## 1 DWG1S1     0           0  22.4 25.3   30.5 ""
## 2 DWG1S1     0           1  40.0 12.8   32.9 ""
```

```
## 3 DWG1S1     0           2  27.6 11.3   50.5 ""
## 4 DWG1S1     0           3  15.3 -5.90  53.1 ""
## 5 DWG1S1     0           4 -12.1  8.49  11.4 ""
```

Compute vertex distance (VD)

Compute and export VD data

```
compute_vertex_distances <- function(mesh_ref, mesh_other) {

  kd <- Rvcg::vcgClostKD(mesh_other, mesh_ref)

  d <- kd$quality

  d[!is.finite(d)] <- 0
  d[d < 0] <- 0

  return(d)
}


compute_distance_table_id <- function(id, mesh_info, meshes) {

  pair <- mesh_info |>
    dplyr::filter(id == !!id & cycle %in% c(0, 2))

  if (!all(c(0, 2) %in% pair$cycle)) {
    message("skipping ", id, ": missing cycle0 or cycle2.")
    return(NULL)
  }

  key0 <- paste0(id, "_cycle0")
  key2 <- paste0(id, "_cycle2")

  mesh0 <- meshes[[key0]]
  mesh2 <- meshes[[key2]]

  dists <- compute_vertex_distances(mesh_ref = mesh0, mesh_other = mesh2)

  tibble::tibble(
    id          = id,
    cycle0_file = key0,
    cycle2_file = key2,
    vertex_id   = seq_along(dists),
    distance    = dists
  )
}


export_all_distances_one_csv <- function(mesh_info,
                                         meshes,
```

```
                                        outfile = "analysis/data/derived_data/all_vertex_distances_cyc

  ids <- unique(mesh_info$id)

  df_list <- lapply(ids, function(id) compute_distance_table_id(id, mesh_info, meshes))

  df_list <- df_list[!sapply(df_list, is.null)]

  df_all <- dplyr::bind_rows(df_list) %>%
    dplyr::left_join(raw_material_lookup, by = "id")
  outdir <- dirname(outfile)
  if (!dir.exists(outdir)) dir.create(outdir, recursive = TRUE)

  readr::write_csv(df_all, outfile)

  message("Exported all vertex distances with raw material → ", outfile)

  return(df_all)
}

df_all <- export_all_distances_one_csv(mesh_info, meshes)
```

```
## Exported all vertex distances with raw material → analysis/data/derived_data/all_vertex_distances_cy
```

```
head(df_all, 5)
```

```
## # A tibble: 5 x 6
##   id     cycle0_file   cycle2_file   vertex_id distance raw_material
##   <chr>  <chr>         <chr>             <int>    <dbl> <chr>
## 1 DWG1S1 DWG1S1_cycle0 DWG1S1_cycle2         1  0       Basalt
## 2 DWG1S1 DWG1S1_cycle0 DWG1S1_cycle2         2  0.00305 Basalt
## 3 DWG1S1 DWG1S1_cycle0 DWG1S1_cycle2         3  0       Basalt
## 4 DWG1S1 DWG1S1_cycle0 DWG1S1_cycle2         4  0       Basalt
## 5 DWG1S1 DWG1S1_cycle0 DWG1S1_cycle2         5  0       Basalt
```

Plot VD data

```
# Use violin plot per raw material

df_plot <- df_all %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = as.factor(raw_material))

p_vertex_rm <- ggplot(df_plot, aes(x = raw_material, y = distance, fill = raw_material)) +
  geom_violin(trim = FALSE, alpha = 0.6) +
  scale_y_continuous(trans = "log10") +
  geom_boxplot(width = 0.15, alpha = 0.5, outlier.shape = NA) +
  labs(
    title = "Per-Vertex Distances by Raw Material",
    subtitle = "Cycle 0 vs Cycle 2 - point-to-surface distances (log scale)",
```

```
    x = "Raw material",
    y = "Distance (log10 scale)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "vertex_distance_violin_facet_rawmaterial.png"),
  plot = p_vertex_rm,
  width = 12,
  height = 10,
  dpi = 300
)
```

```
## Warning in scale_y_continuous(trans = "log10"): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```

```
## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_ydensity()`).
```

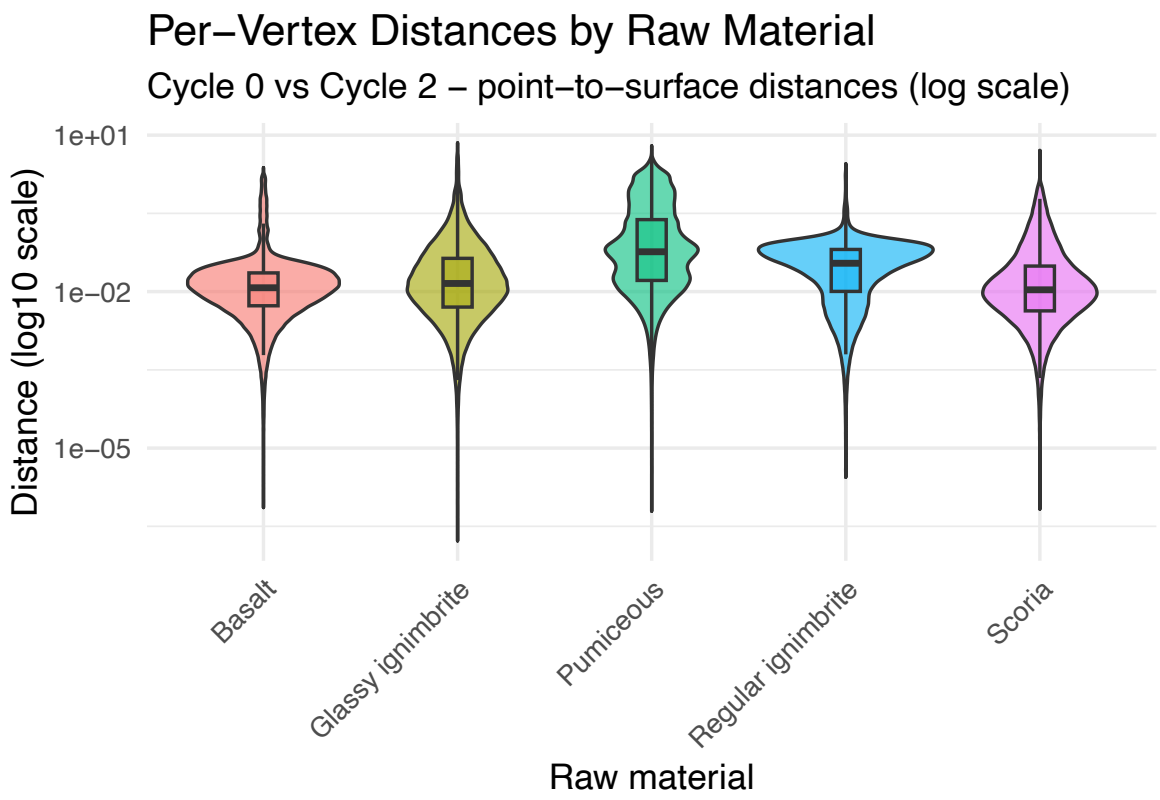
```
## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

```
# Violin plot faceted by sample ID

p_vertex_id <- ggplot(df_plot, aes(x = id, y = distance, fill = raw_material)) +
  geom_violin(trim = FALSE, alpha = 0.6) +
  geom_boxplot(width = 0.15, alpha = 0.5, outlier.shape = NA) +
  scale_y_continuous(trans = "log10") +
  labs(
    title = "Per-Vertex Distances by Raw Material",
    subtitle = "Faceted by Sample ID (Cycle 0 vs Cycle 2)",
    x = "Samples",
    y = "Distance (log10 scale)"
  ) +
  facet_wrap(~ raw_material, scales = "free_x") +  # <- changed here
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    strip.text = element_text(size = 10, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "vertex_distance_violin_facet_id.png"),
  plot = p_vertex_id,
  width = 12,
  height = 10,
  dpi = 300
)
```

```
## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_ydensity()`).

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

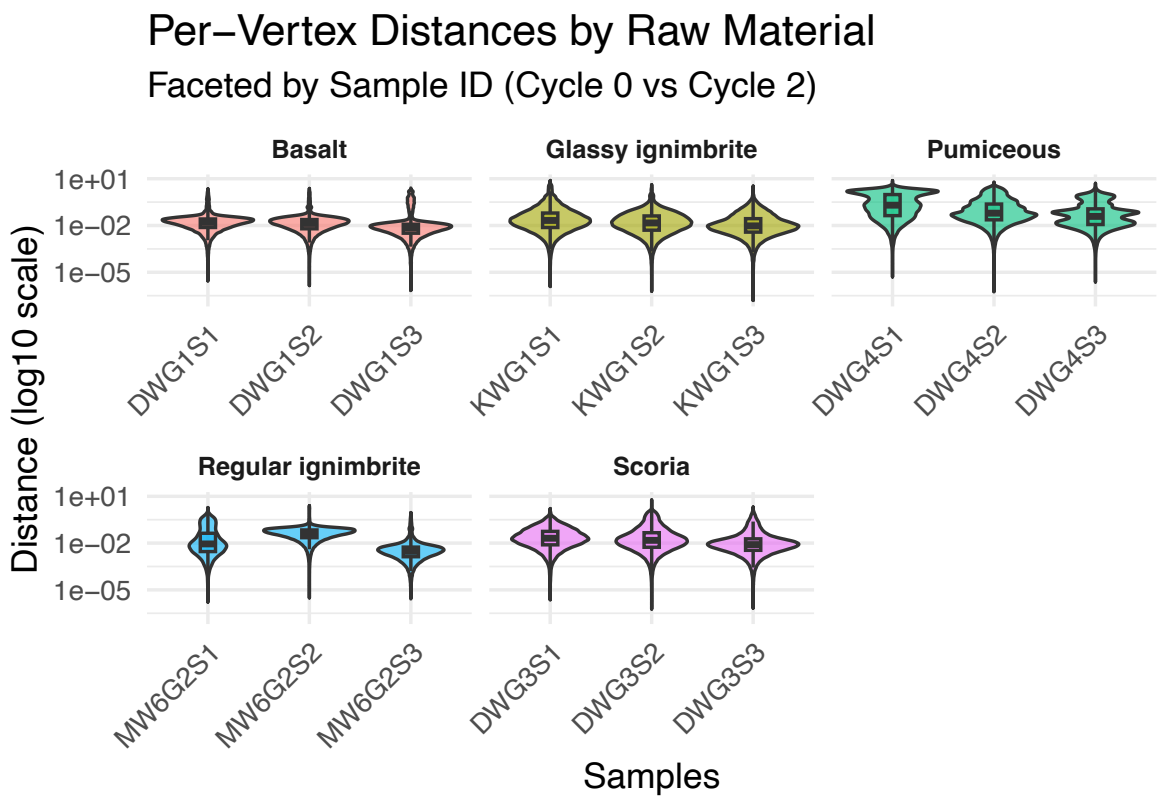
```
p_vertex_rm
```

```
## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_ydensity()`).

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



```
p_vertex_id
```

```
## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning in scale_y_continuous(trans = "log10"): log-10 transformation
## introduced infinite values.

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_ydensity()`).

## Warning: Removed 2750073 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



Compute VD stats (difference between raw material)

```
df_test <- df_all %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = as.factor(raw_material))


# Kruskal-Wallis
kw_result <- kruskal_test(df_test, distance ~ raw_material)
kw_result
```

```
## # A tibble: 1 x 6
##   .y.            n statistic    df     p method
## * <chr>      <int>     <dbl> <int> <dbl> <chr>
## 1 distance 3761250    37370.     4     0 Kruskal-Wallis
```

```
# EFFECT SIZE (Eta-squared)
kw_effect <- df_test %>%
  kruskal_effsize(distance ~ raw_material)
kw_effect
```

```
## # A tibble: 1 x 5
##   .y.            n effsize method  magnitude
## * <chr>      <int>   <dbl> <chr>   <ord>
## 1 distance 3761250 0.00993 eta2[H] small
```

```
# work with means beacause dataset is massive

df_id_summary <- df_test %>%
  group_by(id, raw_material) %>%
  summarise(
    mean_dist   = mean(distance, na.rm = TRUE),
    median_dist = median(distance, na.rm = TRUE),
    .groups = "drop"
  )

# POST-HOC TEST: Dunn test with BH correction
dunn_res <- df_id_summary %>%
  dunn_test(mean_dist ~ raw_material, p.adjust.method = "BH")

dunn_res
```

```
## # A tibble: 10 x 9
##    .y.       group1      group2    n1    n2 statistic      p  p.adj p.adj.signif
##  * <chr>     <chr>       <chr>  <int> <int>     <dbl>  <dbl>  <dbl> <chr>
##  1 mean_dist Basalt      Glass~     3     3    0.822  0.411  0.588  ns
##  2 mean_dist Basalt      Pumic~     3     3    2.10   0.0358 0.119  ns
##  3 mean_dist Basalt      Regul~     3     3   -0.365  0.715  0.871  ns
##  4 mean_dist Basalt      Scoria     3     3   -0.274  0.784  0.871  ns
##  5 mean_dist Glassy ign~ Pumic~     3     3    1.28   0.201  0.456  ns
##  6 mean_dist Glassy ign~ Regul~     3     3   -1.19   0.235  0.456  ns
##  7 mean_dist Glassy ign~ Scoria     3     3   -1.10   0.273  0.456  ns
##  8 mean_dist Pumiceous   Regul~     3     3   -2.46   0.0137 0.0881 ns
##  9 mean_dist Pumiceous   Scoria     3     3   -2.37   0.0176 0.0881 ns
## 10 mean_dist Regular ig~ Scoria     3     3    0.0913 0.927  0.927  ns
```

```
# Kolmogorov-Smirnov tests for every pair of raw materials

# Kolmogorov-Smirnov tests for every pair of raw materials

df_ks <- df_all %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = as.factor(raw_material))
```

```r
materials <- unique(df_ks$raw_material)

ks_two_materials <- function(mat1, mat2, df) {
  d1 <- df %>% filter(raw_material == mat1) %>% pull(distance)
  d2 <- df %>% filter(raw_material == mat2) %>% pull(distance)

  ks <- ks.test(d1, d2)

  tibble(
    material1 = mat1,
    material2 = mat2,
    statistic = ks$statistic,
    p_value   = ks$p.value
  )
}

# CREATE LIST OF ALL PAIRS
material_pairs <- combn(materials, 2, simplify = FALSE)

df_ks_results <- purrr::map_dfr(
  material_pairs,
  ~ ks_two_materials(.x[1], .x[2], df_ks)
)
```

```
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
```

```r
df_ks_results
```

```
## # A tibble: 10 x 4
##    material1         material2          statistic p_value
##    <fct>             <fct>                  <dbl>   <dbl>
##  1 Basalt            Scoria                0.0833       0
##  2 Basalt            Pumiceous             0.117        0
```

```
##   3 Basalt           Glassy ignimbrite     0.0511       0
##   4 Basalt           Regular ignimbrite    0.135        0
##   5 Scoria           Pumiceous             0.103        0
##   6 Scoria           Glassy ignimbrite     0.0776       0
##   7 Scoria           Regular ignimbrite    0.144        0
##   8 Pumiceous        Glassy ignimbrite     0.0666       0
##   9 Pumiceous        Regular ignimbrite    0.0858       0
##  10 Glassy ignimbrite Regular ignimbrite    0.0923       0
```

```r
# Run KS for all pairs
ks_results <- map_dfr(material_pairs, ~ ks_two_materials(.x[1], .x[2], df_ks))
```

```
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
## Warning in ks.test.default(d1, d2): p-value will be approximate in the presence
## of ties
```

```r
# Adjust p-values for multiple testing (BH recommended)
ks_results <- ks_results %>%
  mutate(p_adjusted = p.adjust(p_value, method = "BH"),
         signif = case_when(
           p_adjusted < 0.001 ~ "***",
           p_adjusted < 0.01  ~ "**",
           p_adjusted < 0.05  ~ "*",
           TRUE               ~ "ns"
         ))

ks_results
```

```
## # A tibble: 10 x 6
##    material1         material2          statistic p_value p_adjusted signif
##    <fct>             <fct>                  <dbl>   <dbl>      <dbl> <chr>
##  1 Basalt            Scoria                0.0833       0          0 ***
##  2 Basalt            Pumiceous             0.117        0          0 ***
##  3 Basalt            Glassy ignimbrite     0.0511       0          0 ***
##  4 Basalt            Regular ignimbrite    0.135        0          0 ***
##  5 Scoria            Pumiceous             0.103        0          0 ***
```

```
##  6 Scoria            Glassy ignimbrite     0.0776       0          0 ***
##  7 Scoria            Regular ignimbrite    0.144        0          0 ***
##  8 Pumiceous         Glassy ignimbrite     0.0666       0          0 ***
##  9 Pumiceous         Regular ignimbrite    0.0858       0          0 ***
## 10 Glassy ignimbrite Regular ignimbrite    0.0923       0          0 ***
```

```r
# Plot results of KS analysis

# Use the KS results from the previous chunk
df_heat <- ks_results %>%
  mutate(
    material1 = as.character(material1),
    material2 = as.character(material2)
  )

# Build a symmetric matrix for heat map
materials <- sort(unique(c(df_heat$material1, df_heat$material2)))

mat <- matrix(0, nrow = length(materials), ncol = length(materials),
              dimnames = list(materials, materials))

for (i in seq_len(nrow(df_heat))) {
  m1 <- df_heat$material1[i]
  m2 <- df_heat$material2[i]
  stat <- df_heat$statistic[i]

  mat[m1, m2] <- stat
  mat[m2, m1] <- stat
}

df_mat <- as_tibble(mat, rownames = "material1") %>%
  pivot_longer(-material1, names_to = "material2", values_to = "ks_statistic") %>%
  mutate(pair = paste(material1, material2))   # <-- ADD THIS

df_mat <- df_mat %>%
  left_join(
    ks_results %>%
      mutate(pair = paste(material1, material2)) %>%   # ensure same format
      select(pair, signif),
    by = "pair"
  )

# Remove self-comparisons
df_mat$ks_statistic[df_mat$material1 == df_mat$material2] <- NA
df_mat$signif[df_mat$material1 == df_mat$material2] <- ""

df_mat
```

```
## # A tibble: 25 x 5
##    material1         material2          ks_statistic pair                 signif
##    <chr>             <chr>                     <dbl> <chr>                <chr>
##  1 Basalt            Basalt                       NA Basalt Basalt        ""
##  2 Basalt            Glassy ignimbrite        0.0511 Basalt Glassy ignim~ "***"
##  3 Basalt            Pumiceous                0.117  Basalt Pumiceous     "***"
```

```
##  4 Basalt            Regular ignimbrite       0.135  Basalt Regular igni~ "***"
##  5 Basalt            Scoria                   0.0833 Basalt Scoria        "***"
##  6 Glassy ignimbrite Basalt                   0.0511 Glassy ignimbrite B~  <NA>
##  7 Glassy ignimbrite Glassy ignimbrite       NA      Glassy ignimbrite G~ ""
##  8 Glassy ignimbrite Pumiceous                0.0666 Glassy ignimbrite P~  <NA>
##  9 Glassy ignimbrite Regular ignimbrite       0.0923 Glassy ignimbrite R~ "***"
## 10 Glassy ignimbrite Scoria                   0.0776 Glassy ignimbrite S~  <NA>
## # i 15 more rows
```

```r
# Plot heat map
p_vd_heat <- ggplot(df_mat, aes(x = material1, y = material2, fill = ks_statistic)) +
  geom_tile(color = "white") +
  geom_text(aes(label = signif), size = 6, color = "black") +
  scale_fill_viridis_c(option = "plasma", na.value = "grey90") +
  labs(
    title = "Heatmap of KS Statistics Between Raw Materials",
    subtitle = "KS statistic (D) with significance annotations",
    x = "Material",
    y = "Material",
    fill = "KS Statistic (D)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid = element_blank()
  )

ggsave(
  filename = file.path(out_plots, "KSanalysis_heat_rawmaterial.png"),
  plot = p_vd_heat,
  width = 12,
  height = 10,
  dpi = 300
)
```

```
## Warning: Removed 10 rows containing missing values or values outside the scale range
## (`geom_text()`).
```

Summarise VD data

```r
# Compute summary stats for one ID
compute_distance_summary_id <- function(id, mesh_info, meshes) {

  pair <- mesh_info |>
    dplyr::filter(id == !!id & cycle %in% c(0,2))

  if (!all(c(0,2) %in% pair$cycle)) {
    message("Skipping ", id, ": missing cycle0 or cycle2.")
    return(NULL)
  }

  key0 <- paste0(id, "_cycle0")
```

```r
  key2 <- paste0(id, "_cycle2")

  mesh0 <- meshes[[key0]]
  mesh2 <- meshes[[key2]]

  d <- compute_vertex_distances(mesh0, mesh2)

  tibble::tibble(
    id          = id,
    cycle0_file = key0,
    cycle2_file = key2,
    mean_dist   = mean(d),
    median_dist = median(d),
    rms_dist    = sqrt(mean(d^2)),
    min_dist    = min(d),
    max_dist    = max(d),
    n_vertices  = length(d)
  )
}

# Compute summary stats for ALL IDs and export to CSV
export_all_distance_summaries <- function(mesh_info,
                                          meshes,
                                          outfile = "analysis/data/derived_data/summary_vertex_distance

  ids <- unique(mesh_info$id)

  summaries <- lapply(ids, function(id) compute_distance_summary_id(id, mesh_info, meshes))

  summaries <- summaries[!sapply(summaries, is.null)]

  df_summary <- dplyr::bind_rows(summaries) %>%
    dplyr::left_join(raw_material_lookup, by = "id")

  outdir <- dirname(outfile)
  if (!dir.exists(outdir)) dir.create(outdir, recursive = TRUE)

  readr::write_csv(df_summary, outfile)

  message("Exported summary stats with raw material → ", outfile)

  return(df_summary)
}


df_summary <- export_all_distance_summaries(mesh_info, meshes)
```

```
## Exported summary stats with raw material → analysis/data/derived_data/summary_vertex_distances_cycle
```

```r
df_summary
```

```
## # A tibble: 15 x 10
```

```
##    id      cycle0_file    cycle2_file    mean_dist median_dist rms_dist min_dist
##    <chr>   <chr>          <chr>              <dbl>       <dbl>    <dbl>    <dbl>
##  1 DWG1S1  DWG1S1_cycle0  DWG1S1_cycle2    0.00853      0        0.0531        0
##  2 DWG1S2  DWG1S2_cycle0  DWG1S2_cycle2    0.00910      0        0.0559        0
##  3 DWG1S3  DWG1S3_cycle0  DWG1S3_cycle2    0.0155       0        0.108         0
##  4 DWG3S1  DWG3S1_cycle0  DWG3S1_cycle2    0.0106       0        0.0449        0
##  5 DWG3S2  DWG3S2_cycle0  DWG3S2_cycle2    0.00853      0        0.0623        0
##  6 DWG3S3  DWG3S3_cycle0  DWG3S3_cycle2    0.0116       0        0.0573        0
##  7 DWG4S1  DWG4S1_cycle0  DWG4S1_cycle2    0.0689       0        0.312         0
##  8 DWG4S2  DWG4S2_cycle0  DWG4S2_cycle2    0.0575       0        0.263         0
##  9 DWG4S3  DWG4S3_cycle0  DWG4S3_cycle2    0.0696       0        0.223         0
## 10 KWG1S1  KWG1S1_cycle0  KWG1S1_cycle2    0.0254       0        0.174         0
## 11 KWG1S2  KWG1S2_cycle0  KWG1S2_cycle2    0.0119       0        0.0689        0
## 12 KWG1S3  KWG1S3_cycle0  KWG1S3_cycle2    0.0138       0        0.0514        0
## 13 MW6G2S1 MW6G2S1_cycle0 MW6G2S1_cycle2   0.00268      0        0.0283        0
## 14 MW6G2S2 MW6G2S2_cycle0 MW6G2S2_cycle2   0.0307       0.0142   0.0667        0
## 15 MW6G2S3 MW6G2S3_cycle0 MW6G2S3_cycle2   0.00333      0        0.0206        0
## # i 3 more variables: max_dist <dbl>, n_vertices <int>, raw_material <chr>
```

Compute Root Mean Square (RMS)

Compute RMS distance using vcgClostKD

```r
# Compute RMS for one sample ID
compute_rms_for_id <- function(id, mesh_info, meshes) {

  pair <- mesh_info |>
    filter(id == !!id & cycle %in% c(0,2))

  if (!all(c(0,2) %in% pair$cycle)) {
    message("Skipping ", id, ": missing cycle0 or cycle2.")
    return(NULL)
  }

  key0 <- paste0(id, "_cycle0")
  key2 <- paste0(id, "_cycle2")

  mesh0 <- meshes[[key0]]
  mesh2 <- meshes[[key2]]

  kd <- Rvcg::vcgClostKD(mesh2, mesh0)
  d  <- kd$quality

  d[!is.finite(d)] <- 0
  d[d < 0] <- 0

  tibble(
    id          = id,
    cycle0_file = key0,
    cycle2_file = key2,
    rms_dist    = sqrt(mean(d^2)),
```

```
    mean_dist   = mean(d),
    median_dist = median(d),
    min_dist    = min(d),
    max_dist    = max(d),
    n_vertices  = length(d)
  )
}

# Compute vcgClostKD RMS for all IDs
compute_all_rms <- function(mesh_info, meshes) {

  ids <- unique(mesh_info$id)

  summaries <- lapply(ids, function(id) compute_rms_for_id(id, mesh_info, meshes))
  summaries <- summaries[!sapply(summaries, is.null)]

  df_rms <- bind_rows(summaries) %>%
    left_join(raw_material_lookup, by = "id")

  df_rms
}

# Run it
df_rms <- compute_all_rms(mesh_info, meshes)
df_rms
```

```
## # A tibble: 15 x 10
##    id      cycle0_file    cycle2_file    rms_dist mean_dist median_dist min_dist
##    <chr>   <chr>          <chr>             <dbl>     <dbl>       <dbl>    <dbl>
##  1 DWG1S1  DWG1S1_cycle0  DWG1S1_cycle2    0.0531   0.00853      0             0
##  2 DWG1S2  DWG1S2_cycle0  DWG1S2_cycle2    0.0559   0.00910      0             0
##  3 DWG1S3  DWG1S3_cycle0  DWG1S3_cycle2    0.108    0.0155       0             0
##  4 DWG3S1  DWG3S1_cycle0  DWG3S1_cycle2    0.0449   0.0106       0             0
##  5 DWG3S2  DWG3S2_cycle0  DWG3S2_cycle2    0.0623   0.00853      0             0
##  6 DWG3S3  DWG3S3_cycle0  DWG3S3_cycle2    0.0573   0.0116       0             0
##  7 DWG4S1  DWG4S1_cycle0  DWG4S1_cycle2    0.312    0.0689       0             0
##  8 DWG4S2  DWG4S2_cycle0  DWG4S2_cycle2    0.263    0.0575       0             0
##  9 DWG4S3  DWG4S3_cycle0  DWG4S3_cycle2    0.223    0.0696       0             0
## 10 KWG1S1  KWG1S1_cycle0  KWG1S1_cycle2    0.174    0.0254       0             0
## 11 KWG1S2  KWG1S2_cycle0  KWG1S2_cycle2    0.0689   0.0119       0             0
## 12 KWG1S3  KWG1S3_cycle0  KWG1S3_cycle2    0.0514   0.0138       0             0
## 13 MW6G2S1 MW6G2S1_cycle0 MW6G2S1_cycle2   0.0283   0.00268      0             0
## 14 MW6G2S2 MW6G2S2_cycle0 MW6G2S2_cycle2   0.0667   0.0307       0.0142        0
## 15 MW6G2S3 MW6G2S3_cycle0 MW6G2S3_cycle2   0.0206   0.00333      0             0
## # i 3 more variables: max_dist <dbl>, n_vertices <int>, raw_material <chr>
```

```
readr::write_csv(
  df_rms,
  file.path(out_csv, "ClostKD_cycle0_vs_cycle2.csv")
)
```

Plots RMS vcgClostKD data

```
df_plot <- df_rms %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = as.factor(raw_material))

p_CKD_RMS <- ggplot(df_plot, aes(x = raw_material, y = rms_dist, fill = raw_material)) +
  geom_violin(trim = FALSE, alpha = 0.6) +
  geom_boxplot(width = 0.15, alpha = 0.5, outlier.shape = NA) +
  geom_jitter(width = 0.05, size = 2, alpha = 0.7) +
  labs(
    title = "RMS Distance (Cycle 0 vs Cycle 2) - Grouped by Raw material",
    x = "Raw material",
    y = "RMS distance"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "vcgClostKD_rawmaterial.png"),
  plot = p_CKD_RMS,
  width = 12,
  height = 10,
  dpi = 300
)
```

Plot RMS vcgClostKD data heatmap

```
mean_rms <- df_rms %>%
  filter(!is.na(raw_material)) %>%
  group_by(raw_material) %>%
  summarise(
    mean_RMS = mean(rms_dist),
    .groups = "drop"
  )

materials <- mean_rms$raw_material

diff_mat <- outer(
  mean_rms$mean_RMS,
  mean_rms$mean_RMS,
  FUN = function(a, b) abs(a - b)
)

colnames(diff_mat) <- materials
rownames(diff_mat) <- materials
```

```r
df_diff <- as.data.frame(diff_mat) %>%
  mutate(material1 = rownames(.)) %>%
  pivot_longer(-material1, names_to = "material2", values_to = "diff_RMS")

# 3. Plot heat map
p_CKD_RMS_heat <-ggplot(df_diff, aes(x = material1, y = material2, fill = diff_RMS)) +
  geom_tile(color = "white") +
  geom_text(aes(label = sprintf("%.4f", diff_RMS)), color = "black", size = 4) +
  scale_fill_viridis_c(option = "plasma") +
  labs(
    title = "Heatmap of RMS differences between Raw materials",
    subtitle = "Absolute differences in mean RMS values (Cycle 0 vs Cycle 2)",
    x = "Raw Material",
    y = "Raw Material",
    fill = "RMS"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid = element_blank()
  )

ggsave(
  filename = file.path(out_plots, "vcgClostKD_heat_rawmaterial.png"),
  plot = p_CKD_RMS_heat,
  width = 12,
  height = 10,
  dpi = 300
)
```

Run ANOVA for RMS vcgClostKD data

```r
df_test <- df_rms %>%
  filter(!is.na(raw_material))

# 1. ANOVA (parametric)
anova_model <- aov(rms_dist ~ raw_material, data = df_test)
anova_summary <- summary(anova_model)
anova_summary
```

```
##              Df  Sum Sq  Mean Sq F value   Pr(>F)
## raw_material  4 0.10198 0.025495   15.86 0.000249 ***
## Residuals    10 0.01607 0.001607
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# 2. ANOVA Assumptions

# Normality of residuals
shapiro_result <- shapiro_test(residuals(anova_model))
shapiro_result
```

```
## # A tibble: 1 x 3
##   variable               statistic p.value
##   <chr>                      <dbl>   <dbl>
## 1 residuals(anova_model)     0.942   0.413
```

```
# Homogeneity of variance (Levene test)
levene_result <- levene_test(rms_dist ~ raw_material, data = df_test)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
levene_result
```

```
## # A tibble: 1 x 4
##     df1   df2 statistic     p
##   <int> <int>     <dbl> <dbl>
## 1     4    10     0.531 0.716
```

```
# Interpretation:
# - If residuals are NOT normal OR Levene test is significant (p < 0.05),
#   use Kruskal-Wallis instead of ANOVA.

# 3. Kruskal-Wallis (non-parametric alternative)
kw_result <- kruskal_test(rms_dist ~ raw_material, data = df_test)
kw_result
```

```
## # A tibble: 1 x 6
##   .y.          n statistic    df      p method
## * <chr>    <int>     <dbl> <int>  <dbl> <chr>
## 1 rms_dist    15      8.53     4 0.0739 Kruskal-Wallis
```

```
# Effect size (Kruskal eta-squared)
kw_effect <- kruskal_effsize(rms_dist ~ raw_material, data = df_test)
kw_effect
```

```
## # A tibble: 1 x 5
##   .y.          n effsize method  magnitude
## * <chr>    <int>   <dbl> <chr>   <ord>
## 1 rms_dist    15   0.453 eta2[H] large
```

```
# 4. Post-hoc tests

# ANOVA post-hoc (Tukey HSD)
tukey_result <- TukeyHSD(anova_model)
tukey_result
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rms_dist ~ raw_material, data = df_test)
```

```
##
## $raw_material
##                                          diff         lwr         upr
## Glassy ignimbrite-Basalt              0.02563653 -0.08209489  0.13336795
## Pumiceous-Basalt                      0.19366050  0.08592908  0.30139192
## Regular ignimbrite-Basalt            -0.03391706 -0.14164848  0.07381436
## Scoria-Basalt                        -0.01761034 -0.12534176  0.09012108
## Pumiceous-Glassy ignimbrite           0.16802396  0.06029254  0.27575538
## Regular ignimbrite-Glassy ignimbrite -0.05955360 -0.16728502  0.04817782
## Scoria-Glassy ignimbrite             -0.04324687 -0.15097829  0.06448455
## Regular ignimbrite-Pumiceous         -0.22757756 -0.33530898 -0.11984614
## Scoria-Pumiceous                     -0.21127083 -0.31900225 -0.10353941
## Scoria-Regular ignimbrite             0.01630673 -0.09142469  0.12403815
##                                          p adj
## Glassy ignimbrite-Basalt             0.9298257
## Pumiceous-Basalt                     0.0010797
## Regular ignimbrite-Basalt            0.8333841
## Scoria-Basalt                        0.9810454
## Pumiceous-Glassy ignimbrite          0.0031428
## Regular ignimbrite-Glassy ignimbrite 0.4143498
## Scoria-Glassy ignimbrite             0.6855170
## Regular ignimbrite-Pumiceous         0.0002944
## Scoria-Pumiceous                     0.0005412
## Scoria-Regular ignimbrite            0.9857048
```

```r
# Kruskal-Wallis post-hoc (Dunn test with BH correction)
dunn_result <- df_test %>%
  dunn_test(rms_dist ~ raw_material, p.adjust.method = "BH")

dunn_result
```

```
## # A tibble: 10 x 9
##    .y.      group1      group2    n1    n2 statistic       p  p.adj p.adj.signif
##  * <chr>    <chr>       <chr>  <int> <int>     <dbl>   <dbl>  <dbl> <chr>
##  1 rms_dist Basalt      Glass~     3     3     0.365 0.715   0.715  ns
##  2 rms_dist Basalt      Pumic~     3     3     1.83  0.0679  0.226  ns
##  3 rms_dist Basalt      Regul~     3     3    -0.913 0.361   0.602  ns
##  4 rms_dist Basalt      Scoria     3     3    -0.365 0.715   0.715  ns
##  5 rms_dist Glassy ign~ Pumic~     3     3     1.46  0.144   0.360  ns
##  6 rms_dist Glassy ign~ Regul~     3     3    -1.28  0.201   0.402  ns
##  7 rms_dist Glassy ign~ Scoria     3     3    -0.730 0.465   0.665  ns
##  8 rms_dist Pumiceous   Regul~     3     3    -2.74  0.00617 0.0617 ns
##  9 rms_dist Pumiceous   Scoria     3     3    -2.19  0.0285  0.142  ns
## 10 rms_dist Regular ig~ Scoria     3     3     0.548 0.584   0.715  ns
```

Compute RMS using vcgMetro

```r
compute_metro_pair <- function(mesh1, mesh2) {
  # mesh1, mesh2 are mesh objects from Rvcg
  # order: distances from mesh1 -> mesh2 (ForwardSampling),
  #        and mesh2 -> mesh1 (BackwardSampling)
```

```
  md <- Rvcg::vcgMetro(mesh1, mesh2)

  tibble(
    RMS_forward = md$ForwardSampling$RMSdist,
    RMS_backward = md$BackwardSampling$RMSdist,
    mean_forward = md$ForwardSampling$MeanDist,
    mean_backward = md$BackwardSampling$MeanDist,
    max_forward = md$ForwardSampling$MaxDist,
    max_backward = md$BackwardSampling$MaxDist
  )
}

compute_metro_for_id <- function(id, mesh_info, meshes) {

  sub <- mesh_info %>%
    filter(id == !!id & cycle %in% c(0, 2))

  if (!all(c(0, 2) %in% sub$cycle)) {
    message("Skipping ", id, ": missing cycle0 or cycle2.")
    return(NULL)
  }

  key0 <- paste0(id, "_cycle0")
  key2 <- paste0(id, "_cycle2")

  mesh0 <- meshes[[key0]]
  mesh2 <- meshes[[key2]]

  metro_stats <- compute_metro_pair(mesh0, mesh2)

  tibble(
    id          = id,
    cycle0_file = key0,
    cycle2_file = key2
  ) %>%
    bind_cols(metro_stats)
}

compute_all_metro <- function(mesh_info, meshes) {
  ids <- unique(mesh_info$id)

  res_list <- lapply(ids, function(id) compute_metro_for_id(id, mesh_info, meshes))
  res_list <- res_list[!sapply(res_list, is.null)]

  df_metro <- bind_rows(res_list) %>%
    left_join(raw_material_lookup, by = "id")

  df_metro
}

# run it
df_metro <- compute_all_metro(mesh_info, meshes)
```

```
## Number of samples set to 3037060
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%
##                       Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Samp
```

```
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampling face 6%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Vertex sampling
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
## Sampling vertices 1%Sampling vertices 3%Sampling vertices 5%Sampling vertices 7%Sampling vertices 9%S
##                      Similar Triangles face sampling
## Sampling face 0%Sampling face 1%Sampling face 2%Sampling face 3%Sampling face 4%Sampling face 5%Sampl
```

```
readr::write_csv(
  df_metro,
  file.path(out_csv, "metro_cycle0_vs_cycle2.csv")
)
```

Plot RMS vcgMetro data

```
# firward
df_forward <- df_metro %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = factor(raw_material))

metro_fr <- ggplot(df_forward, aes(x = raw_material, y = RMS_forward, fill = raw_material)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.05, size = 2.5, alpha = 0.9) +
  labs(
    title = "Metro RMS (Forward: mesh0 → mesh2)",
    x = "Raw material",
    y = "RMS forward"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "vector_fr.png"),
  plot = metro_fr,
  width = 12,
  height = 10,
  dpi = 300
)


# backwords
df_backward <- df_metro %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = factor(raw_material))

metro_br <-ggplot(df_backward, aes(x = raw_material, y = RMS_backward, fill = raw_material)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.05, size = 2.5, alpha = 0.9) +
  labs(
    title = "Metro RMS (Backward: mesh2 → mesh0)",
    x = "Raw material",
    y = "RMS backward"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
```

```
  )

ggsave(
  filename = file.path(out_plots, "vector_br.png"),
  plot = metro_br,
  width = 12,
  height = 10,
  dpi = 300
)
```

Run ANOVA for RMS vcgMetro data

```
df_forward <- df_metro %>%
  filter(!is.na(raw_material)) %>%
  mutate(raw_material = factor(raw_material))

# ANOVA
anova_forward <- aov(RMS_forward ~ raw_material, data = df_forward)
anova_summary <- summary(anova_forward)
anova_summary
```

```
##              Df  Sum Sq  Mean Sq F value Pr(>F)
## raw_material  4 0.08104 0.020259   4.368 0.0267 *
## Residuals    10 0.04638 0.004638
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ANOVA ASSUMPTIONS
# 1. Normality of residuals
residuals_normality <- shapiro_test(residuals(anova_forward))
residuals_normality
```

```
## # A tibble: 1 x 3
##   variable                 statistic p.value
##   <chr>                        <dbl>   <dbl>
## 1 residuals(anova_forward)     0.911   0.142
```

```
# 2. Homogeneity of variance (Levene's test)
levene_test_forward <- leveneTest(RMS_forward ~ raw_material, data = df_forward)
levene_test_forward
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  4  0.8658 0.5168
##       10
```

```
# Interpretation message
if (residuals_normality$p.value < 0.05) {
  message("Residuals are NOT normal: ANOVA assumptions violated. Use Kruskal-Wallis instead.")
```

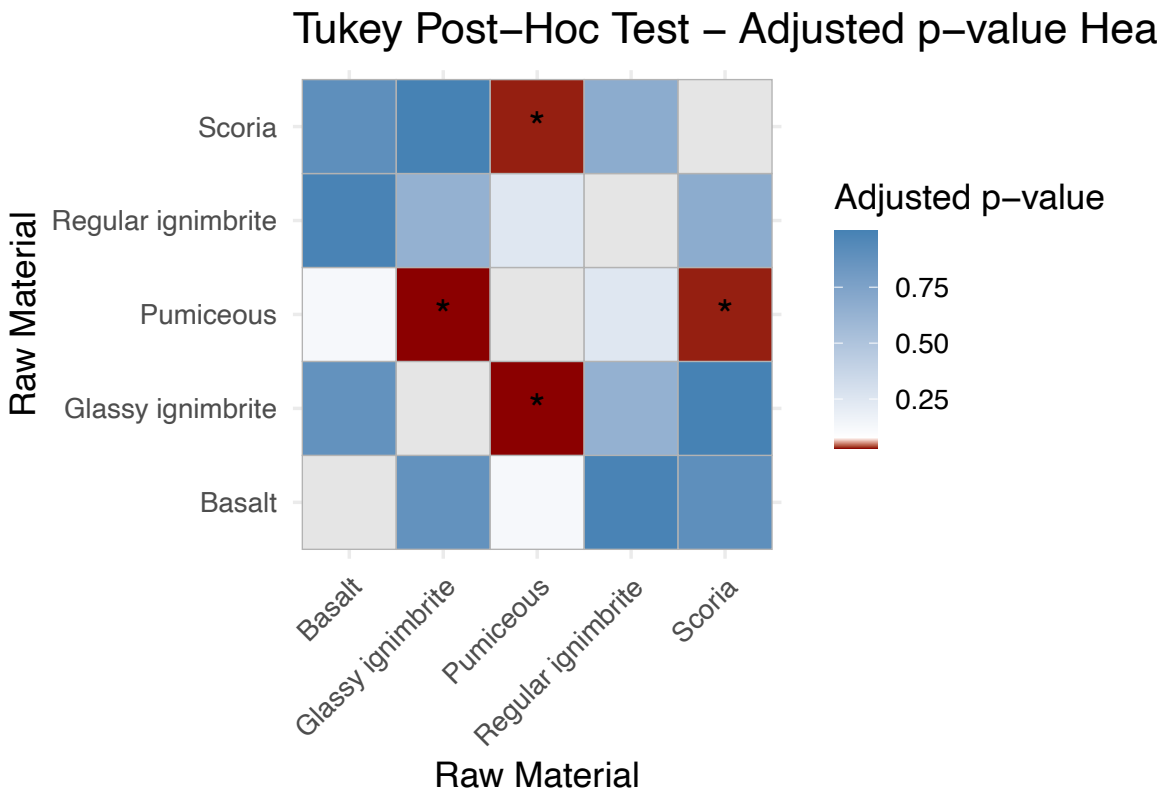
```r
}

if (levene_test_forward$`Pr(>F)`[1] < 0.05) {
  message("Variances are NOT equal: ANOVA assumptions violated. Use Kruskal-Wallis instead.")
}

# Tukey Post-hoc Test (Only valid if assumptions met)
tukey_forward <- TukeyHSD(anova_forward)
tukey_forward
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = RMS_forward ~ raw_material, data = df_forward)
##
## $raw_material
##                                          diff         lwr         upr
## Glassy ignimbrite-Basalt             -0.052056383 -0.23506621  0.13095344
## Pumiceous-Basalt                      0.149728231 -0.03328160  0.33273806
## Regular ignimbrite-Basalt             0.025928992 -0.15708084  0.20893882
## Scoria-Basalt                        -0.048397611 -0.23140744  0.13461222
## Pumiceous-Glassy ignimbrite           0.201784614  0.01877479  0.38479444
## Regular ignimbrite-Glassy ignimbrite  0.077985375 -0.10502445  0.26099520
## Scoria-Glassy ignimbrite              0.003658772 -0.17935106  0.18666860
## Regular ignimbrite-Pumiceous         -0.123799239 -0.30680907  0.05921059
## Scoria-Pumiceous                     -0.198125842 -0.38113567 -0.01511601
## Scoria-Regular ignimbrite            -0.074326603 -0.25733643  0.10868322
##                                          p adj
## Glassy ignimbrite-Basalt             0.8764889
## Pumiceous-Basalt                     0.1251602
## Regular ignimbrite-Basalt            0.9888146
## Scoria-Basalt                        0.9014127
## Pumiceous-Glassy ignimbrite          0.0296049
## Regular ignimbrite-Glassy ignimbrite 0.6397588
## Scoria-Glassy ignimbrite             0.9999949
## Regular ignimbrite-Pumiceous         0.2454605
## Scoria-Pumiceous                     0.0327841
## Scoria-Regular ignimbrite            0.6768545
```

Plot Tukey test

```r
tukey_tbl <- as.data.frame(tukey_forward$raw_material) %>%
  mutate(comparison = rownames(.)) %>%
  select(comparison, p_adj = `p adj`)

# Split "A-B" into two columns
tukey_tbl <- tukey_tbl %>%
  separate(comparison, into = c("mat1", "mat2"), sep = "-")

# Collect all raw materials (factor level order preserved)
```

```r
materials <- levels(df_forward$raw_material)

# --- Build a full matrix of p-values ---
mat <- matrix(NA, nrow = length(materials), ncol = length(materials),
              dimnames = list(materials, materials))

for (i in seq_len(nrow(tukey_tbl))) {
  m1 <- tukey_tbl$mat1[i]
  m2 <- tukey_tbl$mat2[i]
  p  <- tukey_tbl$p_adj[i]

  mat[m1, m2] <- p
  mat[m2, m1] <- p
}

diag(mat) <- NA   # no self-comparisons

# --- Convert to long format for ggplot ---
df_heat <- as.data.frame(mat) %>%
  mutate(material1 = rownames(.)) %>%
  pivot_longer(-material1, names_to = "material2", values_to = "p_value")

# Significance labels
df_heat <- df_heat %>%
  mutate(
    signif = case_when(
      is.na(p_value)         ~ "",
      p_value < 0.001        ~ "***",
      p_value < 0.01         ~ "**",
      p_value < 0.05         ~ "*",
      TRUE                   ~ ""
    )
  )

# --- Plot heatmap ---
ggplot(df_heat, aes(x = material1, y = material2, fill = p_value)) +
  geom_tile(color = "grey70") +
  geom_text(aes(label = signif), size = 6, color = "black") +
  scale_fill_gradientn(
    colours = c("darkred", "white", "steelblue"),
    values = scales::rescale(c(0, 0.05, 1)),
    na.value = "grey90",
    name = "Adjusted p-value"
  ) +
  coord_fixed() +
  labs(
    title = "Tukey Post-Hoc Test - Adjusted p-value Heatmap",
    x = "Raw Material",
    y = "Raw Material"
  ) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
ggsave(
  file.path(out_plots, "tukey_heatmap_RMS_forward.png"),
  width = 10, height = 8, dpi = 300
)
```

Hausdorff distance function

```
library(RANN)
```

```
## Warning: package 'RANN' was built under R version 4.3.3
```

```
hausdorff_mesh <- function(mesh1, mesh2) {
  # Extract vertex coordinates (remove homogeneous 4th row)
  v1 <- t(mesh1$vb[1:3, , drop = FALSE])
  v2 <- t(mesh2$vb[1:3, , drop = FALSE])

  # sanity check
  if (nrow(v1) < 3 | nrow(v2) < 3)
    stop("One of the meshes has fewer than 3 vertices.")

  # Directed distances
  nn12 <- nn2(data = v2, query = v1, k = 1)$nn.dists
  nn21 <- nn2(data = v1, query = v2, k = 1)$nn.dists
```

```
  list(
    directed_1_to_2 = max(nn12),
    directed_2_to_1 = max(nn21),
    hausdorff       = max(max(nn12), max(nn21)),
    mean_1_to_2     = mean(nn12),
    mean_2_to_1     = mean(nn21),
    q12             = quantile(nn12, c(.5, .9, .95, .99)),
    q21             = quantile(nn21, c(.5, .9, .95, .99))
  )
}
```

Generate comparisions

```
# Build table of available IDs and cycles
mesh_pairs <- mesh_info %>%
  select(id, cycle) %>%
  distinct() %>%
  group_by(id) %>%
  filter(all(c(0, 2) %in% cycle)) %>%   # ensure you have cycle0 and cycle2
  ungroup()

# Create pairs: id, mesh0_name, mesh2_name
comparison_table <- mesh_pairs %>%
  mutate(
    mesh0 = paste0(id, "_cycle0"),
    mesh2 = paste0(id, "_cycle2")
  ) %>%
  select(id, mesh0, mesh2) %>%
  left_join(raw_material_lookup, by = "id")
```

Compute Hausdorff distances

```
hausdorff_results <- comparison_table %>%
  mutate(
    res = map2(mesh0, mesh2, ~ {
      m1 <- meshes[[.x]]
      m2 <- meshes[[.y]]
      hausdorff_mesh(m1, m2)
    })
  ) %>%
  mutate(
    H          = map_dbl(res, "hausdorff"),
    h_1_to_2   = map_dbl(res, "directed_1_to_2"),
    h_2_to_1   = map_dbl(res, "directed_2_to_1"),
    mean_12    = map_dbl(res, "mean_1_to_2"),
    mean_21    = map_dbl(res, "mean_2_to_1")
  ) %>%
```

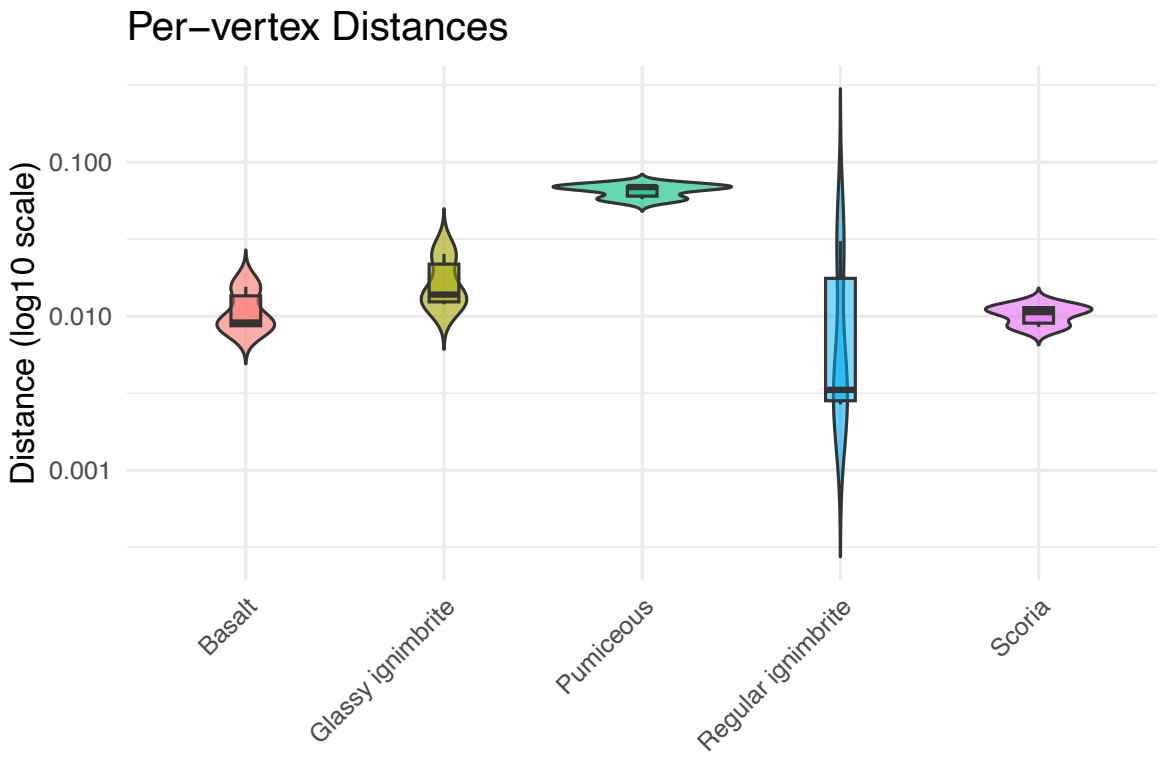
```
  select(id, raw_material, mesh0, mesh2,
         H, h_1_to_2, h_2_to_1, mean_12, mean_21)

write.csv(
  hausdorff_results,
  file.path(out_csv, "hausdorff_distances.csv"),
  row.names = FALSE
)
```

Merge Hausdorff results into the per-vertex dataset

```
df_plot2 <- df_plot %>%
  left_join(
    hausdorff_results %>% select(id, H, h_1_to_2, h_2_to_1),
    by = "id"
  )
```

Per-vertex individuals

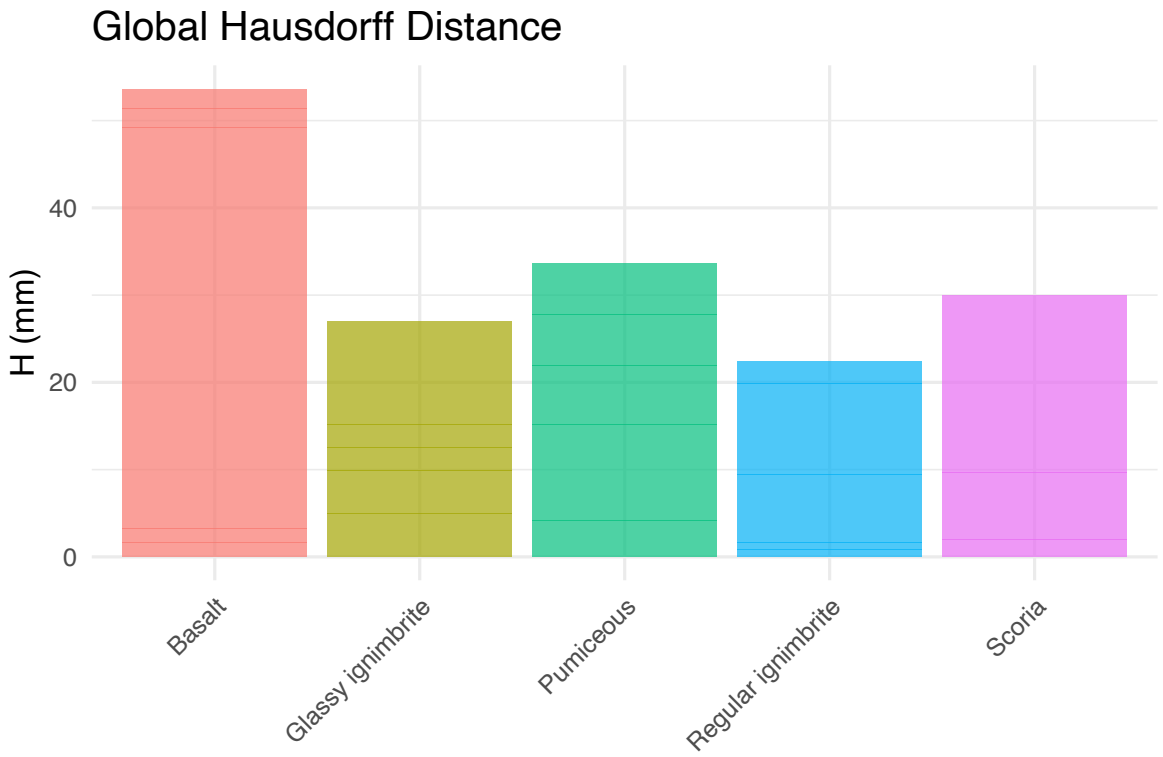
```
pA <- ggplot(df_plot2, aes(x = raw_material, y = mean_dist, fill = raw_material)) +
  geom_violin(trim = FALSE, alpha = .6) +
  geom_boxplot(width = 0.15, alpha = .5, outlier.shape = NA) +
  scale_y_continuous(trans = "log10") +
  labs(
    title = "Per-vertex Distances",
    y = "Distance (log10 scale)",
    x = ""
  ) +
  theme_minimal(base_size = 13) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none"
  )

pA
```




```
ggsave(
  filename = file.path(out_plots, "per-vertex_id.png"),
  plot = pA,
  width = 12,
  height = 10,
  dpi = 300
)
```

Global Hdf distance

```
pB <- ggplot(hausdorff_results, aes(x = raw_material, y = H, fill = raw_material)) +
  geom_col(alpha = 0.7) +
  labs(
    title = "Global Hausdorff Distance",
    x = "",
    y = "H (mm)"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none"
  )

pB
```



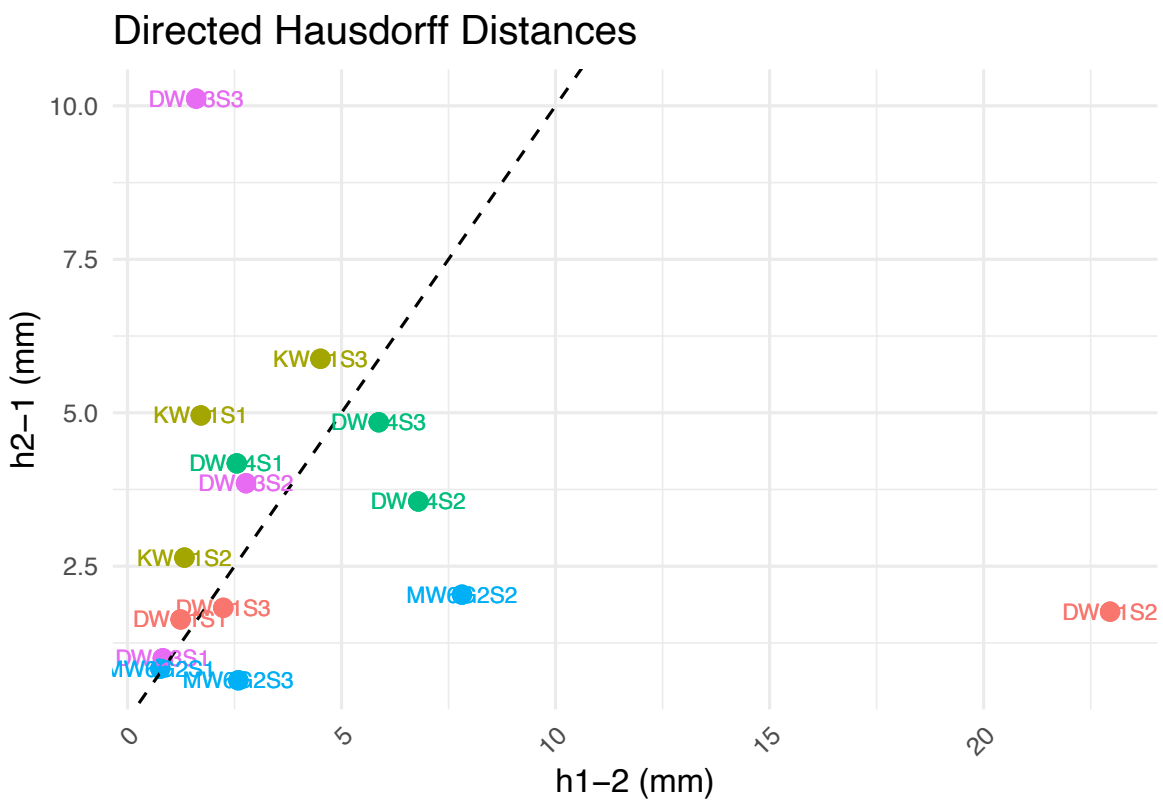

```
ggsave(
  filename = file.path(out_plots, "hdf_dist_global.png"),
  plot = pB,
  width = 12,
  height = 10,
  dpi = 300
)

# test

p_haus_violin <- ggplot(hausdorff_results, aes(x = raw_material, y = mean_21, fill = raw_material)) +
  geom_violin(alpha = 0.6, trim = FALSE) +
  geom_jitter(width = 0.08, alpha = 0.8) +
  labs(
    title = "Distribution of Hausdorff distances per Raw material",
    x = "Raw material",
    y = "Hausdorff distance"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "p_haus.png"),
  plot = p_haus_violin,
  width = 12,
```

```
  height = 10,
  dpi = 300
)

# test2
# boxplots

p_haus_box <- ggplot(hausdorff_results, aes(x = raw_material, y = mean_21, fill = raw_material)) +
  geom_boxplot(alpha = 0.7, outlier.shape = NA) +
  geom_jitter(width = 0.05, size = 2.5, alpha = 0.8) +
  labs(
    title = "Hausdorff Distance by Raw Material",
    x = "Raw material",
    y = "Hausdorff distance"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "haus_box.png"),
  plot = p_haus_box,
  width = 12,
  height = 10,
  dpi = 300
)
```

direct distance

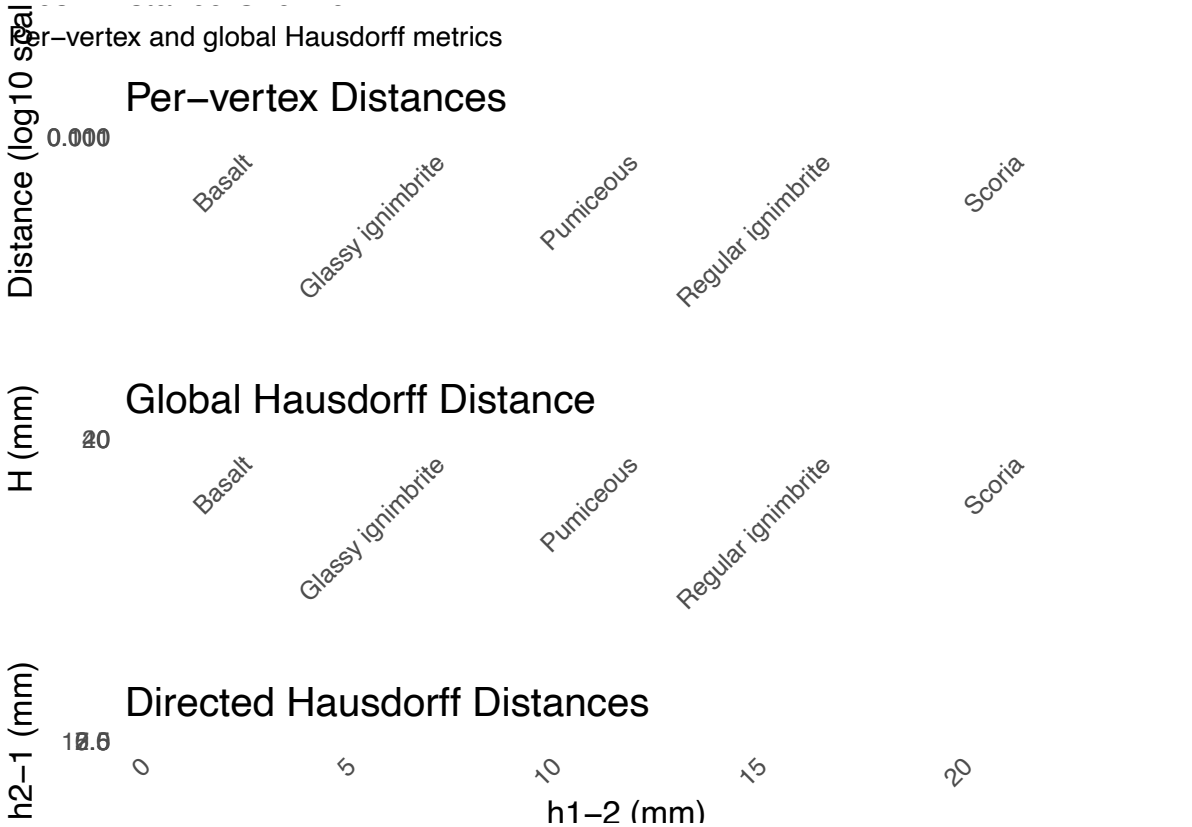
```
pC <- ggplot(hausdorff_results,
             aes(x = h_1_to_2, y = h_2_to_1, label = id, color = raw_material)) +
  geom_point(size = 3) +
  geom_abline(intercept = 0, slope = 1, linetype = 2) +
  geom_text(nudge_y = 0.003, size = 3) +
  labs(
    title = "Directed Hausdorff Distances",
    x = "h1-2 (mm)",
    y = "h2-1 (mm)"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

ggsave(
  filename = file.path(out_plots, "direct_dist_id.png"),
  plot = pC,
  width = 12,
```

```
  height = 10,
  dpi = 300
)

pC
```



combine 3 panels

```
library(patchwork)

multi_panel <- pA / pB / pC +
  plot_annotation(title = "Mesh Distance Overview",
                  subtitle = "Per-vertex and global Hausdorff metrics")

multi_panel
```




```
ggsave(
  filename = file.path(out_plots, "multi_panel.png"),
  plot = multi_panel,
  width = 12,
  height = 10,
  dpi = 300
)
```

End script