# seqCAT: a Bioconductor R-package for variant analysis of high throughput sequencing data: supplementary information

This document shows the code needed to create all the figures in the article and works as a minimal example of what an analysis with `seqCAT` might look like. The data analysed comes from the Gene Expression Omnibus (GEO) under accession number GSE84073, the metadata of which can be found in the `sdata.1.metadata.txt` file. (Broutier et al. 2017)

## Packages

```
# Load the seqCAT package
library("seqCAT")
library("dplyr")
library("tidyr")
library("ggplot2")
```

## Analyses

### Create profiles: WES

All the VCF files were downloaded from the GEO and analysed using `seqCAT` with the standard R implementation for creation of SNV profiles (a faster implementation in Python is also available; please see the `seqCAT` vignette for more information).

```
# List files in directory
files <- list.files(".", full.names = TRUE)
vcfs <- grep(".vcf", files, value = TRUE)

# Create profiles for each VCF in input directory
for (current_vcf in vcfs) {

    # Get current sample
    current_sample <- strsplit(current_vcf, "\\.")[[1]][1]

    # Set current output
    current_output <- paste0("profiles/wes/raw/", current_sample,
                             ".profile.txt")

    # Create profile for current input file
    create_profile(current_vcf, current_sample, current_output, python = FALSE)
}
```

Version `1.3.1` introduces a convenience-function that is equivalent to the code above, and can be used very simply:

```
create_profiles(vcf_dir = ".")
```

## Read profiles: WES

Similarly to the convenience function for creating SNV profiles, version `1.3.1` also introduces another wrapper-function for reading of multiple SNV profiles:

```
profile_list_wes_raw <- read_profiles("profiles/wes/raw")
```

This is equivalent to the code below, which is included for compatibility with version `1.2.0`, the release-version at the time of writing.

```
# List all profile filenames
files <- list.files("profiles/wes/raw", full.names = TRUE)
profiles <- grep(".profile.txt", files, value = TRUE)
```

```
# Read all profiles
profile_list_wes_raw <- list()
for (file in profiles) {

    # Get current sample name
    current_sample <- strsplit(basename(file), "\\.")[[1]][1]

    # Read current profile
    current_profile <- read_profile(file, current_sample)

    # Append profile to profile list
    profile_list_wes_raw[[length(profile_list_wes_raw) + 1]] <- current_profile
}
```

In order to only compare protein coding SNV profiles, we performed annotation of the original downloaded WES VCF files using snpEff. Their SNV profiles were created in the same manner as above.

```
# List all annotated profile filenames
files <- list.files("profiles/wes/annotated/", full.names = TRUE)
profiles <- grep(".profile.txt", files, value = TRUE)
```

```
# Read all annotated WES profiles
profile_list_wes_annotated <- list()
profile_list_wes_coding <- list()
for (file in profiles) {

    # Get current sample name
    current_sample <- strsplit(basename(file), "\\.")[[1]][1]

    # Read current profile
    current_profile <- read_profile(file, current_sample)

    # Subset for coding variants only
    current_coding <-
        current_profile[current_profile$biotype == "protein_coding" &
                        current_profile$effect == "missense_variant", ]

    # Append profiles to profile lists
    profile_list_wes_annotated[[length(profile_list_wes_annotated) + 1]] <-
        current_profile
    profile_list_wes_coding[[length(profile_list_wes_coding) + 1]] <-
        current_coding
}
```

## Figure 1: WES heatmaps

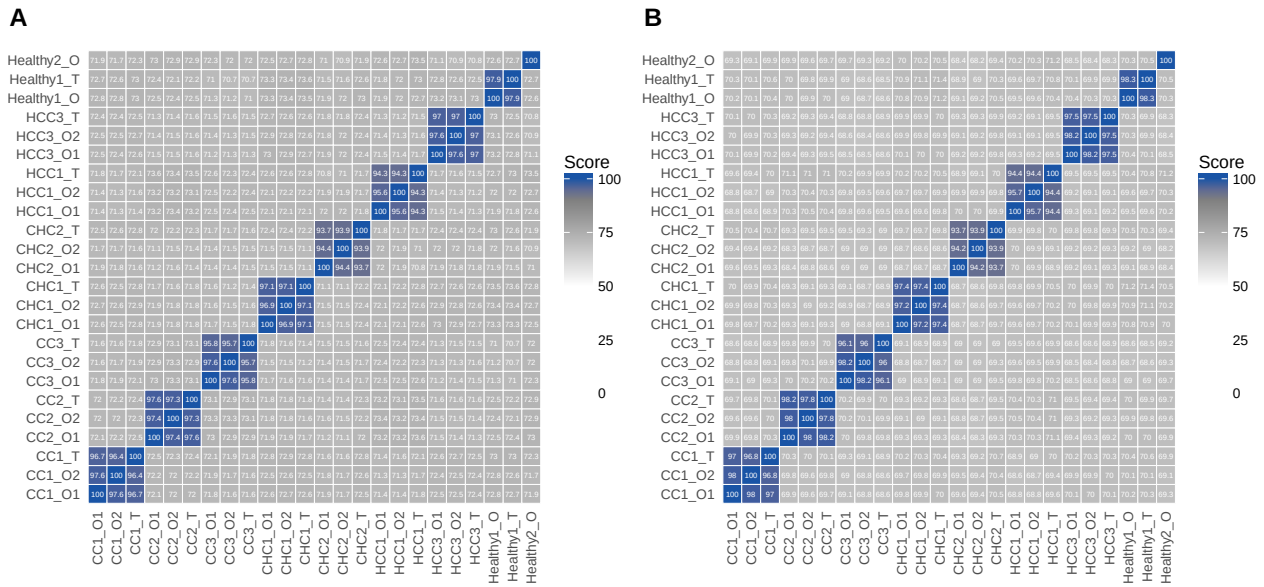
Fig 1 shows heatmaps of genetic similarities across SNVs in all samples of the GSE84073 study. It is clear that all samples originating from the same patient (whether they be tissue or organoid) cluster together, as expected. There is also very little difference between the early (01) and late (02) organoids, corroborating the conclusion drawn by the original authors that these organoids are genetically stable over long periods of time. This indicates that the organoids are good models for their respective tissue in terms of genetic variability and stability.

```
# Compare all raw profiles to each other
comparisons_wes_raw <- compare_many(profile_list_wes_raw)
similarities_wes_raw <- comparisons[[1]]

# Compare all coding profiles to each other
comparisons_wes_coding <- compare_many(profile_list_wes_coding)
similarities_wes_coding <- comparisons[[1]]
```

```
# Create and plot the heatmaps
heatmap_wes_raw <- plot_heatmap(similarities_wes_raw, annotate_size = 6,
                                cluster = FALSE)
heatmap_wes_coding <- plot_heatmap(similarities_wes_coding, annotate_size = 6,
                                   cluster = FALSE)

# Plot in grid
fig_1 <- cowplot::plot_grid(heatmap_wes_raw, heatmap_wes_coding,
                            nrow = 1, labels = c("A", "B"), label_size = 20)
```



## Table 1: WES statistics

Collate the similarity scores across all samples across the creation of the organoids (tissue vs. early) and long-term culturing (early vs. late).

```
# Function for calculating statistics from similarity dataframes
calculate_stats <- function(sim, name) {

    # Remove self-comparisons
    stats <- sim[sim$sample_1 != sim$sample_2, ]

    # Get patients
    stats <- stats %>%
        separate(sample_1, c("patient_1", "type_1"), "_", extra = "drop") %>%
        separate(sample_2, c("patient_2", "type_2"), "_", extra = "drop")

    # Keep only same-patient comparisons
    stats <- stats[stats$patient_1 == stats$patient_2, ]

    # Calculate variant calling stats
    vc_stats <- stats %>%
        group_by(patient_1) %>%
        summarise(median_overlaps=round(median(overlaps), 0),
                  median_score=round(median(similarity_score), 1))

    # Rename variant calling stats header
    names(vc_stats) <- c("patient",
                         paste0("median_overlaps.", name),
                         paste0("median_similarity.", name))

    # Get stability groups
    stats[stats$type_1 == "O" & stats$type_2 == "T", "type"] <- "T vs. O"
    stats[stats$type_1 == "T" & stats$type_2 == "O1", "type"] <- "T vs. O1"
    stats[stats$type_1 == "O1" & stats$type_2 == "T", "type"] <- "T vs. O1"
    stats[stats$type_1 == "O1" & stats$type_2 == "O2", "type"] <- "O1 vs. O2"
    stats[stats$type_1 == "O2" & stats$type_2 == "O1", "type"] <- "O1 vs. O2"
    stats <- stats[complete.cases(stats), ]

    # Add dataset and names
    stats$dataset <- strsplit(name, "_")[[1]][1]
    stats$subset <- strsplit(name, "_")[[1]][2]

    # Remove unneeded columns
    stats <- stats[c("type", "dataset", "subset", "similarity_score")]

    # Return statistics
    return(list(vc_stats, stats))
}

# Initialise statistics dataframes
patients <- c("CC1", "CC2", "CC3", "CHC1", "CHC2", "HCC1", "HCC3", "Healthy1")
variant_calling_stats <- data.frame(patient = patients)
similarity_stats <- data.frame(patient = patients)
stability_stats <- data.frame(type             = character(),
```

```
                              dataset          = character(),
                              subset           = character(),
                              similarity_score = numeric())

# Loop over each similarity dataframe
similarities_list <- list(similarities_wes_raw, similarities_wes_coding)
names_list <- list("WES_All", "WES_Coding")
for (nn in c(1:length(names_list))) {

    # Calculate stats
    all_stats <- calculate_stats(similarities_list[[nn]], names_list[[nn]])

    # Merge VC stats with tables
    vc_stats <- all_stats[[1]]
    variant_calling_stats <- merge(variant_calling_stats, vc_stats[c(1, 2)],
                                   by = "patient")
    similarity_stats <- merge(similarity_stats, vc_stats[c(1, 3)],
                              by = "patient")

    # Merge stability stats with dataframe
    stab_stats <- all_stats[[2]]
    stability_stats <- rbind(stability_stats, stab_stats)
}

# Merge and print tables
stats <- cbind(variant_calling_stats, similarity_stats[c(2,3)])
stats
```

```
##    patient median_overlaps.WES_All median_overlaps.WES_Coding
## 1      CC1                  153816                      111977
## 2      CC2                  137261                       97344
## 3      CC3                  122577                       87604
## 4     CHC1                  153589                      112011
## 5     CHC2                  132805                       95203
## 6     HCC1                  142389                      104087
## 7     HCC3                  130186                       92613
## 8 Healthy1                  155949                      113592
##   median_similarity.WES_All median_similarity.WES_Coding
## 1                      96.7                         97.0
## 2                      97.4                         98.0
## 3                      95.8                         96.1
## 4                      97.1                         97.4
## 5                      93.9                         93.9
## 6                      94.3                         94.4
## 7                      97.0                         97.5
## 8                      97.9                         98.3
```

## Figure 2A: genetic stability of organoids

Genetic stability of tissue vs. early organoid (initial) and early vs. late (long-term).

```
# Order comparisons
stability_stats <- stability_stats[stability_stats$type == "T vs. O1" |
                                   stability_stats$type == "O1 vs. O2", ]
stability_stats$type <- factor(stability_stats$type,
                               levels = c("T vs. O1", "O1 vs. O2"))

# Plot stabilities
fig_2a <- ggplot(stability_stats,
                         aes(x = subset, y = similarity_score, fill = type)) +
    stat_boxplot(geom     = "errorbar",
                 colour   = "#4d4d4d",
                 position = position_dodge(width = 0.75),
                 width    = 0.3) +
    geom_boxplot() +
    geom_point(aes(group = type),
               size      = 2,
               position  = position_jitterdodge(dodge.width  = 0.75,
                                                jitter.width = 0.2)) +
    theme_classic() +
    scale_fill_manual(values = c("#1954a6", "#a6c6f2")) +
    labs(x    = "Variant subset",
         y    = "Similarity score",
         fill = "Comparison")

# Statistical testing
stability_stats %>%
    group_by(subset, type) %>%
    mutate(i = row_number()) %>%
    summarise(sim = list(similarity_score)) %>%
    spread(type, sim) %>%
    group_by(subset) %>%
    mutate(p_value = t.test(unlist(`T vs. O1`), unlist(`O1 vs. O2`))$p.value)
```

```
## # A tibble: 2 x 4
## # Groups:   subset [2]
##   subset `T vs. O1` `O1 vs. O2` p_value
##   <chr>  <list>     <list>        <dbl>
## 1 All    <dbl [7]>  <dbl [7]>     0.363
## 2 Coding <dbl [7]>  <dbl [7]>     0.408
```

## Figure 2B: known variants in organoids

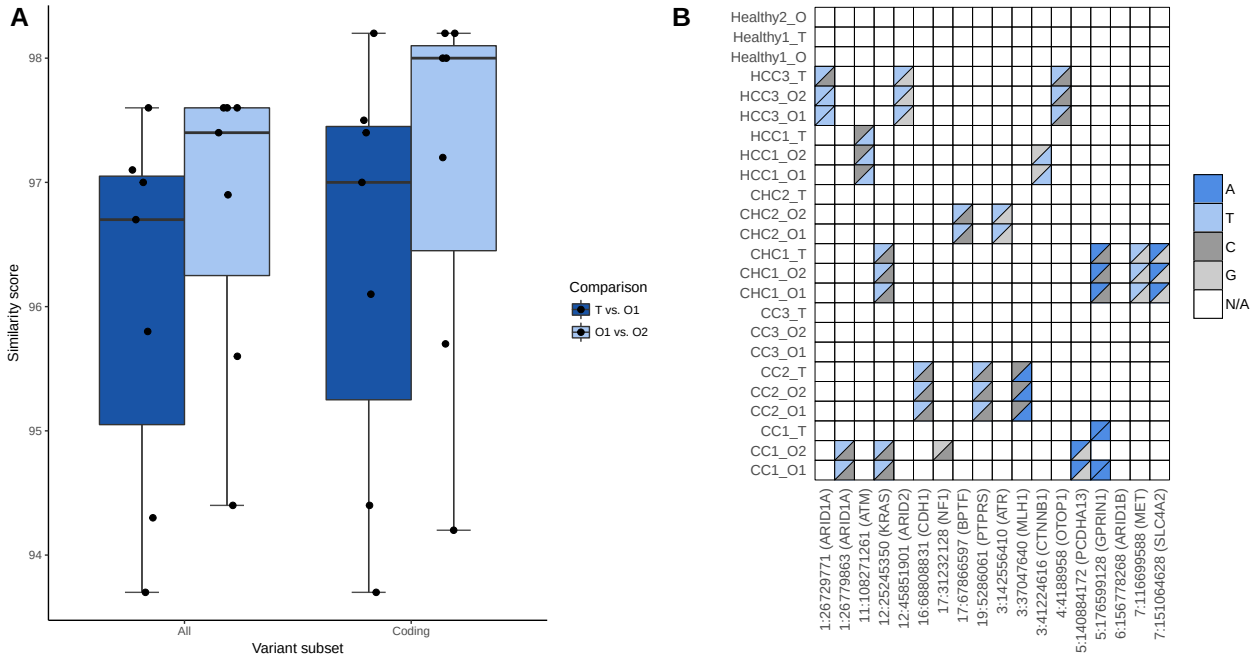
Fig 2B shows the genotypes of all the samples in the original study across the known variants as listed by the authors in their table S13. It is evident that the healthy controls harbour no mutations known to be present in liver cancer, as expected. Ten out of 19 known variants show a perfect match between the tissue/organoid triplets, whereas eight out of the nine mismatches show missing variants for the tissue. These variants should thus be investigated further, to determine why the variant calling pipeline used resulted in those variants being present in the organoids but not the corresponding tissue. Indeed, the original authors analysed their original BAM files and the reads for those specific mutations; please see (Broutier et al. 2017) for details. The last mismatching variant is for the GPRIN1 gene in the CC1 sample, which is not listed as containing that mutation in the original publication. The GPRIN1 mutation in CC1 would thus need to be further analysed in order to fully investigate its possible phenotypic effect, if any.

```
# Read list of known variants
known_variants <- read.table("sdata.2.known_variants.txt", sep = "\t",
                             header = TRUE, stringsAsFactors = FALSE)

# Find known variants in SNV profiles
known_list_wes <- list_variants(profile_list_wes_raw, known_variants)

# Add row names and sort
row.names(known_list_wes) <- paste0(known_list_wes$chr, ":",
                                    known_list_wes$pos, " (",
                                    known_list_wes$gene, ")")
known_list_wes <- known_list_wes[order(known_list_wes$chr), ]
known_list_wes <- known_list_wes[setdiff(names(known_list_wes),
                                         c("chr", "pos", "gene"))]

# Plot known variants
fig_2b <- plot_variant_list(known_list_wes)
```



## Figure 3: impact distributions

Impact distribution of WES comparisons.

```
# Get header for comparisons
comparisons_header <- names(comparisons_wes_coding[[2]][[2]])
comparisons_header <- gsub("HCC1_T", "sample_1", comparisons_header)
comparisons_header <- gsub("HCC3_T", "sample_2", comparisons_header)

# Initialise file with header
wes_collated_file <- "collated.comparisons.wes.txt"
cat(c(comparisons_header, "patient", "cell_1", "cell_2"), sep = "\t",
    file = wes_collated_file)
cat("\n", file = wes_collated_file, append = TRUE)

# Collate all profiles into a single file for downstream analysis
for (current in comparisons_wes_coding[[2]]) {

    # Get samples
    sample_1 <- unique(current$sample_1)
    sample_2 <- unique(current$sample_2)

    # Skip same-sample comparisons
    if (sample_1 == sample_2) {
        next
    }

    # Get patients
    patient_1 <- strsplit(sample_1, "_")[[1]][1]
    patient_2 <- strsplit(sample_2, "_")[[1]][1]

    # Keep only same-patient comparisons
    if (patient_1 != patient_2) {
        next
    }

    # Add patient and cell type
    current$patient <- patient_1
    current$cell_1 <- strsplit(sample_1, "_")[[1]][2]
    current$cell_2 <- strsplit(sample_2, "_")[[1]][2]

    # Append to file
    write.table(current, wes_collated_file, sep = '\t', row.names = FALSE,
                col.names = FALSE, append = TRUE, quote = FALSE)
}
```

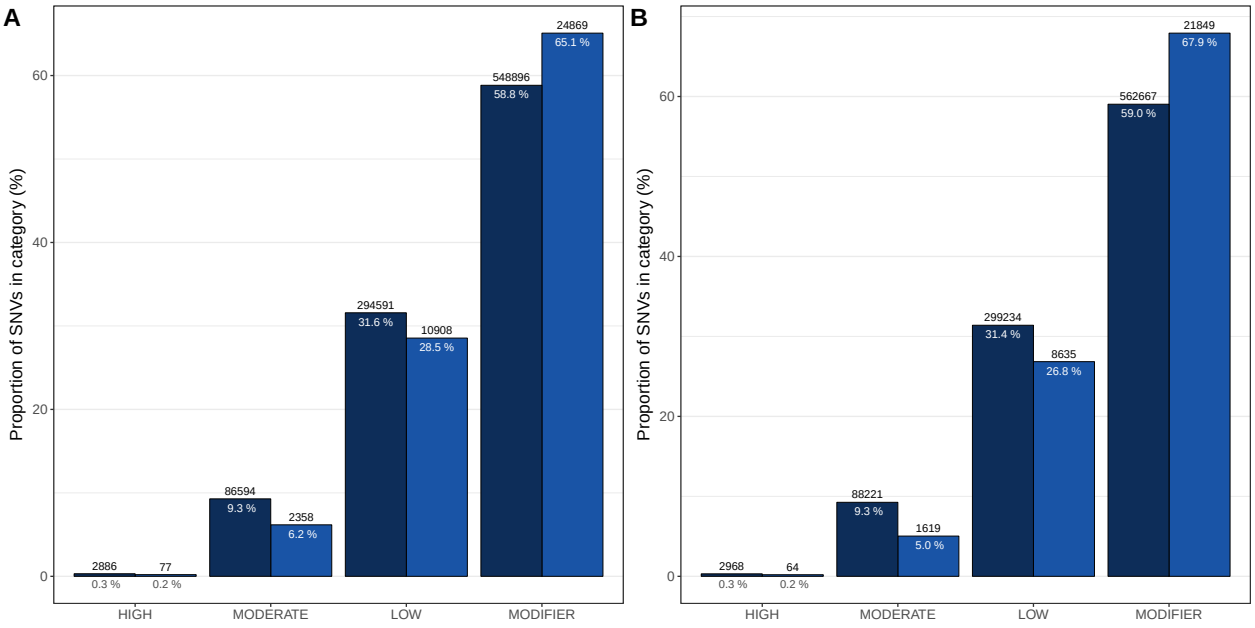
```
# Read collated comparison data
comp <- read.table("collated.comparisons.wes.txt",
                   sep = "\t", quote = "", header = TRUE,
                   stringsAsFactors = FALSE)
```

```
# Add comparison type to data
comp[comp$cell_1 == "T" & comp$cell_2 == "O1", "type"] <- "T vs. O1"
comp[comp$cell_1 == "O1" & comp$cell_2 == "T", "type"] <- "T vs. O1"
comp[comp$cell_1 == "O1" & comp$cell_2 == "O2", "type"] <- "O1 vs. O2"
```

```r
comp[comp$cell_1 == "O2" & comp$cell_2 == "O1", "type"] <- "O1 vs. O2"
comp <- comp[!is.na(comp$type), ]

# Plot impact distributions
impacts_wes_1 <- plot_impacts(comp[comp$type == "T vs. O1", ], legend = FALSE)
impacts_wes_2 <- plot_impacts(comp[comp$type == "O1 vs. O2", ], legend = FALSE)

# Grid
fig_3 <- cowplot::plot_grid(impacts_wes_1, impacts_wes_2,
                            labels     = c("A", "B"),
                            label_size = 20,
                            hjust      = 0,
                            nrow       = 1)
```



### WES genes for enrichment analyses

Get gene sets to perform enrichment analyses on.

```
# Get mismatching variants only
mismatches <- comp[comp$match == "mismatch", ]

# Loop over comparison types
for (type in c("T vs. O1", "O1 vs. O2")) {

    # Subset for impacts and transition
    genes <- sort(unique(mismatches[(mismatches$impact == "HIGH" |
                                     mismatches$impact == "MODERATE") &
                                    mismatches$type == type, "ENSGID"]))

    # Set filenames
    out <- paste0("genes.", gsub(" vs. ", "_", type), ".txt")

    # Save to file
    cat(genes, file = out, sep = "\n")
}
```

## Figure 4: WES vs. RNA-seq

Comparisons between WES and RNA-seq variant callings on a per-sample basis.

```
# List all annotated profile filenames
files <- list.files("profiles/rna_seq", full.names = TRUE)
profiles <- grep(".profile.txt", files, value = TRUE)
```

```
# Read all annotated WES profiles
profile_list_rna <- list()
for (file in profiles) {

    # Get current sample
    sample <- basename(file)
    sample <- paste0(strsplit(sample, "\\.")[[1]][1], "_RNA")

    # Read current profile
    current_profile <- read_profile(file, sample)

    # Append profiles to profile list
    profile_list_rna[[length(profile_list_rna) + 1]] <- current_profile
}

# Append to WES profile list
profile_list_wes_and_rna <- c(profile_list_wes_annotated, profile_list_rna)
```

```
similarities_wes_vs_rna <- compare_many(profile_list_wes_and_rna)[[1]]
```

```
fig_4 <- plot_heatmap(similarities_wes_vs_rna, annotate = FALSE,
                      cluster = FALSE)
```

```
ggsave("../figures/figure_4.png", fig_4, dpi = 300, width = 11, height = 8.5)
fig_4
```

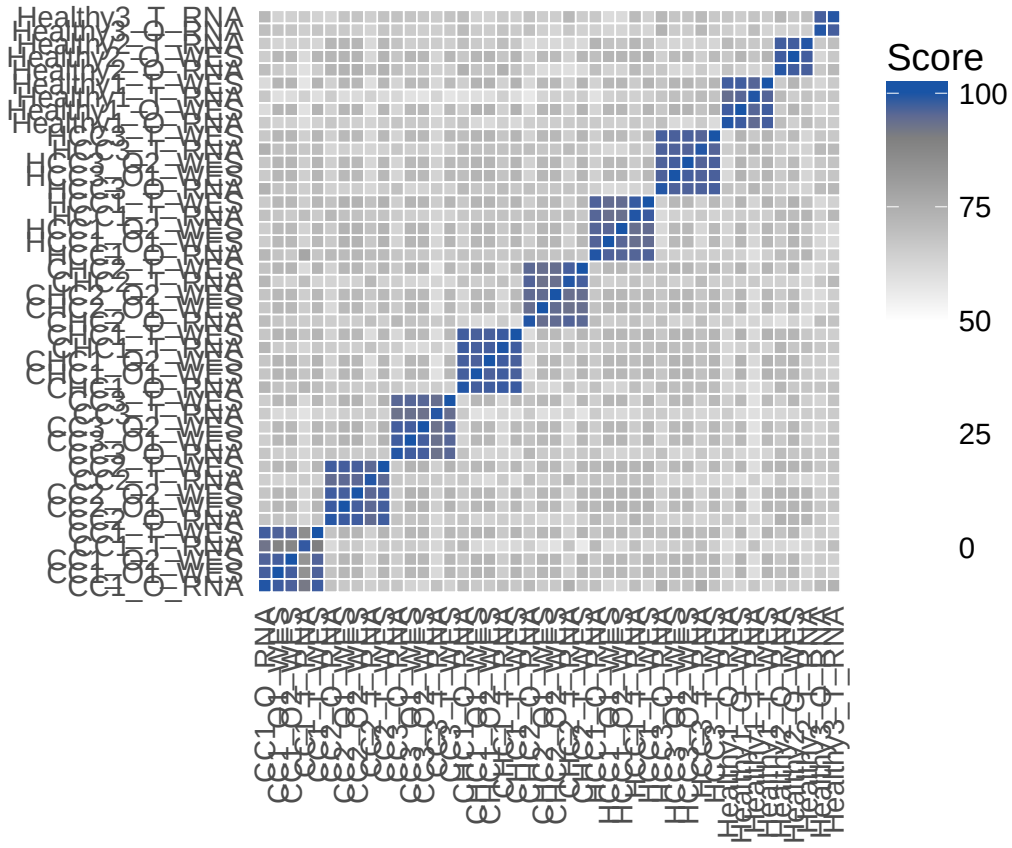



**Table 2: WES vs. RNA-seq**

```
# List the RNA-seq profiles
rna_files <- list.files("profiles/rna-seq", full.names = TRUE)
rna_profiles <- grep(".profile.txt", files, value = TRUE)
```

```
# Initialise results dataframe
wes_vs_rna <- data.frame(sample_1         = character(),
                         sample_2         = character(),
                         overlaps         = numeric(),
                         concordance      = numeric(),
                         similarity_score = numeric())

# Read and compare each matched sample pair
for (wes_profile in profile_list_wes_annotated) {

    # Get WES sample
    wes_sample <- unique(wes_profile$sample)

    # Loop over RNA-seq files
    for (rna_file in rna_profiles) {

        # Read RNA-seq profile
        rna_sample <- grep("profile.txt", strsplit(rna_file, "/")[[1]],
                           value = TRUE)
```

```
        rna_sample <- strsplit(rna_sample, "\\.")[[1]][1]

        # Compare to WES profile for same samples
        if (gsub("O1", "O", wes_sample) == rna_sample |
            gsub("O2", "O", wes_sample) == rna_sample) {

            # Read RNA profile
            rna_profile <- read_profile(rna_file, paste0(rna_sample, "_RNA"))

            # Compare with WES profile
            comp <- compare_profiles(wes_profile, rna_profile)

            # Calculate similarities and add to results dataframe
            sim <- calculate_similarity(comp)
            wes_vs_rna <- rbind(wes_vs_rna, sim)
        }
    }
}

# Add cell type and patient to results dataframe
wes_vs_rna$sample_1 <- paste0(wes_vs_rna$sample_1, "_WES")
wes_vs_rna[grep("T", wes_vs_rna$sample_1), "cell"] <- "T"
wes_vs_rna[grep("O", wes_vs_rna$sample_1), "cell"] <- "O"
wes_vs_rna[grep("O1", wes_vs_rna$sample_1), "cell"] <- "O"
wes_vs_rna[grep("O2", wes_vs_rna$sample_1), "cell"] <- "O"
wes_vs_rna <- wes_vs_rna %>%
    separate(sample_1, by = "_", into = "patient", extra = "drop",
             remove = FALSE) %>%
    select(patient, sample_1, sample_2, cell, overlaps, concordance)
wes_vs_rna
```

```
##     patient       sample_1       sample_2 cell overlaps concordance
## 1       CC1     CC1_O1_WES      CC1_O_RNA    O     3744        97.5
## 2       CC1     CC1_O2_WES      CC1_O_RNA    O     3768        96.9
## 3       CC1      CC1_T_WES      CC1_T_RNA    T      915        81.2
## 4       CC2     CC2_O1_WES      CC2_O_RNA    O      631        99.0
## 5       CC2     CC2_O2_WES      CC2_O_RNA    O      609        98.9
## 6       CC2      CC2_T_WES      CC2_T_RNA    T      508        96.1
## 7       CC3     CC3_O1_WES      CC3_O_RNA    O      764        98.6
## 8       CC3     CC3_O2_WES      CC3_O_RNA    O      718        98.2
## 9       CC3      CC3_T_WES      CC3_T_RNA    T      521        94.8
## 10     CHC1    CHC1_O1_WES     CHC1_O_RNA    O     3164        98.0
## 11     CHC1    CHC1_O2_WES     CHC1_O_RNA    O     3189        97.9
## 12     CHC1     CHC1_T_WES     CHC1_T_RNA    T     3061        97.8
## 13     CHC2    CHC2_O1_WES     CHC2_O_RNA    O      920        94.3
## 14     CHC2    CHC2_O2_WES     CHC2_O_RNA    O      923        95.1
## 15     CHC2     CHC2_T_WES     CHC2_T_RNA    T      718        98.1
## 16     HCC1    HCC1_O1_WES     HCC1_O_RNA    O     1874        96.5
## 17     HCC1    HCC1_O2_WES     HCC1_O_RNA    O     1872        96.4
## 18     HCC1     HCC1_T_WES     HCC1_T_RNA    T     1787        98.8
## 19     HCC3    HCC3_O1_WES     HCC3_O_RNA    O      745        97.4
## 20     HCC3    HCC3_O2_WES     HCC3_O_RNA    O      751        97.3
## 21     HCC3     HCC3_T_WES     HCC3_T_RNA    T      580        96.9
## 22 Healthy1 Healthy1_O_WES Healthy1_O_RNA    O     1846        98.6
```

```
## 23 Healthy1 Healthy1_T_WES Healthy1_T_RNA    T     1367        91.0
## 24 Healthy2 Healthy2_O_WES Healthy2_O_RNA    O     1367        98.7
```

```
# Overall statistics
median(wes_vs_rna$concordance)
```

```
## [1] 97.45
```

```
min(wes_vs_rna$concordance)
```

```
## [1] 81.2
```

```
max(wes_vs_rna$concordance)
```

```
## [1] 99
```

```
# Per-cell type statistics
wes_vs_rna %>%
    group_by(cell) %>%
    summarise(median_concordance = median(concordance),
              median_overlaps = median(overlaps))
```

```
## # A tibble: 2 x 3
##   cell  median_concordance median_overlaps
##   <chr>              <dbl>           <dbl>
## 1 O                   97.7           1145
## 2 T                   96.5            816.
```

```
# Per-patient statistics
wes_vs_rna %>%
    group_by(patient) %>%
    summarise(median_patient_concordance = median(concordance),
              median_overlaps = median(overlaps))
```

```
## # A tibble: 9 x 3
##   patient  median_patient_concordance median_overlaps
##   <chr>                         <dbl>           <dbl>
## 1 CC1                            96.9           3744
## 2 CC2                            98.9            609
## 3 CC3                            98.2            718
## 4 CHC1                           97.9           3164
## 5 CHC2                           95.1            920
## 6 HCC1                           96.5           1872
## 7 HCC3                           97.3            745
## 8 Healthy1                       94.8           1606.
## 9 Healthy2                       98.7           1367
```

## Session info

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.4
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2              ggplot2_2.2.1
##  [3] tidyr_0.8.1                 dplyr_0.7.5
##  [5] seqCAT_1.3.1                VariantAnnotation_1.27.1
##  [7] Rsamtools_1.33.0            Biostrings_2.49.0
##  [9] XVector_0.21.1              SummarizedExperiment_1.11.3
## [11] DelayedArray_0.7.0          BiocParallel_1.15.0
## [13] matrixStats_0.53.1          Biobase_2.41.0
## [15] GenomicRanges_1.33.5        GenomeInfoDb_1.17.1
## [17] IRanges_2.15.13             S4Vectors_0.19.5
## [19] BiocGenerics_0.27.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.17             lattice_0.20-35
##  [3] prettyunits_1.0.2        utf8_1.1.3
##  [5] assertthat_0.2.0         rprojroot_1.3-2
##  [7] digest_0.6.15            plyr_1.8.4
##  [9] R6_2.2.2                 backports_1.1.2
## [11] RSQLite_2.1.1            evaluate_0.10.1
## [13] httr_1.3.1               pillar_1.2.2
## [15] zlibbioc_1.27.0          rlang_0.2.0
## [17] GenomicFeatures_1.33.0   progress_1.1.2
## [19] lazyeval_0.2.1           blob_1.1.1
## [21] Matrix_1.2-14            rmarkdown_1.9
## [23] labeling_0.3             stringr_1.3.0
## [25] RCurl_1.95-4.10          bit_1.1-12
## [27] biomaRt_2.37.0           munsell_0.4.3
## [29] compiler_3.5.0           rtracklayer_1.41.2
## [31] pkgconfig_2.0.1          htmltools_0.3.6
## [33] tidyselect_0.2.4         tibble_1.4.2
## [35] GenomeInfoDbData_1.1.0   XML_3.98-1.11
## [37] crayon_1.3.4             GenomicAlignments_1.17.0
## [39] bitops_1.0-6             grid_3.5.0
## [41] gtable_0.2.0             DBI_1.0.0
## [43] magrittr_1.5             scales_0.5.0
## [45] cli_1.0.0                stringi_1.2.2
## [47] cowplot_0.9.2            tools_3.5.0
```

```
## [49] bit64_0.9-7              BSgenome_1.49.0
## [51] glue_1.2.0               purrr_0.2.4
## [53] yaml_2.1.19              AnnotationDbi_1.43.1
## [55] colorspace_1.3-2         memoise_1.1.0
## [57] knitr_1.20               bindr_0.1.1
```

# References

Broutier, Laura, Gianmarco Mastrogiovanni, Monique Ma Verstegen, Hayley E Francies, Lena Morrill Gavarró, Charles R Bradshaw, George E Allen, et al. 2017. "Human primary liver cancer-derived organoid cultures for disease modeling and drug screening." *Nature Medicine*, November.