Lung Metastasis Dataset Construction

Nie Chenxi

5/17/2022

Lung Metastasis Datset Construction and pre MetaMHN sample workflow

Find Patients with metastasis

In this section, I created a pipeline to extract patients with ≥ 2 samples and the samples are taken in a way that at least one from metastasis, and at least one from primary tumour. In the end, I would generate a list of sample ids.

```
clinical_dataset_2017 <- file("/Users/chenxi/Library/CloudStorage/OneDrive-ETHZurich/second-year-master/
clinical_dataset <- read.csv(clinical_dataset_2017, sep = "\t")
close(clinical_dataset_2017)

metastasis_dataset <- subset(clinical_dataset, Number.of.Samples.Per.Patient >= 2)


# check if they all have primary sample and metastasis samples
patient_Id_selected <- c()
patient_Id <- unique(metastasis_dataset$Patient.ID)
for(patient_id in patient_Id) {
  metastasis_dataset_patient_id <-subset(metastasis_dataset, Patient.ID == patient_id)
  sample_status <- metastasis_dataset_patient_id$Sample.Type
  is_metastasis <- "Metastasis"%in%sample_status
  is_primary <- "Primary"%in%sample_status


  if (is_metastasis && is_primary) {
    patient_Id_selected <- c(patient_Id_selected, patient_id)
  }
}
```

Get the Gene Panel

In this section, I set out to find out which gene pannel they used in the dataset.

```
gene_panels <- file("../../Data/msk_impact_2017/data_gene_panel_matrix.txt", "r")
gene_panels <- read.csv(gene_panels, sep = "\t")
gene_panels <- union(unique(gene_panels$mutations), unique(gene_panels$cna))
gene_panels
```

```
## [1] "IMPACT341" "IMPACT410"
```

As it turns out, in this project, they only used “IMPACT 341” and “IMPACT 410” gene panels. Now I would head out to this link to download the two gene panels.

Get the location for each gene

After we have the two gene panels, we need to headout to HGNC symnbol checker to get their location. In the website, you can copy and paste the gene list, and then it would tell you the location of the genes in that list.

Here I first combine the two gene panels together and generate a gene list file so that I can upload it to the symbol checker.

```
gene_list_341 <- n.readLines("../../Data/msk_impact_2017/data_gene_panel_impact341.txt", n = 1, skip = 1
gene_list_341 <- strsplit(gene_list_341, split = ":")[[1]]
gene_list_341 <- trimws(gene_list_341[2])
gene_list_341 <- strsplit(gene_list_341, split = "\t")[[1]]

gene_list_410 <- n.readLines("../../Data/msk_impact_2017/data_gene_panel_impact410.txt", n = 1, skip = 1
gene_list_410 <- strsplit(gene_list_410, split = ":")[[1]]
gene_list_410 <- trimws(gene_list_410[2])
gene_list_410 <- strsplit(gene_list_410, split = "\t")[[1]]

gene_list_union <- union(gene_list_341, gene_list_410)

dir.create("Lung") # create a directory called "Lung"
```

```
## Warning in dir.create("Lung"): 'Lung' already exists
```

```
fileConn<-file("Lung/gene_list_union.txt")
writeLines(gene_list_union, fileConn)
close(fileConn)
```

Now we can simply upload the gene_list_union.txt file to the HGNC symnbol checker and download the result. Here I would simply put the hgnc-symbol-checker.csv to the "Lung" folder just created.

Split the genes that have multiple places

Now, the HGNC symbol checker's way to handle one gene which has two places is that it writes in the location column something like this: "2p23.2-p23.1". But In future piplelines, we expect something like this "2p23.2" in the first column and "2p23.1" in the second column. Here I do it manually as the gene panel is not that large.

In the future, I would come up with some code to do this (probably using the regular expressions)

Extract Lung Cancer Patient

In this section, I take the patient ID from section 1 and extract those Lung Cancer patient (to be precise, those patients shose primary tumour site is "Lung")

```
clinical_dataset_2017 <- file("/Users/chenxi/Library/CloudStorage/OneDrive-ETHZurich/second-year-master/
clinical_dataset <- read.csv(clinical_dataset_2017, sep = "\t")
close(clinical_dataset_2017)

Lung_metastasis <- subset(clinical_dataset, Primary.Tumor.Site == "Lung")


# check if they all have primary sample and metastasis samples
patient_Id_selected <- c()
patient_Id <- unique(Lung_metastasis$Patient.ID)
for(patient_id in patient_Id) {
  metastasis_dataset_patient_id <-subset(Lung_metastasis, Patient.ID == patient_id)
```

```
  sample_status <- metastasis_dataset_patient_id$Sample.Type
  is_metastasis <- "Metastasis"%in%sample_status
  is_primary <- "Primary"%in%sample_status


  if (is_metastasis && is_primary) {
    patient_Id_selected <- c(patient_Id_selected, patient_id)
  }
}

Lung_metastasis <-subset(Lung_metastasis, Patient.ID %in% patient_Id_selected)
```

Extract seg file for GISTIC

In this section, I take the Lung metastasis patient id, and extract from the seg file a lung specific seg file for GISTIC 2

```
data_cna_hg19 <- file("/Users/chenxi/Library/CloudStorage/OneDrive-ETHZurich/second-year-master/Lab_rota
data_cna_hg19_df<- read.csv(data_cna_hg19, sep = "\t")
close(data_cna_hg19)

sample_id_selected <- Lung_metastasis$Sample.ID
lung_data_cna_hg_19 <- subset(data_cna_hg19_df, ID %in% sample_id_selected)


write.table(lung_data_cna_hg_19, file = "Lung/data_cna_hg19_Lung.seg", sep = "\t", row.names = FALSE, q
```

Running GISTIC 2

Now, I would take the seg file I just created and run GISTIC 2 with the following command:

```
./gistic2 -b Lung/ -seg Lung/data_cna_hg19_Lung.seg \
          -refgene /path/to/hg19/mat_file \
          -genegistic 1 -smallmem 0 -rx 0 -broad 1 -brlen 0.7 -conf 0.99 -armpeel 1 -savegene 1 \
          -gcm extreme -v 30 -maxseg 46000 -ta 0.3 -td 0.3 -cap 1.5 -js 4
```

The command is suggested from this github repository.

Side note: Installing GISTIC 2 I tried to install GISTIC 2 on macOS and my suggestion would be, don't do it. Although gistic 2 is a matlab based program and matlab program, in theory, should run on macOS just fine, there are a lot of problems that I faced during installation.

I suggest getting your hands on a linux machine (desktop, laptop, server etc.) and get this wonderful tool called bioconda which is based on the well-known tool conda. The installation of anaconda and bioconda should be widely available on google.

After you successfully installed anaconda and bioconda, you can simply install GISTIC 2 (on LINUX machine only) by this command

```
conda install -c hcc gistic2
```

One common issue while installing anaconda is that after installation, you have the 'conda command not found'. In that case, you can check out this page by diehard from linuxpip

Side note 2: hg19 mat file The hg19 mat file is available at GISTIC 2's github page. Here you would find also hg16, hg17, hg18, hg19.mat and hg38. You can also check out their source code etc.

Getting the binarised genome of the Lung cancer dataset

In this section, I would step by step describe how to convert the GISTIC2's output to a binarised genome.

Step 1: Get the top 40% regions by Q value

This section, we set out to find the top 40% regions in "all_lesion.conf_99.txt".

```
cancer_type = "Lung"

step_1_get_top_40_by_Q_value <- function (cancer_type){
  selection_threshold = 0.4

  all_lesions_path <- paste(cancer_type, "/all_lesions.conf_99.txt", sep = "")
  flag <- T # a flag indicating if this function finishes successfully

  if(!file.exists(all_lesions_path)) {
    writeLines(paste(cancer_type, "does not have all_leions_file"))
    Missing_cancer_type <- cancer_type
    flag <- F
    return_list <- list(flag, Missing_cacner_type, NaN, NaN, NaN, NaN)
    return(flag, Missing_cancer_type)
  }

  all_lesions.conf_99 <- file(all_lesions_path, "r")

  first_line = readLines(all_lesions.conf_99, n = 1) # discard the first line
  first_line_split <- strsplit(first_line, "\t")
  number_of_samples <- length(first_line_split[[1]]) - 9 # samples only start on the 10th column

  peak_q_values <- c(); delete_q_values <- c();
  peak_names <- c(); delete_names <- c();peak_cytoband <- c(); delete_cytoband <- c()

  # read line by line
  while(TRUE) {
    line = readLines(all_lesions.conf_99, n = 1)
    line_split <- strsplit(line, "\t")
    if (line_split[[1]][9] == "Actual Copy Change Given" ) {
      break
    } # we only need the data from section 1 (for sections
    # please refer to manual of GISTIC 2 from GenePattern)

    amp_or_del <- grepl("Amplification", line_split[[1]][1])
    if(amp_or_del) {
      peak_q_values <- c(peak_q_values, as.numeric(line_split[[1]][6])) # the 6th column is the q value
      peak_names <- c(peak_names, line_split[[1]][1])
      peak_cytoband <- c(peak_cytoband, line_split[[1]][2])
    } else {
      delete_q_values <- c(delete_q_values, as.numeric(line_split[[1]][6])) # the 6th column is the q v
      delete_names <- c(delete_names, line_split[[1]][1])
      delete_cytoband <- c(delete_cytoband, line_split[[1]][2])
    }
  }

  peak_q_values <- as.data.frame(peak_q_values)
```

```
  peak_q_values <- cbind(peak_q_values, peak_names)
  peak_q_values <- cbind(peak_q_values, peak_cytoband)

  delete_q_values <- as.data.frame(delete_q_values)
  delete_q_values <- cbind(delete_q_values, delete_names)
  delete_q_values <- cbind(delete_q_values, delete_cytoband)

  close(all_lesions.conf_99)


  peak_nrow <- nrow(peak_q_values); delete_nrow <- nrow(delete_q_values);
  sum <- peak_nrow + delete_nrow; after_selection = floor(sum * selection_threshold)

  peak_top <- floor(after_selection * peak_nrow / sum); delete_top <- after_selection - peak_top;

  peak_q_values_order <- peak_q_values[order(peak_q_values$peak_q_values), ]
  peak_q_values_top <- peak_q_values_order[1:peak_top, ]
  peak_amplification_events <- peak_q_values_top$peak_names

  delete_q_values_order <- delete_q_values[order(delete_q_values$delete_q_values), ]
  delete_q_values_top <- delete_q_values_order[1 : delete_top, ]
  delete_events <- delete_q_values_top$delete_name

  return_list <- list(flag, cancer_type, peak_q_values_top, delete_q_values_top, number_of_samples, firs
  return(return_list)
}

return_list <- step_1_get_top_40_by_Q_value(cancer_type = cancer_type)
```

Step 2: Convert to Binarised Genotype

In this section, we take the top 40% region that GISTIC finds and convert them to Binarised Genotype.

In the code below, I loop through every sample in the all.lesions file and for every selected peak regions, look for

```
gene_union_HGNC_annotation <- read.csv("/Users/chenxi/Library/CloudStorage/OneDrive-ETHZurich/second-yea

# A function to extract genes at a cytoband in "gene_union_HGNC_annotation"
get_genes_at_cytoband <- function(cytoband) {
  genes_in_cytoband <- gene_union_HGNC_annotation$Input[gene_union_HGNC_annotation$Location == cytoband]
  return(genes_in_cytoband)
}

step_2_binarised_genome_construction <- function(return_list_from_step_1){
  peak_q_values_top <- return_list_from_step_1[[3]]
  delete_q_values_top <- return_list_from_step_1[[4]]
  number_of_samples <- return_list_from_step_1[[5]]
  first_line_split <- return_list_from_step_1[[6]]
  cancer_type <- return_list_from_step_1[[2]]
  all_lesions_path <- paste(cancer_type, "/all_lesions.conf_99.txt", sep = "")

  genotype_amplification <- data.frame(matrix(ncol = number_of_samples,nrow = 0))
  genotype_deletion <- data.frame(matrix(ncol = number_of_samples, nrow = 0))
```

```
colnames(genotype_amplification) = colnames(genotype_deletion) <- first_line_split[[1]][c(10 : (10 + n

all_lesions.conf_99 = file(all_lesions_path, "r")
counter_amplification <- 0
missing_amplification <- c()
counter_deletion <- 0
missing_deletion <- c()

while(TRUE) {
  line = readLines(all_lesions.conf_99, n = 1)

  if(length(line) == 0) {
    break
  }

  line_split <- strsplit(line, "\t")
  if (line_split[[1]][9] != "Actual Copy Change Given" ) {
    next
  }

  amp_or_del <- grepl("Amplification", line_split[[1]][1])
  if(amp_or_del) {
    # only use top 40%
    if(!(str_trim(line_split[[1]][2]) %in% str_trim(peak_q_values_top$peak_cytoband))) {
      next
    }

    # Amplification
    cytoband <- str_trim(line_split[[1]][2])
    gene_list <- get_genes_at_cytoband(cytoband)
    number_of_genes <- length(gene_list)

    if (number_of_genes == 0) {
      counter_amplification <- counter_amplification + 1
      missing_amplification <- c(missing_amplification, cytoband)
      next
    }

    genotype_per_cytoband <- data.frame(matrix(nrow = number_of_genes, ncol = number_of_samples))
    colnames(genotype_per_cytoband) <- first_line_split[[1]][c(10 : (10 + number_of_samples - 1))]
    rownames(genotype_per_cytoband) <- make.names(gene_list, unique=TRUE)
    i <- 1
    for(actual_copy_number in line_split[[1]][c(10:(10 + number_of_samples-1))]) {
      if (as.numeric(actual_copy_number) > 0.3) {
        genotype_per_cytoband[, i] <- rep(1, number_of_genes)
        i <- i + 1
      } else {
        genotype_per_cytoband[, i] <- rep(0, number_of_genes)
        i <- i + 1
      }
    }

    # Deal with identical gene names in different cytoband
```

```
    intersect_genes <- intersect(rownames(genotype_amplification), rownames(genotype_per_cytoband))
    if(length(intersect_genes) == 0){
      genotype_amplification <- rbind(genotype_amplification, genotype_per_cytoband)
    } else {
      for (gene in intersect_genes) {
        genotype_amplification[gene, ] <- as.integer(genotype_amplification[gene, ] | genotype_per_cy
      }
    }
  } else {
    if (!(str_trim(line_split[[1]][2]) %in% str_trim(delete_q_values_top$delete_cytoband))) {
      next
    }

    # Deletion
    cytoband <- str_trim(line_split[[1]][2])
    gene_list <- get_genes_at_cytoband(cytoband)
    gene_list <- unique(gene_list)
    number_of_genes <- length(gene_list)
    if (number_of_genes == 0) {
      counter_deletion <- counter_deletion + 1
      missing_deletion <- c(missing_deletion, cytoband)
      next
    }
    genotype_per_cytoband <- data.frame(matrix(nrow = number_of_genes, ncol = number_of_samples), che
    colnames(genotype_per_cytoband) <- first_line_split[[1]][c(10 : (10 + number_of_samples - 1))]
    rownames(genotype_per_cytoband) <- make.names(gene_list, unique=TRUE)

    i <- 1
    for(actual_copy_number in line_split[[1]][c(10:(10 + number_of_samples -1))]) {
      if (as.numeric(actual_copy_number) < -0.3) {
        genotype_per_cytoband[, i] <- rep(1, number_of_genes)
        i <- i + 1
      } else {
        genotype_per_cytoband[, i] <- rep(0, number_of_genes)
        i <- i + 1
      }
    }

    # Deal with identical gene names in different cytoband
    intersect_genes <- intersect(rownames(genotype_deletion), rownames(genotype_per_cytoband))
    if(length(intersect_genes) == 0){
      genotype_deletion <- rbind(genotype_deletion, genotype_per_cytoband)
    } else {
      for (gene in intersect_genes) {
        genotype_deletion[gene, ] <- as.integer(genotype_deletion[gene, ] | genotype_per_cytoband[gen
      }
    }
  }
}
close(all_lesions.conf_99)

snv_file <- file("../../Data/msk_impact_2017/data_mutations.txt", "r")
line = readLines(snv_file, n = 1)
```

```
gene_pool <- c() # get all the snv related genes
sample_pool <- first_line_split[[1]][10:length(first_line_split[[1]])]

snv_df <- data.frame(matrix(0, nrow = 0, ncol = length(sample_pool)))
colnames(snv_df) <- sample_pool

while(TRUE) {
  line = readLines(snv_file, n = 1)
  if (length(line) == 0) {
    break
  }
  line_split = strsplit(line, "\t")[[1]]
  tumor_barcode <- line_split[17]; Hugo_Symbol <- line_split[1];
  if(tumor_barcode %in% sample_pool) {
    gene_pool <- c(gene_pool, Hugo_Symbol)

    if(Hugo_Symbol %in% rownames(snv_df)) {
      snv_df[Hugo_Symbol, tumor_barcode] <- 1
    } else {
      new_gene <- data.frame(matrix(0, nrow = 1, ncol = length(sample_pool)))
      rownames(new_gene) <- Hugo_Symbol; colnames(new_gene) <- sample_pool
      snv_df = rbind(snv_df, new_gene)
    }
  }
}
gene_pool <- unique(gene_pool)
close(snv_file)

amplification_genes <- rownames(genotype_amplification)
deletion_genes <- rownames(genotype_deletion)

intersection_amp_del <- intersect(amplification_genes, deletion_genes)
genotype_deletion_2 <- genotype_deletion
for(gene in intersection_amp_del) {
  amp_gene <- genotype_amplification[gene, ]
  del_gene <- genotype_deletion[gene, ]
  amp_del_or <- as.numeric(amp_gene | del_gene)
  genotype_amplification[gene, ] <- amp_del_or

  genotype_deletion_2 <- genotype_deletion_2[!row.names(genotype_deletion_2) %in% gene, ]
}

input_of_Meta_MHN <- rbind(genotype_amplification, genotype_deletion_2)
row_name <- rownames(input_of_Meta_MHN)
intersection_input_gene_pool <- intersect(row_name, gene_pool)
snv_df_2 <- snv_df

for(gene in intersection_input_gene_pool) {
  input_gene <- input_of_Meta_MHN[gene, ]
  snv_gene <- snv_df[gene, ]
  or <- as.numeric(input_gene | snv_gene)
  input_of_Meta_MHN[gene, ] <- or
  snv_df_2 <- snv_df_2[!row.names(snv_df_2) %in% gene, ]
```

```
  }

  input_of_Meta_MHN <- rbind(input_of_Meta_MHN, snv_df_2)

  saveRDS(input_of_Meta_MHN, file = "Input_of_Meta_MHN")

  return_list <- list(genotype_amplification, genotype_deletion, snv_df, input_of_Meta_MHN, gene_pool, c
  return(return_list)
}

return_list <- step_2_binarised_genome_construction(return_list)
```

Step 3: Plot a oncoplot

If you simply want to convert GISTIC result to Meta_MHN, you can leave at step 2. In step 3, we draw oncoplots for this whole process to have a "intuitive" understanding of the dataset.

```
step_3_oncoplots <- function(return_list_from_step_2) {
  genotype_amplification <- return_list_from_step_2[[1]]
  genotype_deletion <- return_list_from_step_2[[2]]
  snv_df <- return_list_from_step_2[[3]]
  gene_pool <- return_list_from_step_2[[5]]
  cancer_type <- return_list_from_step_2[[6]]
  number_of_samples <- return_list_from_step_2[[7]]
  first_line_split <- return_list_from_step_2[[8]]

  amplification_genes <- rownames(genotype_amplification)
  deletion_genes <- rownames(genotype_deletion)
  genes_union <- unique(union(amplification_genes, deletion_genes))
  genes_union <- unique(union(genes_union, gene_pool))

  missing_from_deletion = setdiff(genes_union, deletion_genes)
  genotype_deletion_oncoplot <- genotype_deletion
  if(length(missing_from_deletion) != 0) {
    to_be_add_to_deletion = data.frame(matrix(0, length(missing_from_deletion), number_of_samples))
    colnames(to_be_add_to_deletion) <- first_line_split[[1]][c(10 : (10 + number_of_samples - 1))]
    rownames(to_be_add_to_deletion) <- missing_from_deletion
    genotype_deletion_oncoplot <- rbind(genotype_deletion_oncoplot, to_be_add_to_deletion)
  }


  missing_from_amplification = setdiff(genes_union, amplification_genes)
  genotype_amplification_oncoplot <- genotype_amplification
  if(length(missing_from_amplification) != 0) {
    to_be_add_to_amplification = data.frame(matrix(0, length(missing_from_amplification), number_of_samp
    colnames(to_be_add_to_amplification) <- first_line_split[[1]][c(10 : (10 + number_of_samples - 1))]
    rownames(to_be_add_to_amplification) <- missing_from_amplification
    genotype_amplification_oncoplot <- rbind(genotype_amplification_oncoplot, to_be_add_to_amplificatio
  }

  missing_from_snv = setdiff(genes_union, gene_pool)
  genotype_snv_oncoplot <- snv_df
  if(length(missing_from_snv) != 0) {
    to_be_add_to_snv = data.frame(matrix(0, length(missing_from_snv), number_of_samples))
```

```
    colnames(to_be_add_to_snv) <- first_line_split[[1]][c(10 : (10 + number_of_samples - 1))]
    rownames(to_be_add_to_snv) <- missing_from_snv
    genotype_snv_oncoplot <- rbind(genotype_snv_oncoplot, to_be_add_to_snv)
  }

  # Reorder each oncoplot data frame so that they are in the same order
  genes <- rownames(genotype_amplification_oncoplot)
  genotype_deletion_oncoplot <- genotype_deletion_oncoplot[genes, ]
  genotype_snv_oncoplot <- genotype_snv_oncoplot[genes, ]

  # only show top "selection" genes
  genes <- colnames(genotype_amplification_oncoplot)
  row_sum_amplification <- rowSums(genotype_amplification_oncoplot)
  row_sum_deletion <- rowSums(genotype_deletion_oncoplot)
  row_sum_snv <- rowSums(genotype_snv_oncoplot)
  gene_sum <- row_sum_amplification + row_sum_deletion + row_sum_snv
  order <- order(gene_sum, decreasing = TRUE)
  selection <- nrow(genotype_amplification_oncoplot)
  selection <- min(selection, 100)
  top_100_genes <- names(gene_sum)[order[1:selection]]

  metastasis_genotype_amplification_100 <- genotype_amplification_oncoplot[top_100_genes, ]
  metastasis_genotype_deletion_100 <- genotype_deletion_oncoplot[top_100_genes, ]
  metastasis_genotype_snv_100 <- genotype_snv_oncoplot[top_100_genes, ]

  onco_matrix_list <- list(amp = data.matrix(metastasis_genotype_amplification_100),
                           del = data.matrix(metastasis_genotype_deletion_100),
                           snv = data.matrix(metastasis_genotype_snv_100))

  col <- c("amp" = "red", "del" = "blue", "snv" = "green")
  alter_fun = list(
    background = alter_graphic("rect", fill = "#CCCCCC"),
    amp = alter_graphic("rect", fill = col["amp"]),
    del = alter_graphic("rect", fill = col["del"]),
    snv = alter_graphic("rect", fill = col["snv"])
  )

  column_title = paste(cancer_type, "cancer patients with metastasis top ", selection)
  heatmap_legend_param = list(title = "Alterations", at=c("amp", "del", "snv"), labels = c("Amplificati
  png(filename = paste(cancer_type, "/oncoplot_metastasis_top_",selection,".png", sep = ""),width=25,he
  # png(filename = paste("default_gistic/", cancer_type, "/oncoplot_metastasis_top_",selection,".png",
  print(oncoPrint(onco_matrix_list, alter_fun = alter_fun, col = col,
            column_title = column_title, heatmap_legend_param = heatmap_legend_param,
            height = unit(35, "cm"), width = unit(15, "cm")))
  dev.off()
}

step_3_oncoplots(return_list)
```

```
## All mutation types: amp, del, snv.
```

```
## `alter_fun` is assumed vectorizable. If it does not generate correct
## plot, please set `alter_fun_is_vectorized = FALSE` in `oncoPrint()`.
```

```
## pdf
```

```
##  2
```