Informe 5

Datos de Expresión génica y proteica.

Carmen Lebrero Cia

19/10/2020

Contents

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("TCGAbiolinks")
```

```
## Bioconductor version 3.12 (BiocManager 1.30.10), R 4.0.3 (2020-10-10)

## Installing package(s) 'TCGAbiolinks'

## package 'TCGAbiolinks' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Carmen\AppData\Local\Temp\Rtmp8k0nJG\downloaded_packages

## Installation path not writeable, unable to update packages: codetools,
##   KernSmooth, nlme

## Old packages: 'Biobase', 'BiocParallel', 'Biostrings', 'colorspace',
##   'ComplexHeatmap', 'DelayedArray', 'GenomicAlignments', 'GenomicRanges',
##   'IRanges', 'Rsamtools', 'S4Vectors', 'ShortRead', 'XVector', 'zlibbioc'
```

```
library(TCGAbiolinks)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count
```

```
##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
library(DT)
```

Archivos de datos clínicos

Si utilizamos la función `str()` sobre `ClinicKIRC2` (datos clínicos de TCGA-KIRC) observamos que se trata de un archivo Dataframe con 537 observaciones y 66 variables.

```
str(ClinicKIRC2)
```

```
## 'data.frame':    537 obs. of  66 variables:
##  $ bcr_patient_barcode                          : chr  "TCGA-3Z-A93Z" "TCGA-6D-AA2E" "TCGA-A3-3306" "TC
##  $ additional_studies                           : Factor w/ 2 levels "","TCGAFPPP": 1 1 1 1 1 1 1 1 1 1
##  $ tumor_tissue_site                            : Factor w/ 1 level "Kidney": 1 1 1 1 1 1 1 1 1 1 ...
##  $ histological_type                            : Factor w/ 1 level "Kidney Clear Cell Renal Carcinoma"
##  $ other_dx                                     : Factor w/ 4 levels "No","Yes","Yes, History of Prior
##  $ gender                                       : Factor w/ 2 levels "FEMALE","MALE": 2 1 2 2 1 2 2 2 2
##  $ vital_status                                 : Factor w/ 3 levels "","Alive","Dead": 2 2 2 2 2 3 3 2
##  $ days_to_birth                                : int  -25205 -25043 -24569 -24315 -28287 -21183 -21556
##  $ days_to_last_known_alive                     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ days_to_death                                : int  NA NA NA NA NA 1191 735 NA NA NA ...
##  $ days_to_last_followup                        : int  4 135 1120 1436 16 NA NA 1493 1491 1130 ...
##  $ race_list                                    : Factor w/ 4 levels "","ASIAN","BLACK OR AFRICAN AMERI
##  $ tissue_source_site                           : Factor w/ 20 levels "3Z","6D","A3",..: 1 2 3 3 3 3 3 3
##  $ patient_id                                   : chr  "A93Z" "AA2E" "3306" "3307" ...
##  $ bcr_patient_uuid                             : chr  "2B1DEA0A-6D55-4FDD-9C1C-0D9FBE03BD78" "D3B47E53-
##  $ history_of_neoadjuvant_treatment             : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ..
##  $ informed_consent_verified                    : Factor w/ 1 level "YES": 1 1 1 1 1 1 1 1 1 1 ...
##  $ icd_o_3_site                                 : Factor w/ 1 level "C64.9": 1 1 1 1 1 1 1 1 1 1 ...
##  $ icd_o_3_histology                            : Factor w/ 2 levels "8310/3","8312/3": 1 1 1 1 1 1 1 1
##  $ icd_10                                       : Factor w/ 1 level "C64.9": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tissue_prospective_collection_indicator      : Factor w/ 3 levels "","NO","YES": 3 3 2 2 2 2 2 2 2 2
##  $ tissue_retrospective_collection_indicator    : Factor w/ 3 levels "","NO","YES": 2 2 3 3 3 3 3 3 3 3
##  $ ethnicity                                    : Factor w/ 3 levels "","HISPANIC OR LATINO",..: 3 3 3
##  $ person_neoplasm_cancer_status                : Factor w/ 3 levels "","TUMOR FREE",..: 2 2 1 3 2 2 2 2
##  $ performance_status_scale_timing              : Factor w/ 3 levels "","Other","Preoperative": 2 1 1 1
##  $ days_to_initial_pathologic_diagnosis         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age_at_initial_pathologic_diagnosis          : int  69 68 67 66 77 57 59 57 67 70 ...
##  $ year_of_initial_pathologic_diagnosis         : int  2013 2013 2005 2005 2006 2005 2005 2005 2005 200
##  $ day_of_form_completion                       : int  11 17 23 13 12 20 14 14 14 16 ...
##  $ month_of_form_completion                     : int  11 3 8 4 4 4 4 4 4 4 ...
##  $ year_of_form_completion                      : int  2014 2014 2010 2010 2010 2010 2010 2010 2010 201
##  $ laterality                                   : Factor w/ 3 levels "Bilateral","Left",..: 3 3 2 3 3 3
##  $ lactate_dehydrogenase_result                 : Factor w/ 3 levels "","Elevated",..: 3 1 1 1 1 1 1 1
##  $ serum_calcium_result                         : Factor w/ 4 levels "","Elevated",..: 4 1 1 1 4 1 1 1
##  $ hemoglobin_result                            : Factor w/ 4 levels "","Elevated",..: 4 1 1 1 4 3 3 1
##  $ platelet_qualitative_result                  : Factor w/ 4 levels "","Elevated",..: 4 1 1 1 4 1 1 1
##  $ white_cell_count_result                      : Factor w/ 4 levels "","Elevated",..: 4 1 1 1 4 1 1 1
##  $ erythrocyte_sedimentation_rate_result        : Factor w/ 3 levels "","Elevated",..: 1 1 1 1 1 1 1 1
##  $ lymph_node_examined_count                    : int  NA NA NA 4 NA NA NA NA NA NA ...
##  $ number_of_lymphnodes_positive                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ karnofsky_performance_score                  : int  100 NA NA NA NA NA NA NA NA NA ...
##  $ eastern_cancer_oncology_group                : int  0 1 NA NA NA NA NA NA NA NA ...
##  $ primary_lymph_node_presentation_assessment: Factor w/ 3 levels "","NO","YES": 2 2 2 3 2 2 1 2 2 2
##  $ neoplasm_histologic_grade                    : Factor w/ 6 levels "","G1","G2","G3",..: 3 3 4 4 3 3 4
##  $ tobacco_smoking_history                      : int  5 1 NA NA NA NA NA NA NA NA ...
```

```
##  $ year_of_tobacco_smoking_onset             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ stopped_smoking_year                      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ number_pack_years_smoked                  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ targeted_molecular_therapy                : Factor w/ 3 levels "","NO","YES": 2 2 1 1 1 1 1 1 1 1
##  $ radiation_therapy                         : Factor w/ 2 levels "","NO": 2 2 1 1 1 1 1 1 1 1 ...
##  $ primary_therapy_outcome_success           : Factor w/ 4 levels "","Complete Remission/Response",.
##  $ has_new_tumor_events_information          : chr  "NO" "NO" "NO" "NO" ...
##  $ has_drugs_information                     : chr  "NO" "NO" "NO" "NO" ...
##  $ has_radiations_information                : chr  "NO" "NO" "NO" "NO" ...
##  $ has_follow_ups_information                : chr  "YES" "YES" "NO" "NO" ...
##  $ project                                   : Factor w/ 1 level "TCGA-KIRC": 1 1 1 1 1 1 1 1 1 1 ..
##  $ stage_event_system_version                : Factor w/ 4 levels "","5th","6th",..: 4 4 1 1 1 1 1 1
##  $ stage_event_clinical_stage                : logi  NA NA NA NA NA NA ...
##  $ stage_event_pathologic_stage              : Factor w/ 5 levels "","Stage I","Stage II",..: 2 2 2 4
##  $ stage_event_tnm_categories                : Factor w/ 59 levels "M0T1aN0M0","M0T1aNX",..: 1 15 23
##  $ stage_event_psa                           : logi  NA NA NA NA NA NA ...
##  $ stage_event_gleason_grading               : logi  NA NA NA NA NA NA ...
##  $ stage_event_ann_arbor                     : logi  NA NA NA NA NA NA ...
##  $ stage_event_serum_markers                 : logi  NA NA NA NA NA NA ...
##  $ stage_event_igcccg_stage                  : logi  NA NA NA NA NA NA ...
##  $ stage_event_masaoka_stage                 : logi  NA NA NA NA NA NA ...
```

De entre todas las variables las más interesantes parecen ser `bcr_patient_barcode`, `vital_status` o `stage_event_pathologic_stage`. Vamos a encontrar los índices de cada variable en el dataframe y podemos hacer un dataframe más pequeño.

```
grep("bcr_patient_barcode", colnames(ClinicKIRC2))
```

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

```
grep("vital_status", colnames(ClinicKIRC2))
```

```
## [1] 7
```

```
grep("stage_event_pathologic_stage", colnames(ClinicKIRC2))
```

```
## [1] 59
```

```
ClinicS <- ClinicKIRC2[,c(1,7,59)]
head(ClinicS)
```

```
##   bcr_patient_barcode vital_status stage_event_pathologic_stage
## 1        TCGA-3Z-A93Z        Alive                      Stage I
## 2        TCGA-6D-AA2E        Alive                      Stage I
## 3        TCGA-A3-3306        Alive                      Stage I
## 4        TCGA-A3-3307        Alive                     Stage III
## 5        TCGA-A3-3308        Alive                     Stage III
## 6        TCGA-A3-3311         Dead                      Stage I
```

Vamos a ver si existen missing values con `is.na()`.

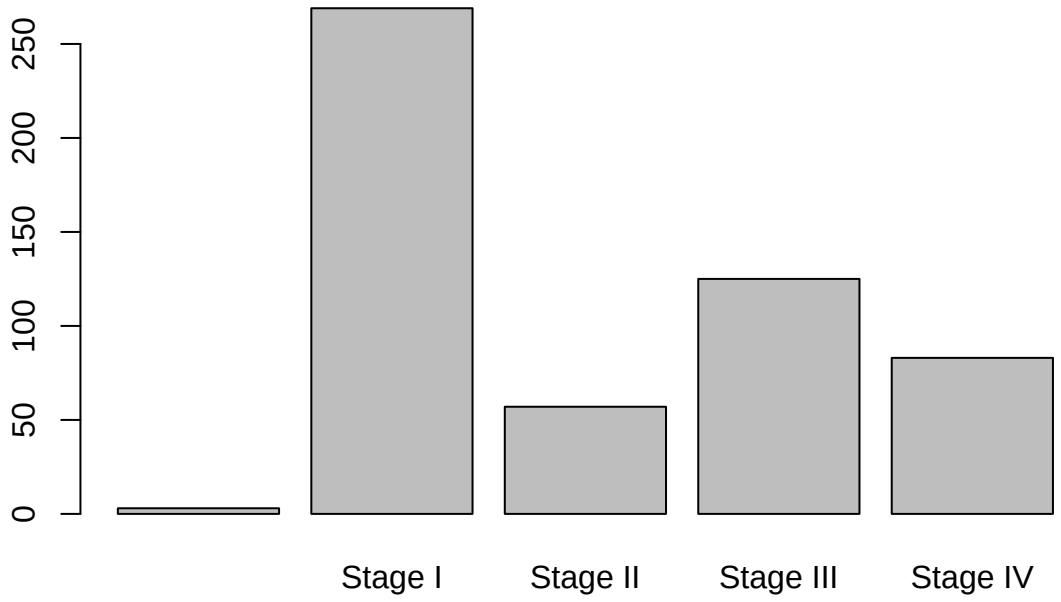
```
sum(is.na(ClinicS))
```

```
## [1] 0
```

Observamos que no tenemos ningún NA en este Dataset, por lo que tenemos datos de nuestra variable de interés para los 537 pacientes del estudio TCGA-KIRC.

Podemos sacar más información a partir de estas variables. Por ejemplo, podemos observar con una gráfica según la variable de estadío patológico que la mayoría de nuestras pertenecen al estadío I, seguido de las muestras en el estadío III, IV, II y 0.

```
plot(ClinicKIRC2$stage_event_pathologic_stage)
```



Podemos obtener el número exacto de pacientes con la función `summmary()`.

```
summary(ClinicKIRC2$stage_event_pathologic_stage)
```

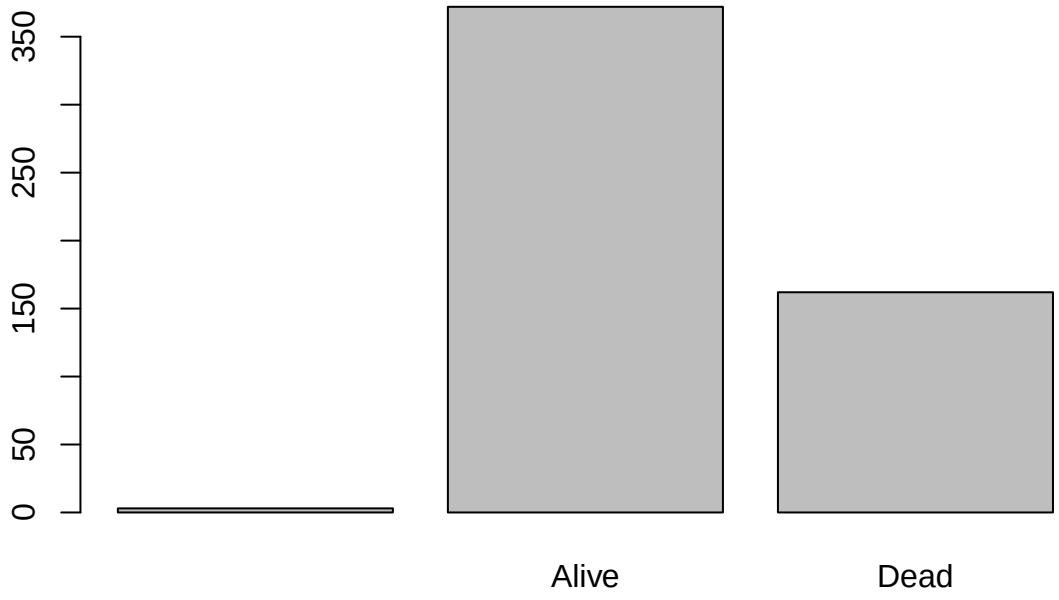
```
##             Stage I  Stage II Stage III  Stage IV
##         3       269        57       125        83
```

Esta clasificación por estadíos se refiere a lo siguiente a un método de agrupación de los pacientes según una serie de características de los tumores como la localización y el tamaño del tumor (T), si se observan ganglios linfáticos cerca (N) o si hay metástasis (M).

- **Estadío 0**. Describe cáncer "in situ" que están localizados en el lugar de origen y no se han esparcido a tejidos cercanos. Son tumores fácilmente curables que se pueden quitar con una cirugía.
- **Estadío 1**. Se trata de un tumor pequeño que no se ha extendido de forma muy profunda a tejidos colindantes ni a los ganglios linfáticos. A veces se le denomina cáncer temprano.
- **Estadíos II y III**. Estos dos estadíos indican tumores más grandes que se han extendido de forma más profunda a tejidos y que pueden haber llegado a los ganglios linfáticos pero no a otros órganos.
- **Estadío IV**. Este estadío significa que el cáncer se ha extendido a otros órganos. También conocido como cáncer metastásico avanzado.

También podemos obtener información acerca de los fallecimientos de nuestra muestra. Y observamos que 372 siguen vivos y 162 han fallecido. Volvemos a ver 3 muestras que no están catalogadas en ninguno de estos dos grupos, debemos ver si esas muestras, aunque hayamos hecho anteriormente la búsqueda de missing values, no contienen información.

```
plot(ClinicKIRC2$vital_status)
```

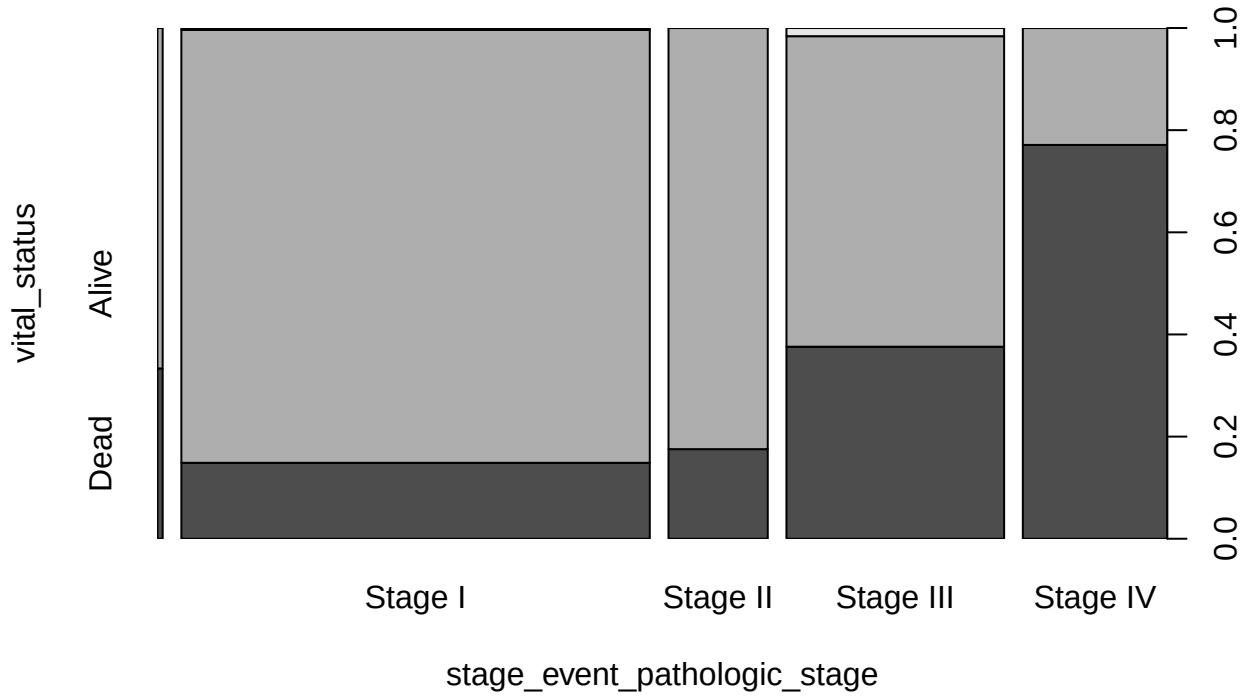



```
summary(ClinicKIRC2$vital_status)
```

```
##       Alive  Dead
##     3   372   162
```

Otro análisis interesante sería saber cuántos fallecidos hay según el estadío del tumor.

```r
plot(vital_status ~ stage_event_pathologic_stage, data = ClinicKIRC2)
```



Fase I (n=269)

Fallecidos

```r
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Dead" & ClinicKIRC2$stage_event_patl
```

```
## [1] 40
```

Vivos

```r
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Alive" & ClinicKIRC2$stage_event_pa
```

```
## [1] 228
```

Fase II (n=57)

Fallecidos

```r
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Dead" & ClinicKIRC2$stage_event_patl
```

```
## [1] 10
```

Vivos

```
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Alive" & ClinicKIRC2$stage_event_pa
```

```
## [1] 47
```

Fase III (n=125)

Fallecidos

```
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Dead" & ClinicKIRC2$stage_event_pat
```

```
## [1] 47
```

Vivos

```
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Alive" & ClinicKIRC2$stage_event_pa
```

```
## [1] 76
```

Fase IV (n=83)

Fallecidos

```
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Dead" & ClinicKIRC2$stage_event_pat
```

```
## [1] 64
```

Vivos

```
length(subset(ClinicKIRC2$patient_id , ClinicKIRC2$vital_status == "Alive" & ClinicKIRC2$stage_event_pa
```

```
## [1] 19
```

Vamos a buscar esos pacientes que no tienen vital_status asignado:

```
subset(ClinicKIRC2$bcr_patient_barcode , ClinicKIRC2$vital_status != "Dead" & ClinicKIRC2$vital_status
```

```
## [1] "TCGA-BP-4326" "TCGA-BP-4329" "TCGA-BP-4334"
```

Y ver si estos tres son los mismos que no tienen asignada la fase del tumor:

```
subset(ClinicKIRC2$bcr_patient_barcode , ClinicKIRC2$stage_event_pathologic_stage != "Stage I" & Clinic
```

```
## [1] "TCGA-B4-5838" "TCGA-BP-4798" "TCGA-MM-A563"
```

No son los mismos pacientes los que no tienen etiqueta para vital_status y para la Fase del cancer. Creamos dos Datasets. `ClinicSVital` sin los tres pacientes que no tienen etiqueta para Vital_status y `ClinicSStage` sin los pacientes que no tienen etiqueta para Stage. Además se crearán dos cadenas de caracteres con los barcodes de cada Dataframe.

```
ClinicSVital <- subset(ClinicS, ClinicS$bcr_patient_barcode != "TCGA-BP-4326" & ClinicS$bcr_patient_bar
```

```
ClinicSStage <- subset(ClinicS, ClinicS$bcr_patient_barcode != "TCGA-B4-5838" & ClinicS$bcr_patient_bar
```

Si elegimos como variable a modelar `vital_status`se tratará de un problema de clasificaicón binaria. Mientras que si elegimos el estadío del tumor, sería un problema de clasificación multiclase.

Datos de Expresión Génica

Vamos a explorar nuestros datos de Expresión Génica `ExpGenKIRC3`. Observamos que se trata de un archivo `RangedSummarizedExperiment`, para tratar este tipo de archivos podemos seguir la siguiente guía: https://www.bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html.

Introducción

La clase `SummarizedExperiment` se usa para llenar matrices rectangulares de resultados experimentales producidos normalmente en experimentos de secuenciación o microarrays. Cada objeto almacena observaciones de una o más muestras, junto con metadatos adicionales que describen obsevaciones (características) y muestras (fenotipos).

Un aspecto clave de la clase `SummarizedExperiment` es la coordinación de los metadatos y los ensayos cuando se realizan subagrupaciones o subconjuntos. Por ejemplo, si quieres excluir una muestra se puede hacer en los metadatos y en los ensayos en una única operación, lo que asegura que los metadatos y los datos observados permanezcan sincronizados.

Anatomía de SummarizedExperiment

El paquete `SummarizedExperiment` contiene dos clases: `SummarizedExperiment` y `RangedSummarizedExperiment`.

`SummarizedExperiment` es un contenedor similar a una matriz donde las filas representan características de interés (por ejemplo: genes, transcritos, exones, etc.) y las columnas representan las muestras. Los objetos contienen uno o más ensayos, cada uno representado por un objeto matriz de número u otro modo. Las filas del objeto `SummarizedExperiment` representan características de interés. La información acerca de estas características está almacenada en un objeto Dataframe, accesible usando la función `rowData()`. Cada fila del Dataframe aporta información de la característica en la fila correspondiente del objeto `SummarizedExperiment`. Las columnas del Datafram representan diferentes atributos de las características de interés como IDs de genes o transcritos.

`RangedSummarizedExperiment` es el "hijo" de la clase `SummarizedExperiment`, lo que significaa que todos los métodos de `SummarizedExperiment`también funcionan sobre `RangedSummarizedExperiment`.

La diferencia fundamental entre las dos clases es que las filas de `RangedSummarizedExperiment`representan rangos genómicos de interés en vez de un Dataframe de carcterísticas. Los rangos de `RangedSummarizedExperiment`se describen en el objeto `GRanges`o `GRangesList`, accesible utilizando la función `rowRanges()`.

Assays

```
library(SummarizedExperiment)
se <- ExpGenKIRC3
se
```

```
## class: RangedSummarizedExperiment
## dim: 19947 606
## metadata(1): data_release
## assays(2): raw_count scaled_estimate
## rownames(19947): A1BG|1 A2M|2 ... TMED7-TICAM2|100302736
##   LOC100303728|100303728
## rowData names(4): gene_id entrezgene ensembl_gene_id
##   transcript_id.transcript_id_TCGA-B0-5694-01A-11R-1541-07
## colnames(606): TCGA-B0-5694-01A-11R-1541-07
##   TCGA-CJ-4637-01A-02R-1325-07 ... TCGA-CJ-4871-01A-01R-1305-07
##   TCGA-AK-3460-01A-02R-1277-07
## colData names(70): barcode patient ... paper_mRNA_cluster
##   paper_microRNA_cluster
```

Para recuperar los datos a partir del experimento a partir de un objeto `SummarizedExperiment` se puede utilizar `assays()`. Un objeto puede tener múltiples dataset de ensayos a los que se puede acceder usando el operador `$`. En nuestro caso tenemos dos datasets: `raw_count` y `scaled_estimate`. Partiremos desde el archivo `raw_count` para realizar nuestro análisis.

```
assays(se)$raw_count[1:3,1:4]
```

```
##        TCGA-B0-5694-01A-11R-1541-07 TCGA-CJ-4637-01A-02R-1325-07
## A1BG|1                        36.00                        87.66
## A2M|2                      71105.61                     47586.92
## NAT1|9                       111.00                       295.00
##        TCGA-CZ-4860-01A-01R-1305-07 TCGA-B0-4706-01A-01R-1503-07
## A1BG|1                        60.00                       180.09
## A2M|2                      39077.92                     69333.51
## NAT1|9                       208.00                       140.00
```

```
rowRanges(se)
```

```
## GRanges object with 19947 ranges and 4 metadata columns:
##                            seqnames              ranges strand |      gene_id
##                               <Rle>           <IRanges>  <Rle> |  <character>
##                   A1BG|1    chr19   58856544-58864865      - |          A1BG
##                    A2M|2    chr12     9220260-9268825      - |           A2M
##                   NAT1|9     chr8   18027986-18081198      + |          NAT1
##                  NAT2|10     chr8   18248755-18258728      + |          NAT2
##              SERPINA3|12    chr14   95058395-95090983      + | RP11-986E7.7
##                      ...      ...                 ...    ... .           ...
##   LOC100302401|100302401     chr1 178060643-178063119      - |   RASAL2-AS1
##   LOC100302640|100302640     chr3 106555658-106959488      - |    LINC00882
##     NCRNA00182|100302692     chrX   73183790-73513409      - |           FTX
##   TMED7-TICAM2|100302736     chr5 114914339-114961876      - |       TICAM2
##   LOC100303728|100303728     chrX 118599997-118603061      - |  SLC25A5-AS1
##                            entrezgene ensembl_gene_id
##                             <integer>     <character>
##                   A1BG|1          1 ENSG00000121410
##                    A2M|2          2 ENSG00000175899
##                   NAT1|9          9 ENSG00000171428
##                  NAT2|10         10 ENSG00000156006
```

```
##              SERPINA3|12         12 ENSG00000273259
##                      ...        ...             ...
##   LOC100302401|100302401  100302401 ENSG00000224687
##   LOC100302640|100302640  100302640 ENSG00000242759
##     NCRNA00182|100302692  100302692 ENSG00000230590
##   TMED7-TICAM2|100302736  100302736 ENSG00000243414
##   LOC100303728|100303728  100303728 ENSG00000224281
##                          transcript_id.transcript_id_TCGA-B0-5694-01A-11R-1541-07
##                                                                      <character>
##                   A1BG|1                                      uc002qsd.3,uc002qsf.1
##                    A2M|2                                     uc001qvj.1,uc001qvk...
##                   NAT1|9                                     uc003wyq.2,uc003wyr...
##                  NAT2|10                                                uc003wyw.1
##              SERPINA3|12                                     uc001ydo.3,uc001ydp...
##                      ...                                                       ...
##   LOC100302401|100302401                                      uc001gln.1,uc001glo.1
##   LOC100302640|100302640                                      uc003dwf.3,uc011bhk.1
##     NCRNA00182|100302692                                      uc004ebr.1,uc010nlq.1
##   TMED7-TICAM2|100302736                                      uc003krd.2,uc003kre.2
##   LOC100303728|100303728                                      uc004ere.1,uc004erg.1
##   -------
##   seqinfo: 24 sequences from an unspecified genome; no seqlengths
```

Columnas (datos de las muestras)

Se puede acceder a los metadatos que describen las muestras usando ‘`colData()`, y es un Dataframe que puede almacenar cualquier número de columna.

```
colData(se)
```

Se puede acceder a estos metadatos usando `$`, lo que hace más sencillo sustraer un objeto entero dado un fenotipo. Por ejemplo, podemos extraer todas las muestras que tengan una etiqueta para el estado vital:

```
se[, se$vital_status == "Dead" | se$vital_status == "Alive"]
```

```
## class: RangedSummarizedExperiment
## dim: 19947 606
## metadata(1): data_release
## assays(2): raw_count scaled_estimate
## rownames(19947): A1BG|1 A2M|2 ... TMED7-TICAM2|100302736
##   LOC100303728|100303728
## rowData names(4): gene_id entrezgene ensembl_gene_id
##   transcript_id.transcript_id_TCGA-B0-5694-01A-11R-1541-07
## colnames(606): TCGA-B0-5694-01A-11R-1541-07
##   TCGA-CJ-4637-01A-02R-1325-07 ... TCGA-CJ-4871-01A-01R-1305-07
##   TCGA-AK-3460-01A-02R-1277-07
## colData names(70): barcode patient ... paper_mRNA_cluster
##   paper_microRNA_cluster
```

En este Dataframe, a pesar de ser del proyecto TCGA-KIRC, hay 606 muestras, cuando en el de datos clínicos había 537. Parece que todas las muestras de datos de Expresión génica tienen su etiqueta de `vital_status` correspondiente. Encontramos 202 muertos y 404 vivos.

Vamos a buscar Missing values para la etiqueta de Fase del tumor para lo cual utilizaremos el siguiente código:

```
which(is.na(se$ajcc_pathologic_stage))
```

```
## [1] 128 290 460
```

```
se$patient[c(128,190,460)]
```

```
## [1] "TCGA-BP-4798" "TCGA-BP-4338" "TCGA-MM-A563"
```

Los códigos de los pacientes que no tienen esta etiqueta disponible son los mismos que no la tenían para los datos clínicos, lo cual tiene sentido.

Preprocesado: Exploración, Control de Calidad y Normalización de datos TCGA

Seguiremos el Pipeline mostrado en: https://bioconductor.org/packages/release/bioc/vignettes/TCGAbiolinks/inst/doc/analysis.html#TCGAanalyze:_Analyze_data_from_TCGA

Control de calidad

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-3399-8

```
KIRCnaseq_CorOutliers <- TCGAanalyze_Preprocessing(ExpGenKIRC3)
```

```
knitr::include_graphics("figures/PreprocessingOutput.png")
```

```
BiocManager::install(c("shiny", "FactoMineR", "factoextra", "som", "psych", "data.table", "ape", "corrp
```

```
## Bioconductor version 3.12 (BiocManager 1.30.10), R 4.0.3 (2020-10-10)
```

```
## Installing package(s) 'shiny', 'FactoMineR', 'factoextra', 'som', 'psych',
##   'data.table', 'ape', 'corrplot', 'limma', 'DESeq2'
```

```
## package 'shiny' successfully unpacked and MD5 sums checked
## package 'FactoMineR' successfully unpacked and MD5 sums checked
## package 'factoextra' successfully unpacked and MD5 sums checked
## package 'som' successfully unpacked and MD5 sums checked
## package 'psych' successfully unpacked and MD5 sums checked
## package 'data.table' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'data.table'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE):
## problema al copiar D:\Users\Carmen\Documents\R\win-
## library\4.0\00LOCK\data.table\libs\x64\datatable.dll a D:
## \Users\Carmen\Documents\R\win-library\4.0\data.table\libs\x64\datatable.dll:
## Permission denied
```

```
## Warning: restored 'data.table'

## package 'ape' successfully unpacked and MD5 sums checked
## package 'corrplot' successfully unpacked and MD5 sums checked
## package 'limma' successfully unpacked and MD5 sums checked
## package 'DESeq2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Carmen\AppData\Local\Temp\Rtmp8k0nJG\downloaded_packages

## Installation path not writeable, unable to update packages: codetools,
##   KernSmooth, nlme

## Old packages: 'Biobase', 'BiocParallel', 'Biostrings', 'colorspace',
##   'ComplexHeatmap', 'DelayedArray', 'GenomicAlignments', 'GenomicRanges',
##   'IRanges', 'Rsamtools', 'S4Vectors', 'ShortRead', 'XVector', 'zlibbioc'
```

```
library("shiny")
```

```
##
## Attaching package: 'shiny'

## The following objects are masked from 'package:DT':
##
##     dataTableOutput, renderDataTable
```

iSeqQC requires two files for the analysis. Both files should be ASCII formatted tab-delimited file only

Creando archivo 1 File 1- Sample phenotype data: First 4 columns should strictly match the names and order as mentioned below (names case-sensitive) Sample names in first column 'samples' should match the names in counts matrix file column 1: samples column 2: shortnames column 3: groups column 4: include column 5-11: any factors such as library method, protocol etc.

```
QCinput1 <- data.frame(samples = ExpGenKIRC3$barcode, shortnames = ExpGenKIRC3$sample, groups = ExpGenK
QCinput1[1:10,1:4]
```

```
##                         samples       shortnames groups include
## 1  TCGA-B0-5694-01A-11R-1541-07 TCGA-B0-5694-01A   Dead    TRUE
## 2  TCGA-CJ-4637-01A-02R-1325-07 TCGA-CJ-4637-01A   Dead    TRUE
## 3  TCGA-CZ-4860-01A-01R-1305-07 TCGA-CZ-4860-01A   Dead    TRUE
## 4  TCGA-B0-4706-01A-01R-1503-07 TCGA-B0-4706-01A   Dead    TRUE
## 5  TCGA-B4-5844-01A-11R-1672-07 TCGA-B4-5844-01A  Alive    TRUE
## 6  TCGA-BP-5001-01A-01R-1334-07 TCGA-BP-5001-01A  Alive    TRUE
## 7  TCGA-B8-A54F-01A-11R-A266-07 TCGA-B8-A54F-01A  Alive    TRUE
## 8  TCGA-B8-5552-01B-11R-1672-07 TCGA-B8-5552-01B  Alive    TRUE
## 9  TCGA-B0-5690-11A-01R-1541-07 TCGA-B0-5690-11A  Alive    TRUE
## 10 TCGA-BP-4760-01A-02R-1420-07 TCGA-BP-4760-01A  Alive    TRUE
```

```
write.table(QCinput1, file = "QCinput1.txt" , append = FALSE,
            row.names = FALSE)
```

```
QCinput2 <- TCGAanalyze_Normalization(tabDF = ExpGenKIRC3, geneInfo =  geneInfo)
```

Creando archivo 2

```
## I Need about  150 seconds for this Complete Normalization Upper Quantile  [Processing 80k elements /s
```

```
## Step 1 of 4: newSeqExpressionSet ...
```

```
## Step 2 of 4: withinLaneNormalization ...
```

```
## Step 3 of 4: betweenLaneNormalization ...
```

```
## Step 4 of 4: exprs ...
```

```
write.table(QCinput2, file = "QCinput2.txt")
```

```
#setwd("D:/Users/Carmen/Documents/R/win-library/4.0/iSeqQC_shinyapp")
#runApp(appDir = getwd())
```

Iniciar y utilizar el programa Resultado del programa: La mayoría de las gráficas no pueden apreciarse debido a una gran cantidad de muestras. Mostraremos una gráfica de la densidad de lecturas mapeadas por muestra.

Normalización y Filtrado

Prefiltrado de la base de datos La matriz de conteos presenta varias filas nulas o con pocos fragmentos por muestra/gen. Con el objetivo de disminuir el tamaño del objeto, optimizar el análisis, y reducir el tiempo de ejecución de las funciones, eliminaremos los valores de la matriz de conteos que no nos aporten información relevante sobre la expresión génica, es decir las filas de `assay(se)` que no tengan o tengan pocos conteos.

```
#Número de filas que tiene el archivo assay(se)
nrow(assay(se))
```

```
## [1] 19947
```

```
#subconjunto quedándonos con las filas con sumatorio de conteos mayores a 1 y guardado de este cambio e
keep <- rowSums(assay(se)) > 1
se <- se[keep,]
nrow(assay(se))
```

```
## [1] 19662
```

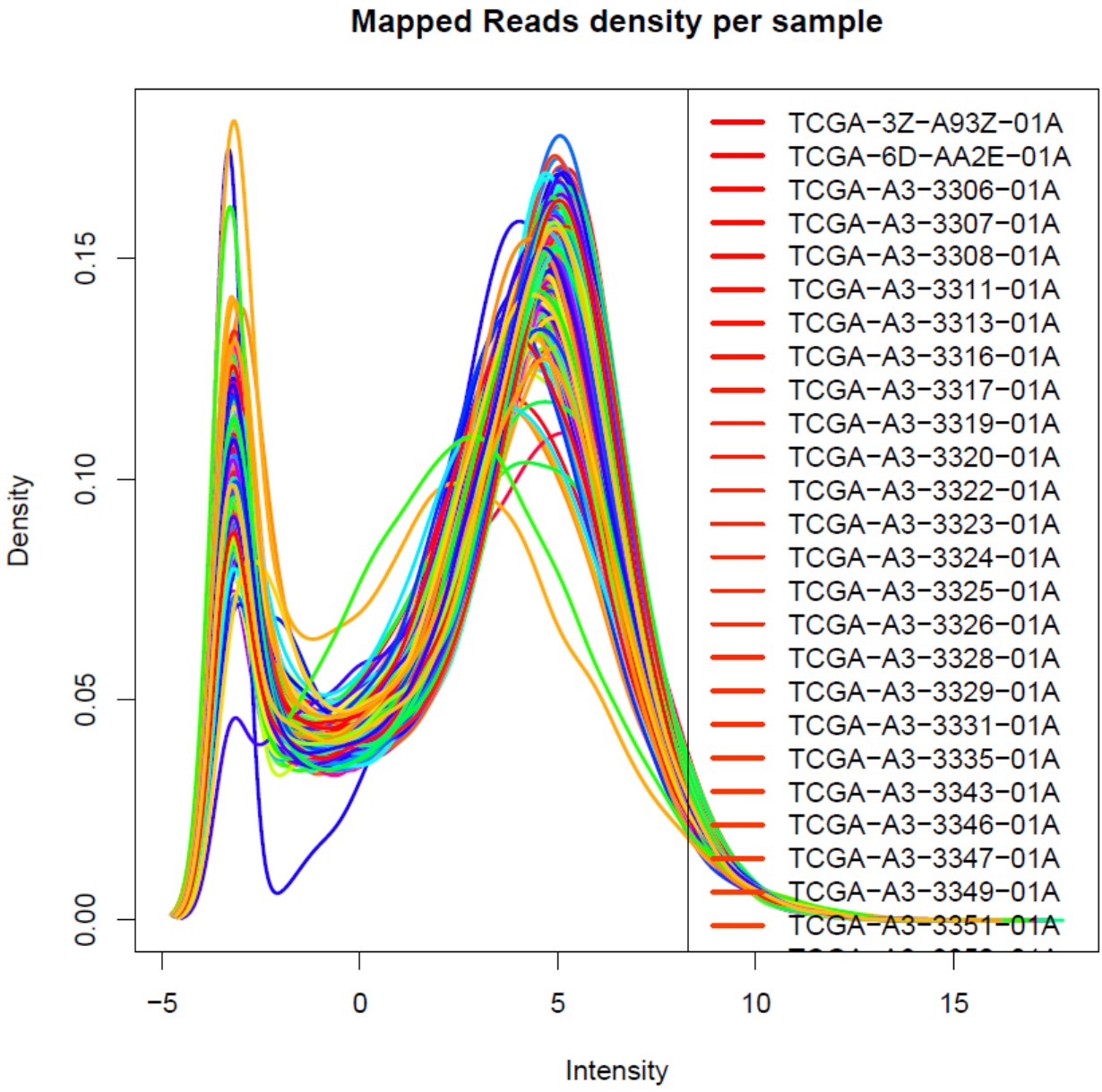


Figure 1: Gráfica de la densidad de lecturas mapeadas por muestra

```
#Downstream análisis usando datos de expresión génica de muestras dde TCGA de IlluminaHiSeq_RNASeqV2 co

library(TCGAbiolinks)
dataNorm <- TCGAanalyze_Normalization(tabDF = se, geneInfo =  geneInfo)
```

Normalización

```
## I Need about  148 seconds for this Complete Normalization Upper Quantile  [Processing 80k elements /s

## Step 1 of 4: newSeqExpressionSet ...

## Step 2 of 4: withinLaneNormalization ...

## Step 3 of 4: betweenLaneNormalization ...

## Step 4 of 4: exprs ...
```

`TCGAanalyze_Normalization` nos ha devuelto el archivo `dataNorm`, que se trata de una matriz bastante grande. Si lo observamos vemos que se trata de la matriz de conteos pero sin decimales.

```
dataNorm[1:10,1:3]
```

```
##          TCGA-B0-5694-01A-11R-1541-07 TCGA-CJ-4637-01A-02R-1325-07
## A1BG                               36                           88
## A2M                             71106                        47587
## NAT1                              111                          295
## NAT2                                5                          126
## SERPINA3                          422                          643
## AADAC                               0                           77
## AAMP                             4536                         4774
## AANAT                               1                            0
## AARS                             4255                         6358
## ABAT                              703                          951
##          TCGA-CZ-4860-01A-01R-1305-07
## A1BG                               60
## A2M                             39078
## NAT1                              208
## NAT2                                9
## SERPINA3                         1568
## AADAC                               4
## AAMP                            11681
## AANAT                               0
## AARS                            30018
## ABAT                              302
```

Pasamos de tener un objeto con 19648 filas a un objeto con 19433 filas.

Transformación log2 Muchos métodos estadísticos de visualización y exploración de datos trabajan mejor con versiones de los datos de conteos que han sido transformados. La distribución de los datos de la matriz de conteos está bastante sesgada debido a que los genes con más conteos muestran diferencias más grandes entre muestras, es por ello que una transformación de los datos ayuda a normalizar la distribución. Estos datos podemos transformarlos con distintos métodos entre los que destacan VST, rlog o log2 pero, ¿qué transformación debemos escoger? En nuestro caso utilizaremos la transformación log2(x+1) que deja los valores menores de 1 como 0, ya que este tipo de valores suele ser ruido.

```
dataNormt <- log2(dataNorm+1)
```

```
dataNormt[1:10,1:3]
```

```
##          TCGA-B0-5694-01A-11R-1541-07 TCGA-CJ-4637-01A-02R-1325-07
## A1BG                         5.209453                     6.475733
## A2M                         16.117704                    15.538310
## NAT1                         6.807355                     8.209453
## NAT2                         2.584963                     6.988685
## SERPINA3                     8.724514                     9.330917
## AADAC                        0.000000                     6.285402
## AAMP                        12.147523                    12.221285
## AANAT                        1.000000                     0.000000
## AARS                        12.055282                    12.634584
## ABAT                         9.459432                     9.894818
##          TCGA-CZ-4860-01A-01R-1305-07
## A1BG                         5.930737
## A2M                         15.254106
## NAT1                         7.707359
## NAT2                         3.321928
## SERPINA3                    10.615630
## AADAC                        2.321928
## AAMP                        13.512000
## AANAT                        0.000000
## AARS                        14.873588
## ABAT                         8.243174
```

Exploración estadística de los datos

Distancias muestrales Una primera aproximación para el análisis de RNA-seq es observar la similitud general entre muestras, para ello podemos utilizar la función `dist()`de R, que nos permite calcular la distancia Euclidea entre muestras. Este abordaje lo realizaremos con los datos transformados para estar seguros que hay una contribución equivalente de todos los genes.

Análisis de expresión diferencial (DEA)

Realizar DEA (*Differential expression analysis*) para identificar genes expresados diferencialmente (DEGs) utilizando la función `TCGAanalyze_DEA`.

`TCGAAnalyze_DEA` utiliza las siguiente funciones de R:

1. `edgeR::DGEList` que convierte la matriz de conteos en un objeto edgeR.
2. `edgeR::estimateCommonDisp` se le asigna a cada gen el mismo estimador de dispersión

3. `edgeR::exactTest`realiza el test de comparación por parejas (*pair-wise test*) para la expresión diferenciaal entre dos grupos.
4. `edgeR::topTags` coge el resultado de `exactTest()`, ajusta los p-valores crudos utilizando la corrección FDR y devuelve los genes más diferencialmente expresados.

Después, filtramos el resultado de dataDEGs con `abs(LogFC) >= 1`, y utilizamos la función `TCGAanalyze_levelTab`para crear una tabla con DEGs (genes diferencialmente expresados), log Fol Change (FC), false discovery rate (FDR), el nivel de expresión génica para las muestras de la condición 1 y la condición 2 y el valor Delta.

Filtrado Queremos una variabilidad mayor del percentil 75 y hacer el experimento con estos genes

```
#quantile filter of genes
dim(dataNorm)
```

```
## [1] 19586   606
```

```
dataFilt2 <- TCGAanalyze_Filtering(tabDF = dataNorm, method = "quantile", qnt.cut = 0.75)
dim(dataFilt2)
```

```
## [1] 4897  606
```

El archivo resultante `dataFilt` se ha quedado en un tamaño de 23 Mb (2967582 elementos), mientras que el archivo con los datos normalizados pesaba 91.8 Mb.

```
dataFilt2[1:10, 1:3]
```

```
##       TCGA-B0-5694-01A-11R-1541-07 TCGA-CJ-4637-01A-02R-1325-07
## A2M                          71106                         47587
## AAMP                          4536                          4774
## AARS                          4255                          6358
## ABAT                           703                           951
## ABCA1                         1960                          5553
## ABCA2                         2298                          3403
## ABCA3                         2663                          3541
## ABCF1                         1669                          3168
## ABL1                          2211                          3642
## ABP1                         18841                          9079
##       TCGA-CZ-4860-01A-01R-1305-07
## A2M                          39078
## AAMP                         11681
## AARS                         30018
## ABAT                           302
## ABCA1                         6526
## ABCA2                         1892
## ABCA3                         4032
## ABCF1                         3886
## ABL1                          4225
## ABP1                         19257
```

Separación de muestras Si queremos seguir con el DEA tendremos que separar las muestras según nuestra variable de interés. En nuestro caso, la variable de interés es vital_status.

```r
samplesD <- subset(se$barcode, se$vital_status == "Dead")
samplesA <- subset(se$barcode, se$vital_status == "Alive")
```

Diff.expr.analysis (DEA) Vamos a tener en cuenta qué significa cada uno de los parámetros que se incluyen como argumentos en la función. El `log2FoldChange` es el efecto de muestra estimado, es decir, cuánto parece haber cambiado la expresión génica debido al grupo al que pertenece la muestra o tipo de infiltración. Este valor tiene una incertidumbre que se ve reflejada en la columna lfcSE.

```r
# Diff.expr.analysis (DEA)
dataDEGs <- TCGAanalyze_DEA(mat1 = dataFilt2[,samplesD],
                            mat2 = dataFilt2[,samplesA],
                            Cond1type = "Dead",
                            Cond2type = "Alive",
                            fdr.cut = 0.10,
                            logFC.cut = 0.4,
                            method = "glmLRT")
```

```
## Batch correction skipped since no factors provided

## ----------------------- DEA -------------------------------

## there are Cond1 type Dead in  202 samples

## there are Cond2 type Alive in  404 samples

## there are  4897 features as miRNA or genes

## I Need about  99 seconds for this DEA. [Processing 30k elements /s]

## ----------------------- END DEA -------------------------------
```

Tras esto, podemos crear la tabla resumen:

```r
dataDEGsFiltLevel <- TCGAanalyze_LevelTab(dataDEGs, "Dead", "Alive", dataFilt2[,samplesD], dataFilt2[,sa
```

...

```r
dim(dataDEGsFiltLevel)
```

```
## [1] 238   6
```

```r
dataDEGsFiltLevel[1:10,1:6]
```

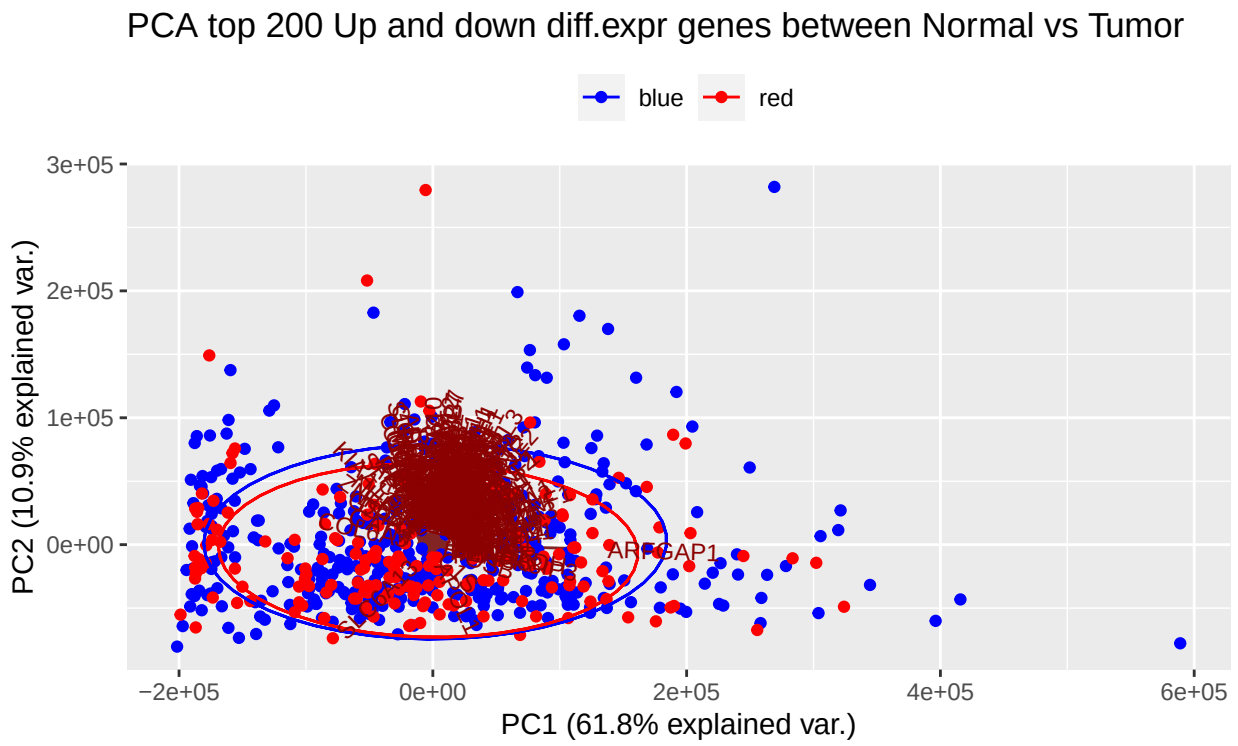
```
##              mRNA      logFC          FDR     Dead     Alive    Delta
## COL1A1     COL1A1 -0.7976182 4.930658e-07 70969.97 42538.998 56606.94
## IGF2         IGF2 -1.5526790 4.781284e-10 30784.38  9484.958 47798.26
## TGFBI       TGFBI -0.4206435 3.411698e-02 99552.24 74632.302 41876.00
## H19           H19 -1.2156873 3.859709e-10 32273.15 14034.324 39234.06
```

```
## ADAM6       ADAM6 -0.6511942 1.776480e-03 59208.42 38980.710 38556.18
## LOC96610 LOC96610 -0.7128045 9.266717e-04 41769.08 29767.351 29773.19
## COL1A2     COL1A2 -0.4418575 8.540759e-04 54407.95 41550.782 24040.56
## PDK4         PDK4  0.9063183 3.318169e-09 26507.48 49861.824 24024.21
## VWF           VWF  0.4807805 1.659783e-04 47744.58 69167.960 22954.66
## SERPINE1 SERPINE1 -0.7796567 4.964183e-07 27901.31 16303.317 21753.45
```

PCA plot

```
pca <- TCGAvisualize_PCA(dataFilt2, dataDEGsFiltLevel, ntopgenes = 200, samplesA, samplesD)
```

```
## Warning: In prcomp.default(t(expr2), cor = TRUE) :
##  extra argument 'cor' will be disregarded
```



Aunque en la gráfica ponga que se trata de genes diferencialmente expresados entre muestras tumorales y normales, se trata de muestras de pacientes fallecidos (rojo) y vivos (azul).

Expresión proteica

De acuerdo con la información mostrada en: https://www.bioconductor.org/packages/devel/bioc/vignettes/TCGAbiolinks/inst/doc/download_prepare.html y http://www.linkedomics.org/data_download/TCGA-KIRC/, los datos de expresión proteica que hemos descargado del archivo legacy se trata de los resultados de un experimento *Reverse Phase Protein Array* (RPPA).

Encontramos información sobre este tipo de técnica en el siguiente artículo:

Boellner S., Becker K.-F. Reverse Phase Protein Arrays—Quantitative Assessment of Multiple Biomarkers in Biopsies for Clinical Use. Microarrays. 2015;4:98–114. doi: 10.3390/microarrays4020098

La técnica RPPA es una tecnología utilizada para medir cuantitativamente cientos de proteínas señal en muestras biológicas y clínicas. Este formato en forma de array permite la cuantificación de proteínas o fosfoproteínas en muestras múltiples bajo las mismas condiciones experimentales de manera simultánea. Además es posible utilizarla para obtener los perfiles de transducción de señales de pequeños grupos de células en cultivo o células aisladas de biopsias humanas incluídos tejidos fijados con formaldehído o embebidos en parafina. En cuanto a **ventajas** sobre otras técnicas proteomicas como la espectroscopía de masas encontramos que la **preparación de muestras es mucho más sencilla** y que tiene una **buena detección de proteínas de señalización poco abundantes.

```
knitr::include_graphics("figures/RPPA.png")
```

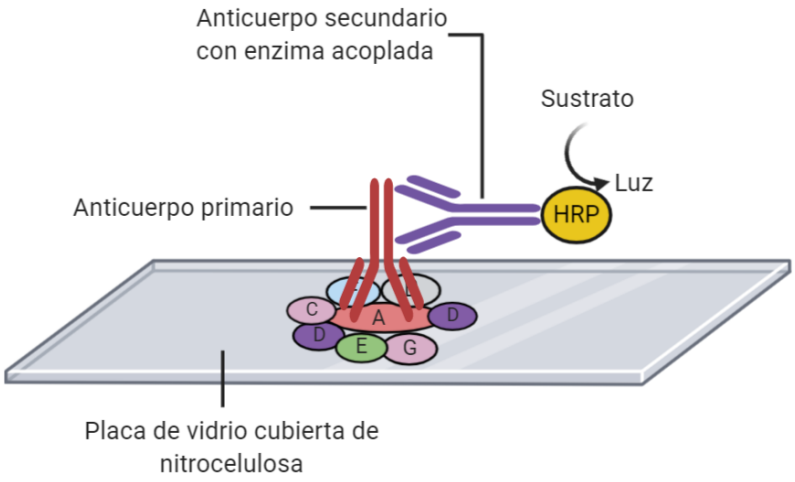


Figure 2: Presentación esquemática de un array de proteínas en fase reversa (RPPA), donde la proteína A es nuestra proteína de interés

Introducción a RPPA

source: https://bioinformatics.mdanderson.org/public-software/tcpa/

What is RPPA?

Reverse phase protein array (RPPA) is a high-throughput antibody-based technique with the procedures similar to that of Western blots. Proteins are extracted from tumor tissue or cultured cells, denatured by SDS, printed on nitrocellulose-coated slides followed by antibody probe. Our RPPA platform currently allows for the analysis of >1000 samples using at least 130 different antibodies.

What are the advantages of RPPA?

- Inexpensive, high-throughput method utilizing automation for increased quality and reliability
- Sample preparation requirements are similar to that of Western blots
- Complete assay requires only 40 microliters of each sample for 150 antibodies
- Robust quantification due to serial dilution of samples

How are the RPPA data processed?

Level 1 data

Cellular proteins are first denatured by 1% SDS (with beta mercaptoethanol) and diluted in five 2-fold dilutions in dilution buffer (lysis buffer containing 1% SDS). Serial diluted lysates are arrayed on nitrocellulose-coated slides (Grace Biolabs) by the Aushon 2470 Arrayer and probed with validated antibodies. Signals are amplified by TSA and captured by DAB colorimetric reaction. The slides are then scanned, analyzed and quantified by ArrayPro Analyzer to generate spot intensity.

Level 2 data

Based on Level 1 data, each dilution curve of spot intensities is fitted using the monotone increasing B-spline model in the SuperCurve R package. This fits a single curve using all the samples on a slide with the signal intensity as the response variable and the dilution steps as independent variables. The fitted curve is plotted with the signal intensities on the y-axis and the log2-concentration of proteins on the x-axis for diagnostic purposes.

Level 3 data

Based on Level 2 data, the data normalization is processed as follows:

1. Calculate the median for each protein across all the samples.
2. Subtract the median (from step 1) from values within each protein.
3. Calculate the median for each sample across all proteins.
4. Subtract the median (from step 3) from values within each sample.

How do we quantify protein expression and modification?

We use the approach of "SuperCurve Fitting" developed by the Department of Bioinformatics and Computational Biology at MD Anderson Cancer Center to quantify protein expression and modification. Briefly, a "standard curve" is constructed from 5808 spots on each slide (one slide probed for one antibody). These spots include 5 serial dilutions of each sample plus 528 QC spots of standard lysates at different concentrations. Relative levels of protein expression and modification for each sample are determined by interpolation of each dilution curve to the "standard curve" (supercurve) of the slide (antibody).

Procedimiento de RPPA

Para utilizar RPPA en la rutina clínica, esta técnica tiene que ser reporducible y fácil de adaptar a pruebas de alto rendimiento. Entre los pasos para la realización de este microarray de proteínas enumeramos:

1- Preparación de muestras 2- Validación de anticuerpos 3- Spotting 4- Detección de señal 5- Análisis de datos

```
knitr::include_graphics("figures/RPPAworkflow.png")
```

En esta técnica es importante utilizar anticuerpos específicos que se unan a la proteína de interés obteniendo una única banda a un peso molecular correcto (análisis de anticuerpos mediante Western Blot). Los anticuerpos se comprueban primero con un WB con lisados de 10 o más líneas celulares y después para lisados de nuestra muestra de interés.

En cuanto a la fase de Spotting, nuestra placa está recubierta de nitrocelulosa y tiene un tamaño de 7 cm x 2 cm. Para los estudios tipo RPPA se necesitan volúmenes muy pequeños de muestra (1 nL de una disolución 2 ng/nL). Tras inmovilizar las proteínas en la cuadrícula, estas son detectadas con anticuerpos cuya especificidad por el antigeno ha sido validada. La señal puede ser de muchos tipos (tinciones cercanas al infrarrojo, chivatos cromogénicos, quimioluminiscencia...). Se utiliza el softwaere MicroVigene array software para tratar las imagenes obtenidas. Dado que se han hecho diluciones seriales a la hora de realizar el spotting, podemos cuantificar los niveles de proteína para cada muestra.

Figure 3: Workflow de los estudios de RPPA

Análisis

```
ExprProtKIRC
```

```
str(ExprProtKIRC)
```

```
summary(ExprProtKIRC)
```

Vamos a comprobar si existen duplicados dentro del dataset:

```
nrow(ExprProtKIRC[duplicated(ExprProtKIRC), ])
```

```
## [1] 0
```

Y si hay misissing values:

```
sapply(ExprProtKIRC, function(x) sum(is.na(x)))
```

Nos aparecen los Nas por columna. Vemos que casi todas las columnas tienen unos 41 NAs. Por lo que estas filas de genes con NAs tenemos que omitirlas. Utilizaremos `na.omit()` que devuelve el objeto con los casos incompletos retirados.

```
datos <- na.omit(ExprProtKIRC)
```

Ahora tenemos 177 genes.

Por otro lado, tenemos que fijarnos también en las muestras y compararlas con los datos de expresión génica. Guardamos el nombre de las columnas de `ExprProtKIRC` en `ExprProtColAn`, por si acaso necesitamos este vector en algún momento.

```
ExprProtColAn <- colnames(ExprProtKIRC)[2:479]
```

Vamos a cambiar el nombre de las columnas de `ExprProtKIRC` que son del estilo `"TCGA-B4-5835-01A-13-1742-20"` por el nombre acortado de las muestras. Así podremos coger justo las muestras que coincidan entre `ExprProtKIRC` y `ExpGenKIRC3$sample`. Para eso necesitaremos un bucle `for`. Las columnas renombradas se guardarán en el vector `muestras`.

```
muestras <- c()
for (j in colnames(ExprProtKIRC)[2:479]){
  muestras <- c(muestras, sub("(.*-.*-.*-.*)-.*-.*-.*", "\\1", j))
}
muestras
```

```
##   [1] "TCGA-A3-3316-01A" "TCGA-BP-4970-01A" "TCGA-CJ-4884-01A"
##   [4] "TCGA-B8-A54D-01A" "TCGA-BP-4334-01A" "TCGA-B0-5702-01A"
##   [7] "TCGA-A3-3323-01A" "TCGA-CW-5588-01A" "TCGA-BP-4327-01A"
##  [10] "TCGA-CJ-5682-01A" "TCGA-B8-4143-01A" "TCGA-A3-3347-01A"
##  [13] "TCGA-CJ-4905-01A" "TCGA-A3-3346-01A" "TCGA-AK-3434-01A"
##  [16] "TCGA-B8-5164-01A" "TCGA-CZ-5985-01A" "TCGA-BP-4354-01A"
##  [19] "TCGA-CZ-4862-01A" "TCGA-BP-4993-01A" "TCGA-CZ-4863-01A"
##  [22] "TCGA-AK-3428-01A" "TCGA-B0-4836-01A" "TCGA-CJ-5679-01A"
##  [25] "TCGA-B0-4688-01A" "TCGA-B0-4694-01A" "TCGA-B0-4701-01A"
##  [28] "TCGA-B8-4622-01A" "TCGA-G6-A8L8-01A" "TCGA-CJ-4912-01A"
##  [31] "TCGA-A3-3319-01A" "TCGA-B0-5100-01A" "TCGA-CJ-4885-01A"
##  [34] "TCGA-B0-5092-01A" "TCGA-BP-4174-01A" "TCGA-BP-5190-01A"
##  [37] "TCGA-CJ-4870-01A" "TCGA-B0-4844-01A" "TCGA-CJ-4899-01A"
##  [40] "TCGA-DV-5565-01A" "TCGA-B0-4713-01A" "TCGA-BP-4968-01A"
##  [43] "TCGA-BP-5200-01A" "TCGA-CJ-4897-01A" "TCGA-B0-5399-01A"
##  [46] "TCGA-EU-5906-01A" "TCGA-CZ-5461-01A" "TCGA-CZ-4854-01A"
##  [49] "TCGA-BP-4974-01A" "TCGA-CZ-5455-01A" "TCGA-A3-3326-01A"
##  [52] "TCGA-CZ-4861-01A" "TCGA-B8-A54E-01A" "TCGA-CZ-4857-01A"
##  [55] "TCGA-CJ-4891-01A" "TCGA-BP-5199-01A" "TCGA-B4-5843-01A"
##  [58] "TCGA-6D-AA2E-01A" "TCGA-B8-4621-01A" "TCGA-CZ-4860-01A"
##  [61] "TCGA-CZ-5982-01A" "TCGA-B2-5633-01A" "TCGA-AK-3444-01A"
##  [64] "TCGA-B2-3924-01A" "TCGA-B2-4102-01A" "TCGA-BP-5001-01A"
##  [67] "TCGA-B8-5158-01A" "TCGA-A3-3357-01A" "TCGA-B0-4714-01A"
##  [70] "TCGA-A3-3320-01A" "TCGA-CZ-4858-01A" "TCGA-BP-4347-01A"
##  [73] "TCGA-B0-4833-01A" "TCGA-CW-6096-01A" "TCGA-AK-3456-01A"
##  [76] "TCGA-B8-A7U6-01A" "TCGA-B0-5095-01A" "TCGA-BP-4329-01A"
##  [79] "TCGA-B4-5836-01A" "TCGA-BP-5185-01A" "TCGA-A3-3308-01A"
##  [82] "TCGA-BP-4331-01A" "TCGA-EU-5905-01A" "TCGA-B0-4838-01A"
##  [85] "TCGA-BP-5191-01A" "TCGA-B0-5106-01A" "TCGA-BP-4173-01A"
##  [88] "TCGA-BP-4158-01A" "TCGA-B0-5701-01A" "TCGA-B0-5113-01A"
##  [91] "TCGA-B0-4700-01A" "TCGA-CZ-5463-01A" "TCGA-MM-A563-01A"
##  [94] "TCGA-B0-5097-01A" "TCGA-CJ-4634-01A" "TCGA-CW-5584-01A"
##  [97] "TCGA-BP-4798-01A" "TCGA-B0-4718-01A" "TCGA-B0-4841-01A"
## [100] "TCGA-CJ-6027-01A" "TCGA-B0-4810-01A" "TCGA-B0-4693-01A"
## [103] "TCGA-A3-3329-01A" "TCGA-B0-5120-01A" "TCGA-A3-3317-01A"
## [106] "TCGA-B8-5546-01A" "TCGA-BP-5195-01A" "TCGA-DV-5576-01A"
## [109] "TCGA-CJ-5675-01A" "TCGA-B0-4690-01A" "TCGA-CJ-4923-01A"
## [112] "TCGA-B8-5162-01A" "TCGA-B8-5550-01A" "TCGA-CJ-5686-01A"
## [115] "TCGA-DV-5574-01A" "TCGA-CZ-5984-01A" "TCGA-A3-3322-01A"
## [118] "TCGA-B0-4817-01A" "TCGA-BP-5173-01A" "TCGA-BP-4965-01A"
## [121] "TCGA-BP-4776-01A" "TCGA-CZ-5460-01A" "TCGA-BP-4987-01A"
## [124] "TCGA-B4-5838-01A" "TCGA-BP-4763-01A" "TCGA-BP-4784-01A"
## [127] "TCGA-B0-5109-01A" "TCGA-A3-3311-01A" "TCGA-BP-5176-01A"
## [130] "TCGA-CW-5589-01A" "TCGA-B0-4814-01A" "TCGA-DV-A4VX-01A"
## [133] "TCGA-BP-4164-01A" "TCGA-BP-4765-01A" "TCGA-B0-5094-01A"
## [136] "TCGA-B8-A54F-01A" "TCGA-BP-4963-01A" "TCGA-B8-5165-01A"
## [139] "TCGA-BP-4344-01A" "TCGA-BP-4995-01A" "TCGA-B0-5400-01A"
```

```
## [142] "TCGA-CZ-5468-01A" "TCGA-BP-4771-01A" "TCGA-CJ-4872-01A"
## [145] "TCGA-CJ-6028-01A" "TCGA-BP-4769-01A" "TCGA-A3-3363-01A"
## [148] "TCGA-B0-4842-01A" "TCGA-BP-4807-01A" "TCGA-B0-5692-01A"
## [151] "TCGA-A3-3362-01A" "TCGA-B0-5099-01A" "TCGA-B0-4852-01A"
## [154] "TCGA-B0-5117-01A" "TCGA-CJ-4903-01A" "TCGA-3Z-A93Z-01A"
## [157] "TCGA-B8-A8YJ-01A" "TCGA-BP-4766-01A" "TCGA-BP-5184-01A"
## [160] "TCGA-B0-4691-01A" "TCGA-A3-3385-01A" "TCGA-EU-5907-01A"
## [163] "TCGA-BP-4976-01A" "TCGA-BP-4985-01A" "TCGA-CZ-5986-01A"
## [166] "TCGA-CZ-5456-01A" "TCGA-CZ-4864-01A" "TCGA-BP-4355-01A"
## [169] "TCGA-B0-4848-01A" "TCGA-BP-4343-01A" "TCGA-CZ-4865-01A"
## [172] "TCGA-CW-5587-01A" "TCGA-BP-4797-01A" "TCGA-CJ-4871-01A"
## [175] "TCGA-B4-5834-01A" "TCGA-B0-5713-01A" "TCGA-CJ-4908-01A"
## [178] "TCGA-CJ-4907-01A" "TCGA-B0-4827-01A" "TCGA-A3-3373-01A"
## [181] "TCGA-BP-4789-01A" "TCGA-DV-5566-01A" "TCGA-CZ-5467-01A"
## [184] "TCGA-B0-5712-01A" "TCGA-B8-5549-01A" "TCGA-A3-3387-01A"
## [187] "TCGA-BP-5198-01A" "TCGA-A3-3349-01A" "TCGA-B8-5552-01B"
## [190] "TCGA-B0-4839-01A" "TCGA-B0-4821-01A" "TCGA-CZ-5454-01A"
## [193] "TCGA-CJ-4888-01A" "TCGA-CJ-4886-01A" "TCGA-CZ-5988-01A"
## [196] "TCGA-BP-4960-01A" "TCGA-AK-3427-01A" "TCGA-B0-4813-01A"
## [199] "TCGA-B0-5402-01A" "TCGA-CZ-5989-01A" "TCGA-CJ-4904-01A"
## [202] "TCGA-BP-4977-01A" "TCGA-BP-5007-01A" "TCGA-B8-A54J-01A"
## [205] "TCGA-B0-4703-01A" "TCGA-BP-5178-01A" "TCGA-CZ-4853-01A"
## [208] "TCGA-BP-4330-01A" "TCGA-DV-A4VZ-01A" "TCGA-B0-5711-01A"
## [211] "TCGA-CJ-4895-01A" "TCGA-A3-3374-01A" "TCGA-BP-4803-01A"
## [214] "TCGA-B0-4823-01A" "TCGA-BP-4325-01A" "TCGA-A3-3376-01A"
## [217] "TCGA-CJ-4881-01A" "TCGA-B0-4843-01A" "TCGA-B2-4099-01A"
## [220] "TCGA-MM-A564-01A" "TCGA-CZ-5458-01A" "TCGA-B0-4834-01A"
## [223] "TCGA-B0-4847-01A" "TCGA-A3-3313-01A" "TCGA-BP-4349-01A"
## [226] "TCGA-CJ-4635-01A" "TCGA-BP-4959-01A" "TCGA-B0-4811-01A"
## [229] "TCGA-BP-4756-01A" "TCGA-BP-5180-01A" "TCGA-B4-5832-01A"
## [232] "TCGA-CW-6090-01A" "TCGA-BP-4351-01A" "TCGA-B0-5116-01A"
## [235] "TCGA-BP-4971-01A" "TCGA-A3-3380-01A" "TCGA-CZ-5459-01B"
## [238] "TCGA-BP-4161-01A" "TCGA-B8-A54K-01A" "TCGA-CJ-5671-01A"
## [241] "TCGA-BP-4981-01A" "TCGA-BP-5168-01A" "TCGA-DV-5575-01A"
## [244] "TCGA-CJ-4916-01A" "TCGA-CZ-5987-01A" "TCGA-B0-4696-01A"
## [247] "TCGA-CJ-5672-01A" "TCGA-BP-4964-01A" "TCGA-BP-4346-01A"
## [250] "TCGA-CZ-4866-01A" "TCGA-BP-5008-01A" "TCGA-BP-5174-01A"
## [253] "TCGA-CZ-5457-01A" "TCGA-BP-5192-01A" "TCGA-A3-3383-01A"
## [256] "TCGA-BP-4983-01A" "TCGA-BP-4169-01A" "TCGA-CJ-5678-01A"
## [259] "TCGA-B0-4845-01A" "TCGA-B0-4815-01A" "TCGA-B0-5077-01A"
## [262] "TCGA-BP-5177-01A" "TCGA-B4-5378-01A" "TCGA-CZ-5464-01A"
## [265] "TCGA-BP-5004-01A" "TCGA-B0-5115-01A" "TCGA-BP-5187-01A"
## [268] "TCGA-B2-5636-01A" "TCGA-BP-4969-01A" "TCGA-CJ-5689-01A"
## [271] "TCGA-BP-4801-01A" "TCGA-CJ-4875-01A" "TCGA-BP-4961-01A"
## [274] "TCGA-CJ-6030-01A" "TCGA-B0-5812-01A" "TCGA-B8-A54G-01A"
## [277] "TCGA-B2-3923-01A" "TCGA-CJ-4882-01A" "TCGA-BP-4345-01A"
## [280] "TCGA-B0-4846-01A" "TCGA-B8-5551-01A" "TCGA-CW-5591-01A"
## [283] "TCGA-B0-4699-01A" "TCGA-BP-4335-01A" "TCGA-B0-4849-01A"
## [286] "TCGA-CZ-5469-01A" "TCGA-B0-4697-01A" "TCGA-B0-5121-01A"
## [289] "TCGA-B0-5698-01A" "TCGA-CJ-6031-01A" "TCGA-CJ-4876-01A"
## [292] "TCGA-BP-5186-01A" "TCGA-CW-6088-01A" "TCGA-A3-3372-01A"
## [295] "TCGA-CZ-5465-01A" "TCGA-CJ-4890-01A" "TCGA-BP-4799-01A"
## [298] "TCGA-B0-4828-01A" "TCGA-CJ-4874-01A" "TCGA-A3-3382-01A"
## [301] "TCGA-BP-4962-01A" "TCGA-B0-5081-01A" "TCGA-B8-5553-01A"
```

```
## [304] "TCGA-B8-5545-01A" "TCGA-G6-A5PC-01A" "TCGA-CJ-6033-01A"
## [307] "TCGA-B0-4945-01A" "TCGA-B0-5694-01A" "TCGA-BP-5202-01A"
## [310] "TCGA-CJ-5677-01A" "TCGA-A3-3351-01A" "TCGA-BP-5196-01A"
## [313] "TCGA-CJ-4901-01A" "TCGA-CZ-5452-01A" "TCGA-B8-5163-01A"
## [316] "TCGA-B0-5706-01A" "TCGA-B0-5075-01A" "TCGA-B0-5690-01A"
## [319] "TCGA-BP-4337-01A" "TCGA-B0-5710-01A" "TCGA-BP-4775-01A"
## [322] "TCGA-BP-4988-01A" "TCGA-BP-5169-01A" "TCGA-BP-5201-01A"
## [325] "TCGA-CJ-4894-01A" "TCGA-CJ-4868-01A" "TCGA-BP-4777-01A"
## [328] "TCGA-BP-4973-01A" "TCGA-CW-5580-01A" "TCGA-CJ-4913-01A"
## [331] "TCGA-B8-4620-01A" "TCGA-BP-4989-01A" "TCGA-B0-5108-01A"
## [334] "TCGA-DV-5567-01A" "TCGA-B0-5104-01A" "TCGA-BP-4975-01A"
## [337] "TCGA-A3-3335-01A" "TCGA-BP-5183-01A" "TCGA-CJ-4638-01A"
## [340] "TCGA-CJ-4887-01A" "TCGA-BP-4787-01A" "TCGA-B0-5709-01A"
## [343] "TCGA-BP-4781-01A" "TCGA-BP-4998-01A" "TCGA-G6-A8L7-01A"
## [346] "TCGA-B0-4712-01A" "TCGA-CW-5585-01A" "TCGA-AK-3451-01A"
## [349] "TCGA-BP-4790-01A" "TCGA-B0-4710-01A" "TCGA-A3-3352-01A"
## [352] "TCGA-B0-4819-01A" "TCGA-B0-5102-01A" "TCGA-B0-4822-01A"
## [355] "TCGA-EU-5904-01A" "TCGA-CZ-5470-01A" "TCGA-B0-5695-01A"
## [358] "TCGA-B0-5707-01A" "TCGA-BP-5182-01A" "TCGA-B0-5703-01A"
## [361] "TCGA-B2-4098-01A" "TCGA-BP-4768-01A" "TCGA-B0-5085-01A"
## [364] "TCGA-B4-5377-01A" "TCGA-B0-4816-01A" "TCGA-CW-6097-01A"
## [367] "TCGA-A3-3370-01A" "TCGA-B0-5110-01A" "TCGA-BP-4165-01A"
## [370] "TCGA-B0-4698-01A" "TCGA-BP-5170-01A" "TCGA-BP-5189-01A"
## [373] "TCGA-CJ-4878-01A" "TCGA-BP-4332-01A" "TCGA-B0-5107-01A"
## [376] "TCGA-DV-5569-01A" "TCGA-DV-A4W0-01A" "TCGA-G6-A8L6-01A"
## [379] "TCGA-CJ-4893-01A" "TCGA-CJ-4900-01A" "TCGA-BP-4760-01A"
## [382] "TCGA-BP-5181-01A" "TCGA-B0-4706-01A" "TCGA-BP-4774-01A"
## [385] "TCGA-CW-6093-01A" "TCGA-BP-4770-01A" "TCGA-CJ-5680-01A"
## [388] "TCGA-GK-A6C7-01A" "TCGA-B0-4824-01A" "TCGA-CZ-5451-01A"
## [391] "TCGA-DV-5568-01A" "TCGA-B8-A54I-01A" "TCGA-DV-5573-01A"
## [394] "TCGA-BP-4804-01A" "TCGA-B0-5096-01A" "TCGA-B2-A4SR-01A"
## [397] "TCGA-CJ-4889-01A" "TCGA-BP-4758-01A" "TCGA-A3-3306-01A"
## [400] "TCGA-CJ-4869-01A" "TCGA-B0-5119-01A" "TCGA-B8-4146-01B"
## [403] "TCGA-B0-5691-01A" "TCGA-BP-5006-01A" "TCGA-B0-5697-01A"
## [406] "TCGA-CJ-4873-01A" "TCGA-CZ-4856-01A" "TCGA-B0-5098-01A"
## [409] "TCGA-CJ-5683-01A" "TCGA-AK-3436-01A" "TCGA-BP-4338-01A"
## [412] "TCGA-B0-4818-01A" "TCGA-CJ-6032-01A" "TCGA-CZ-5462-01A"
## [415] "TCGA-CJ-4918-01A" "TCGA-BP-4994-01A" "TCGA-BP-4341-01A"
## [418] "TCGA-CJ-5681-01A" "TCGA-B0-5696-01A" "TCGA-B2-5639-01A"
## [421] "TCGA-A3-3365-01A" "TCGA-BP-5194-01A" "TCGA-CJ-4640-01A"
## [424] "TCGA-CJ-5684-01A" "TCGA-B0-5088-01A" "TCGA-BP-4762-01A"
## [427] "TCGA-BP-4991-01A" "TCGA-CZ-4859-01A" "TCGA-CW-5581-01A"
## [430] "TCGA-CJ-4920-01A" "TCGA-B2-4101-01A" "TCGA-A3-3367-01A"
## [433] "TCGA-BP-5009-01A" "TCGA-MW-A4EC-01A" "TCGA-BP-4992-01A"
## [436] "TCGA-A3-3378-01A" "TCGA-CZ-5453-01A" "TCGA-BP-4967-01A"
## [439] "TCGA-B0-4707-01A" "TCGA-CJ-4642-01B" "TCGA-BP-5010-01A"
## [442] "TCGA-A3-3331-01A" "TCGA-BP-4986-01A" "TCGA-BP-4326-01A"
## [445] "TCGA-BP-4170-01A" "TCGA-CJ-5676-01A" "TCGA-A3-3324-01A"
## [448] "TCGA-B8-A54H-01A" "TCGA-BP-5000-01A" "TCGA-CJ-4892-01A"
## [451] "TCGA-CW-6087-01A" "TCGA-BP-4759-01A" "TCGA-B0-5700-01A"
## [454] "TCGA-B0-5705-01A" "TCGA-B0-5693-01A" "TCGA-BP-5175-01A"
## [457] "TCGA-B4-5844-01A" "TCGA-BP-4761-01A" "TCGA-BP-4342-01A"
## [460] "TCGA-CW-5590-01A" "TCGA-B2-5641-01A" "TCGA-A3-3336-01A"
## [463] "TCGA-BP-4352-01A" "TCGA-B0-5084-01A" "TCGA-CJ-4902-01A"
```

```
## [466] "TCGA-BP-4972-01A" "TCGA-B0-5699-01A" "TCGA-BP-4999-01A"
## [469] "TCGA-BP-4982-01A" "TCGA-BP-4353-01A" "TCGA-B2-5635-01A"
## [472] "TCGA-B4-5835-01A" "TCGA-CW-5583-01A" "TCGA-B8-5159-01A"
## [475] "TCGA-B0-4837-01A" "TCGA-B8-4153-01B" "TCGA-B0-5080-01A"
## [478] "TCGA-CZ-5466-01A"
```

Ahora vamos a recuperar los índices en el vector de Expresión Génica `se$sample` de los elementos del vector `muestras`que coincidan, y guardamos estos índices en el vector `index`.

```
index <- c()
for (i in muestras){
  index <- c(index, which(se$sample %in% i))
}
str(index)
```

```
##  int [1:474] 161 546 202 114 235 595 25 52 38 593 ...
```

Vemos que el vector index tiene 474 elementos, sin embargo el vector muestras tenía 478 elementos, por lo que 4 muestras del dataset de Expresión Proteica no se encuentran en el dataset de Expresión Génica ¿Qué muestras de `ExprProtKIRC` no se encuentran en `ExpGenKIRC3`?

```
which(match(muestras, se$sample, nomatch = "0") %in% 0)
```

```
## [1]  74 237 330 462
```

Ahora tenemos que quitar de los datos de expresión génica los que no se encuentran en los datos de expresión proteica y viceversa. De nuestro vector `muestras`queremos quitar los elementos que se encuentran en los índices 74, 237, 330 y 462.

```
ncol(datos)
```

```
## [1] 479
```

```
k <- sort(c(74,237,330,462), decreasing = TRUE)
for (k in k){
  datos[k] <- NULL
}
ncol(datos)
```

```
## [1] 475
```

```
length(muestras)
```

```
## [1] 478
```

```
k <- c(74,237,330,462)
muestras <- muestras[-k]
length(muestras)
```

```
## [1] 474
```

Ahora podemos conseguir las etiquetas de `vital_status` para los datos de expresión Proteica.

```
ExpProtLabels <- c()
for (i in index){
  ExpProtLabels <- c(ExpProtLabels, se$vital_status[i])
}
```

Ahora vamos a quitar las muestras de Expresión Génica que no estén en Expresión Proteica

```
index2 <- sort(which(match(se$sample, muestras,  nomatch = "0") %in% 0), decreasing = TRUE)
ncol(dataNorm)
```

```
## [1] 606
```

```
for (k in index2){
  dataNorm <- dataNorm[, -k]
}

ncol(dataNorm)
```

```
## [1] 474
```

```
index3 <- sort(which(match(se$sample, muestras,  nomatch = "0") %in% 0), decreasing = TRUE)
ncol(dataNormt)
```

```
## [1] 606
```

```
for (k in index3){
  dataNormt <- dataNormt[, -k]
}
ncol(dataNormt)
```

```
## [1] 474
```

Vemos que ahora tanto el archivo de Expresión proteica `datos` como el de los datos de Expresión Génica normalizados `dataNorm` y transformados `dataNormt`, tienen todos 474 muestras.

Ahora conseguimos las etiquetas de `vital_status` para los datos de expresión génica.

```
'%nin%' <- Negate('%in%')
index3 <- sort(which(match(se$barcode, colnames(dataNorm),  nomatch = "0") %nin% 0), decreasing = TRUE)
ExpGenLabels <- c()
for (i in index3){
  ExpGenLabels <- c(ExpGenLabels, se$vital_status[i])
}
```

Con `dim(datos)` vemos que tenemos un archivo de expresión proteica con 475 observaciones y 177 proteínas, pero ¿cuántas de esas 177 proteínas se corresponden con los 238 genes que hemos obtenido en el análisis de expresión diferencial de los datos de expresión génica?

```r
dim(datos)
```

```
## [1] 177 475
```

Análisis Epigénetico

Introducción

La metilación del ADN es un componente importante en numerososo procesos celulares como en el desarrollo embrionario, huella genética, inactivación del cromosoma X y la conservación de la estabilidad cromosómica (Phillips 2008).

En mamíferos la metilación del ADN es escasa pero se extiende por todo el genoma distribuyendose en secuencias CpG (CGIs) que son secuencias pequeñas de ADN interespaciadas que están enriquecidas en GC. Estas islas se encuentran normalmente en sitios de iniciación dde la transcripción y su metilación conlleva al silenciamiento de los genes.

Por lo que, la investigación de la metilación del ADN es crucial para entender las redes de regulación génica en cáncer dado que la metilación de ADN reprime la transcripción. La detección de DMR (*Differentially Methylation Region*), regiones diferencialmente metiladas, pueden ayudarnos a investigar mecanismos de regulación génica.

En esta sección se describe el análisis de metilación del ADN utilizando el paquete de Bioconductor TCGAbiolinks.

Ver más: https://www.bioconductor.org/packages/release/workflows/vignettes/TCGAWorkflow/inst/doc/TCGAWorkflow.html#epigenetic_analysis

Obtención de los datos

Mirar: https://www.researchgate.net/post/How_to_handle_big_data_sets_10GB_in_R_Bioconductor_packages. Es imposible conseguir el dataset entero porque no tenemos tanta RAM. Utilizaremos un Dataset más pequeño. Se hará uso de la función `matchedMetExp()` que obtiene los barcodes de muestras con datos tanto de metilación como de expresión génica para un proyecto de TCGA. A partir de aquí el código estará comentado ya que, a pesar de que hemos conseguido obtener el objeto de metilación en R, a la hora de realizar el archivo Markdown nos vuelve a dar problemas de RAM.

```r
#setwd("D:/datos")
#samples <- matchedMetExp("TCGA-KIRC", n = 166)
#query <- GDCquery(project = c("TCGA-KIRC"),
                  #data.category = "DNA methylation",
                  #platform = "Illumina Human Methylation 450",
                  #legacy = TRUE,
                  #barcode = samples)
#GDCdownload(query)
#met <- GDCprepare(query, save = FALSE)
```

Hemos puesto un argumento de `n=166`, que es el límite para que nuestro ordenador pueda crear el objeto `met`.

Preprocesado

Los datos obtenidos necesitan un preprocesado. Los datos de metilación de la plataforma 450K tienen tres tipos de sondas cg (CpG loci), ch (no CpG loci) y rs (ensayo SNP). El último tipo puede usarse para identificación de las muestras y debe quitarse del análisis de metilación diferencial según el manual de Illumina. Por lo que tendremos que quitar las sondas rs. También tendremos que quitar las sondas de los cromosomas X e Y para eliminar errores debidos a la presencia de una proporción de hombres y mujeres diferente. El último paso de preprocesado es quitar las sondas con al menos un valor de NA.

Tras todo este preoprocesado utilizaremos la función `TCGAvisualize_meanMethylation`, que nos permite mirar la media de la metilación del ADN de cada paciente para cada grupo.

Primero vamos a echarle un vistazo a nuestro archivo `met` que es un objeto tipo `RangedSummarizeExperiment`.

```
#met
```

Podemos sacar la tabla con datos con `assay()`.

```
#assay(met)[1:10,1:2]
```

```
#met2 <- met
#nrow(assay(met))
```

Quitar las sondas SNP

```
#SNPprobes <- grep("rs", rownames(met))
#length(SNPprobes)
```

```
#dim(met2)
#k <- sort(SNPprobes, decreasing = TRUE)
#for (k in k){
  #met2 <- met2[-k,]
#}
#dim(met2)
```

Quitar sondas de los cromosomas X e Y

```
#met2 <- subset(met2, subset = !as.character(seqnames(met2)) %in% c("chrNA", "chrX", "chrY"))
#dim(met2)
```

Quitar sondas con al menos un missing value

```
#met2 <- subset(met2, subset = (rowSums(is.na(assay(met2)))==0))
#dim(met2)
```

Explorando archivos

Ya hemos realizado el preprocesado de los datos de metilación, quitando las sondas de SNPs, las de cromosomas sexuales y las que tienen missing values.

Tras este preprocesado y utilizando la función `TCGAvisualizr_meanMethylation` podemos observar la metilación del ADN media para cada paciente en cada grupo. Este recibe como argumento un objeto `SummarizedExperiment`con los datos de metilación del ADN y los argumentos `groupCol` y `subgroupCol`que deben ser las dos columnas de la matriz de información de muestras del objeto `SummarizedExperiment`(se accede a esto mediante la función `colData`).

```
#df <- data.frame("Sample.mean" = colMeans(assay(met2), na.rm = TRUE), "groups" = met2$vital_status)
#library(ggpubr)
#ggpubr::ggboxplot(df,
                  #y = "Sample.mean",
                  #x = "groups",
                  #color = "groups",
                  #add = "jitter",
                  #ylab = expression(paste("Mean DNA methylation (", beta, "-values)")),
                  #xlab = "") +
#stat_compare_means()
```

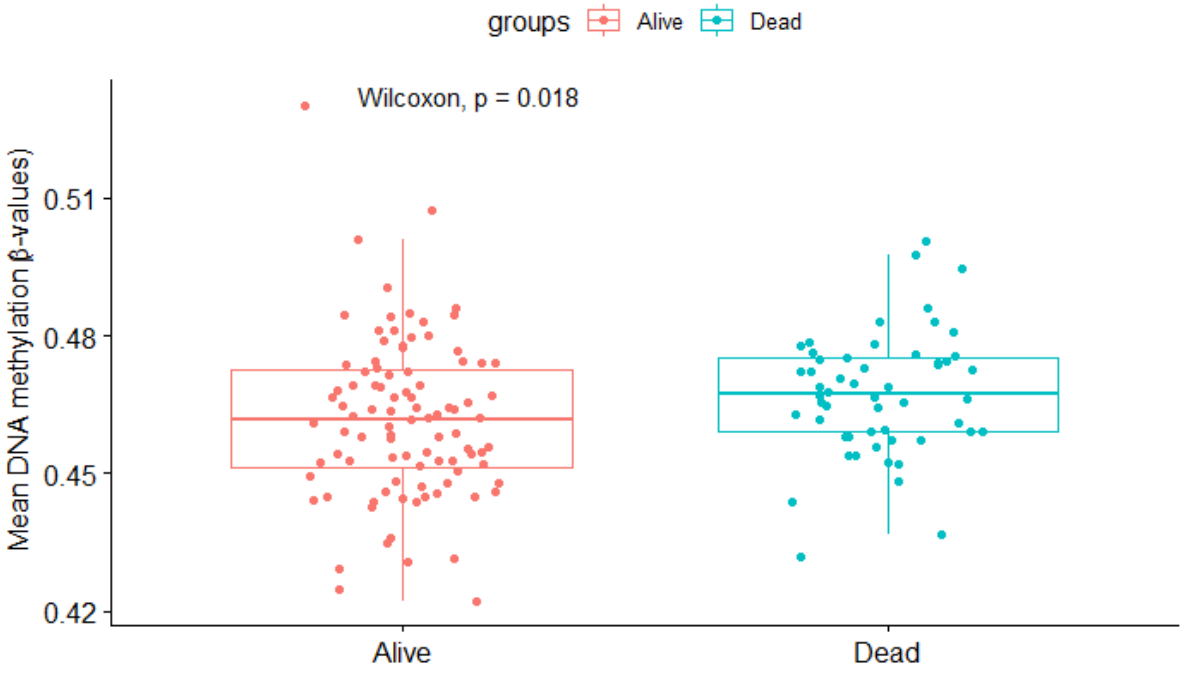


Figure 4: Gráfica la media de la metilación del ADN para cada muestra del grupo de vivos (n = 96) y muertos (n = 54)

Realizando el test de Wilcoxon, una prueba no paramétrica que compara el rango medio de dos muestras y determina si existen diferencias entre ellas, obtenemos un p = 0.018, que es menor de 0.05, por lo que rechazamos la hipótesis nula a favor de la alternativa de que las medias de metilación son distintas entre grupos.

Buscando sitios CpG diferencialmente metilados

El siguiente paso es definir sitios diferencialmente metilados entre ds grupos. Este puede realizarse utilizando la función `TCGAnalyze_DMC`. Los datos de metilación de ADN (nivel 3) están presentados en forma de valores beta que usan una escala que va desde 0 (sondas completamente sin metilar) hasta 1 (sondas completamente metiladas).

Para encontrar estos sitios diferencialmente metilados, primero, la función calcula la diferencia entre la media de la metilación del ADN (media de valores beta) de cada grupo para cada sonda. Segundo, se comprueba la expresión diferencial entre dos grupos utilizando el test de Wilcoxon ajustado por ele método de Benjamini-Hochberg. Se determinaron los argumentos de `TCGAanalyze_DMR` para requerir una diferencia mínima absoluta de valores beta de 0.15 y un p-valor ajustado menor de 0.05.

Tras estos test, se crea una gráfica tipo volcán (eje x: diferencia de medias de metilación de ADN, eje y: significancia estadística) para ayudar a los usuarios a identificar sitios CpG diferencialmente metilados y devuelven un objeto con los resultados en los rowRanges.

```
#------- Searching for differentially methylated CpG sites     ----------
#dmc <- TCGAanalyze_DMC(met2,
                        #groupCol = "vital_status", # a column in the colData matrix
                        #group1 = "Dead", # a type of the disease type column
                        #group2 = "Alive", # a type of the disease column
                        #p.cut = 0.05,
                        #diffmean.cut = 0.15,
                        #save = FALSE,
                        #legend = "State",
                        #plot.filename = "survival_metvolcano.png",
                        #cores = 1 # if set to 1 there will be a progress bar
#)
```

Este código ha tardado en correr 5 h. La figura de abajo muestra la gráfica de tipo volcán creada. Esta gráfica ayuda al usuario a seleccionar umbrales relevantes cuando buscamos regiones diferencialmente metiladas.

Podemos obtener las sondas que están hipermetiladas o hipometiladas

```
# get the probes that are Hypermethylated or Hypomethylated
# met is the same object of the section 'DNA methylation analysis'
#status.col <- "status"
#probes <- rownames(dmc)[grep("hypo|hyper",dmc$status,ignore.case = TRUE)]
#sig.met <- met2[probes,]
#probes

#Results
#[1] "cg01530101" "cg03257743" "cg05003554" "cg06822943" "cg06879394" "cg08960448" "cg09047734"
#[8] "cg09827761" "cg13608171" "cg18157012" "cg18582260" "cg19206146" "cg19473623" "cg20321706"
#[15] "cg21538684" "cg22605290" "cg23709902" "cg25542878" "cg26286961" "cg27088072"
```

Tenemos muy pocas sondas porque ya empezamos con menos muestras que en las otras ómicas, además hemos sido demasiado restrictivos a la hora de hacer el análisis diferencial de metilación.

Para visualizar los niveles de metilación de estas sondas a lo largo de las muestras, utilizamos mapas de calor que pueden generarse con el paquete de Bioconductor `complexHeatmap` (instalación: `BiocManager::install("ComplexHeatmap")`). Para crear un mapa de calor con el paquete `complexHeatmap`, el usuario debe proporcionar al menos una matriz con los niveles de metilación de ADN. También se pueden añadir capas de anotación que se pueden colocar abajo, arriba, a la izquierda o a la derecha del heatmap para tener la descripción proporcionada en los metadatos (`ClinicKIRC2`).

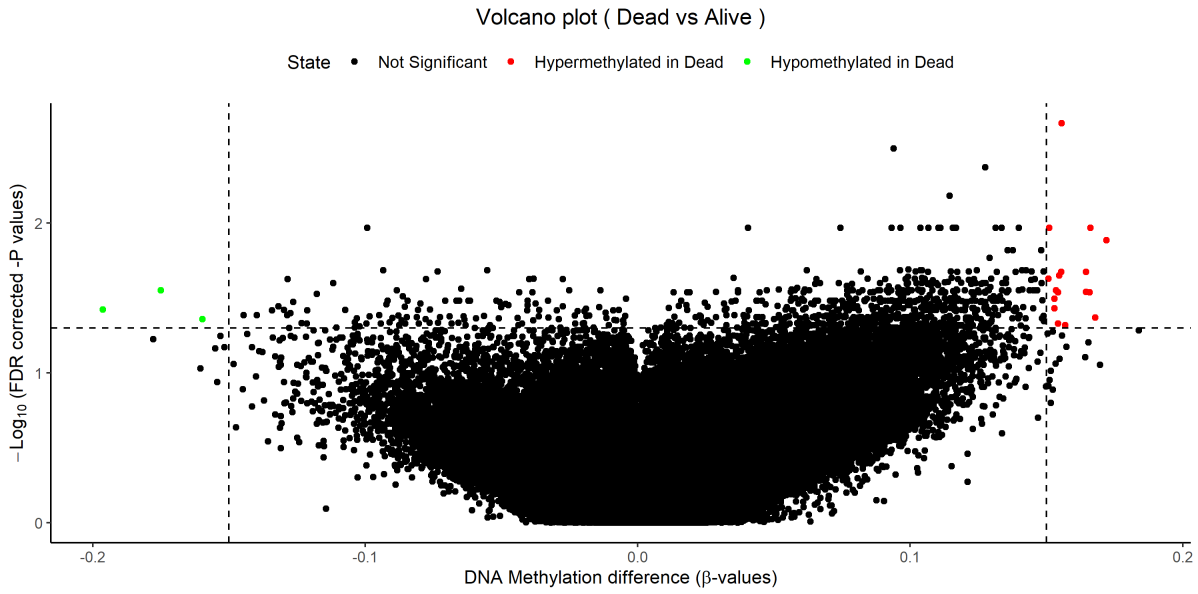


Figure 5: Volcano plot: Buscando sitios CpG diferencialmente metilados (eje x: diferencia de metilación de ADN media, eje y: significación estadística)

```
#library(ComplexHeatmap)
#clinical.order <- ClinicKIRC2[match(substr(colnames(sig.met),1,12),ClinicKIRC2$bcr_patient_barcode),]
#ta = HeatmapAnnotation(df = clinical.order[, c("vital_status", "gender", #"stage_event_pathologic_stag
                       #col = list(
                         #vital_status = c("Alive" = "grey", "Dead" = "black")
                       #))
#ra = rowAnnotation(
  #df = dmc[probes, status.col],
  #col = list(
    #"status.survival" =
      #c("Hypomethylated" = "orange",
        #"Hypermethylated" = "darkgreen")
  #),
  #width = unit(1, "cm")
#)

#heatmap  <- Heatmap(assay(sig.met),
                    #name = "DNA methylation",
                    #col = matlab::jet.colors(200),
                    #show_row_names = TRUE,
                    #cluster_rows = TRUE,
                    #cluster_columns = TRUE,
                    #show_column_names = FALSE,
                    #bottom_annotation = ta,
                    #column_title = "DNA Methylation")
#png("heatmap.png",width = 600, height = 400)
#draw(heatmap, annotation_legend_side =  "bottom")
#dev.off()
```

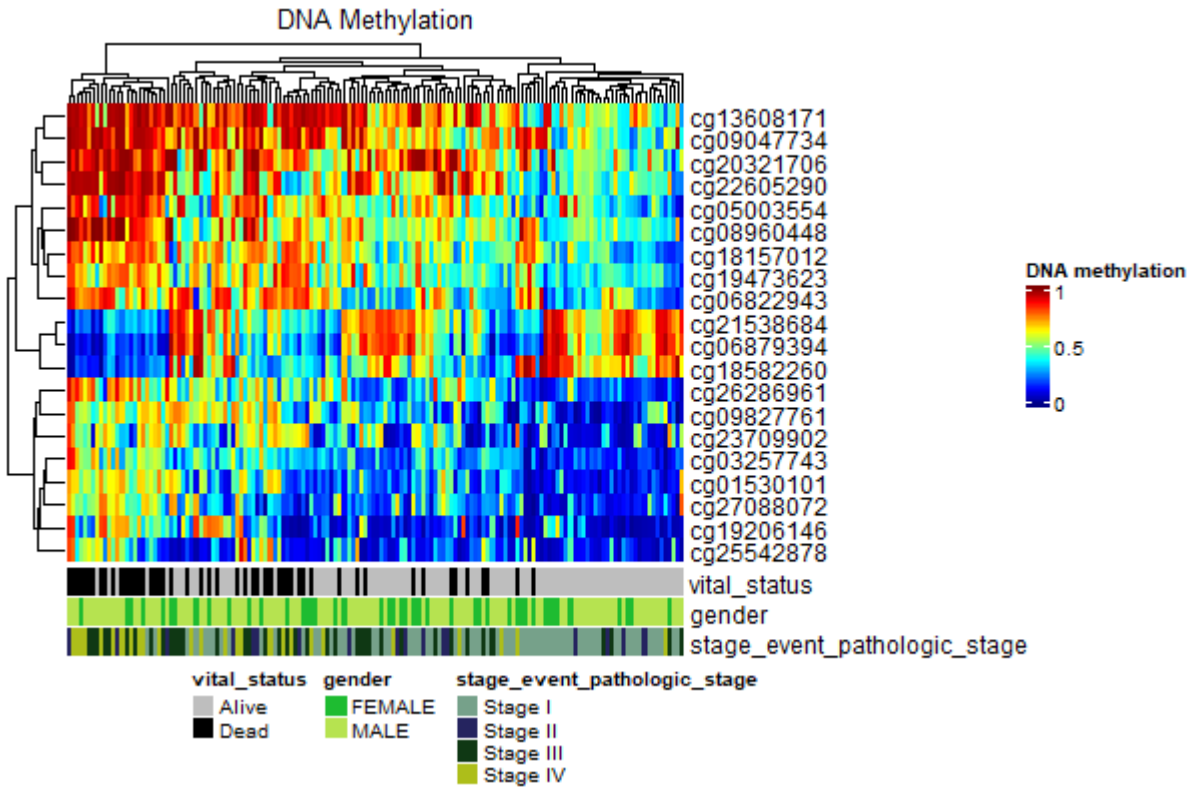


Figure 6: Heatmap: Se han agrupado tanto las muestras como las sondas en clusters. Se muestran las sondas hiper o hipometiladas para cada muestra de nuestros datos epigenéticos)

Igualando los datos de Expresión Génica, proteica y metilación

Ya teníamos igualados los datos de Expresión Génica y proteica y teniamos una n total de 474.

```
#dim(datos)
```

```
#dim(dataNorm)
```

Sin embargo, de datos de metilación al final solo teniamos una n = 150

```
#dim(met2)
```

Las muestras en el archivo de metilación tienen nombres tipo `TCGA-G6-A8L8-01A-21D-A36Y-05` mientras que en Expresión proteica `TCGA-B4-5835-01A-13-1742-20` y en Expresión génica `TCGA-B0-5694-01A-11R-1541-07`.

Vamos a cambiar el nombre de las columnas de ExprProtKIRC que son del estilo "TCGA-B4-5835-01A-13-1742-20" por el nombre acortado de las muestras. Así podremos coger justo las muestras que coincidan entre `datos` y `met2$sample`. Para eso necesitaremos un bucle for. Las columnas renombradas se guardarán en el vector muestras.

```
#muestras2 <- c()
#for (j in colnames(datos)[2:475]){
#muestras2 <- c(muestras2, sub("(.*-.*-.*-.*)-.*-.*-.*", "\\1", j))
#}
#length(muestras2)
```

Ahora vamos a recuperar los índices en el vector de metilación `met2$sample` de los elementos del vector muestras2 que coincidan, y guardamos estos índices en el vector index4.

```
#index4 <- c()
#for (i in muestras2){
#index4 <- c(index4, which(met2$sample %in% i))
#}
#str(index4)
```

Vemos que el vector index tiene 131 elementos, sin embargo el vector muestras2 tenía 474 elementos, por lo que 343 muestras del dataset de Expresión Proteica no se encuentran en el dataset de metilación ¿Qué muestras de Expresión Proteica no se encuentran en el de metilación?

```
#index5 <- which(match(muestras2, met2$sample, nomatch = "0") %in% 0)
#length(index5)
```

Ahora tenemos que quitar de los datos de metilación los que no se encuentran en los datos de expresión proteica y viceversa. De nuestro vector muestras2 queremos quitar los elementos que se encuentran en los índices de index5.

```
#k <- sort(index5, decreasing = TRUE)
#datos2 <- datos
#for (k in k){
#datos2[k] <- NULL
#}
#ncol(datos2)
```

Ahora podemos conseguir las etiquetas de vital_status para los datos de expresión Proteica.

```
#ExpProtLabels2 <- c()
#for (i in index4){
#ExpProtLabels2 <- c(ExpProtLabels2, met2$vital_status[i])
#}
```

Ahora tenemos una n=131 en datos de Expresión proteica. Ahora vamos a quitar las muestras de metilación que no estén en Expresión Proteica.

```
#index6 <- sort(which(match(met2$sample, muestras2, nomatch = "0") %in% 0), decreasing = TRUE)
#ncol(met2)
```

El vector `index6` tiene una longitud de 19 elementos porque `150 - 131 = 19`. Las muestras iniciales que teníamos de metilación restándoles las muestras que tenemos al final en común entre las dos ómicas que estamos comparando.

```
#met3 <- met2
#for (k in index6){
#met3 <- met3[, -k]
#}
#ncol(met3)
```

Vemos que ahora tanto el archivo de Expresión proteica datos como el de los datos de metilación tienen todos 131 muestras. Vamos ahora a igualar el archivo de Expresión génica con el de metilación.

```
#muestras3 <- c()
#for (j in colnames(dataNorm)){
#muestras3 <- c(muestras3, sub("(.*-.*-.*-.*)-.*-.*-.*", "\\1", j))
#}
#length(muestras3)
```

```
#index7 <- c()
#for (i in muestras3){
#index7 <- c(index7, which(met3$sample %in% i))
#}
#length(index7)
```

met3$sample tiene los elementos 82 y 83 repetidos, son las mismas muestras. Por lo que en realidad

```
#index8 <- which(match(muestras3, met3$sample, nomatch = "0") %in% 0)
#length(index8)
```

```
#k <- sort(index8, decreasing = TRUE)
#dataNorm2 <- dataNorm
#dataNormt2 <- dataNormt
#for (k in k){
#dataNorm2 <- dataNorm2[, -k]
#dataNormt2 <- dataNormt2[, -k]
#}
#ncol(dataNorm2)
#ncol(dataNormt2)
```

Ahora podemos conseguir las etiquetas de vital_status para los datos de expresión génica.

```
#ExpProtLabels3 <- c()
#for (i in index7){
#ExpProtLabels3 <- c(ExpProtLabels3, met3$vital_status[i])
#}
```