

Informe 7: Servidor UOC Ubuntu. Puesta en marcha.

Carmen Lebrero Cia

Contents

Introducción	2
Puesta en marcha del servidor	2
Acceso al servidor	2
Instalación de programas en el servidor	2
Instalación R	4
Instalación Anaconda	4
Preparación de directorios	4
Procesado de datos en R	5
R en terminal Ubuntu	5
Instalación y carga de paquetes de R	5
Descarga de datos	5
Datos de Expresión Génica	5
Datos de Expresión Proteica	5
Datos de metilación	6
Procesado de datos	6
Expresión Génica	6
Filtrado: Genes con sumatorio de conteos > 1	6
Normalización	6
Transformación log2	6
Filtrado no específico	6
Análisis de expresión diferencial	6
Expresión Proteica	7
Buscar duplicados	7
Quitar Missing Values	7
Obtener los nombres cortos de las muestras de Expresión Proteica	7
Metilación	7
Obtener los nombres cortos de las muestras de metilación	7

Buscar y quitar duplicados	8
Quitar Sondas de SNPs	8
Quitar sondas de cromosomas X e Y	8
Quitar sondas que tienen al menos un missing value	8
Análisis de expresión diferencial	8
Igualar muestras entre ómicas	8
Igualar Expresión proteica y expresión génica	9
Igualar Expresión proteica y metilación	10

Introducción

Dado que nuestro espacio de RAM era demasiado reducido como para obtener todos los objetos que son necesarios para nuestro proyecto en cuanto a los datos de metilación, la UOC nos ha proporcionado un servidor Ubuntu 20.04 con características adaptadas a nuestras necesidades.

Sin embargo, nos encontramos con que el servidor no tiene instalados los programas que hemos solicitado, por lo que tenemos que poner el servidor en marcha para seguir con nuestro trabajo. En este informe se muestra el código utilizado tanto en la consola del servidor como el código R para tener los datos listos para trabajar.

Puesta en marcha del servidor

Acceso al servidor

Este es un paso sencillo que realizaremos desde nuestro `cmd` de Windows. Teniendo instalado `ssh` y habiendo suministrado nuestra clave pública a servicios informáticos de la UOC.

```
ssh carmenlc@84.88.58.67
```

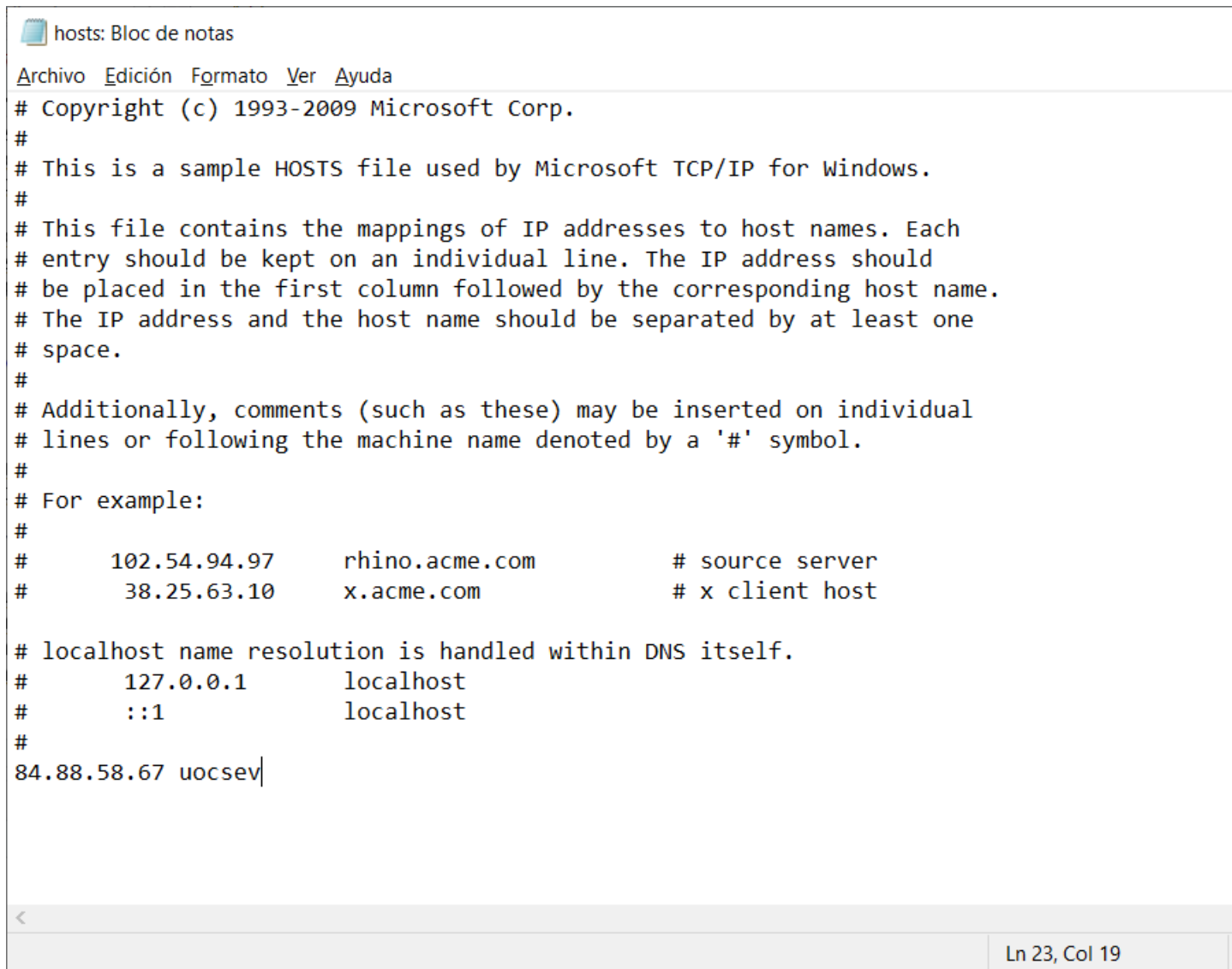
Para hacer más sencillas las próximas conexiones y no tener que poner la IP entera, podemos realizar unas modificaciones al archivo `hosts` encontrado en el path `C:\Windows\System32\drivers\etc`. Abrimos este archivo con un bloc de notas como administrador y simplemente tenemos que añadir al final la IP `84.88.58.67` junto con el nombre que le queramos dar al host, en nuestro caso `uocsev` (ver Imagen adjunta). Ahora podemos entrar en el servidor utilizando el siguiente comando:

```
ssh carmenlc@uocsev
```

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

Instalación de programas en el servidor

Como ya se ha adelantado, el servidor estaba completamente vacío por lo que se instalaron tanto R como Anaconda.



```
hosts: Bloc de notas
Archivo Edición Formato Ver Ayuda
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10       x.acme.com           # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1               localhost
#
84.88.58.67 uocsev|
```

<

Ln 23, Col 19

Figure 1: Captura de pantalla del archivo hosts modificado.

Instalación R

Para instalar R seguimos el siguiente procedimiento: <https://www.digitalocean.com/community/tutorials/how-to-install-r-on-ubuntu-20-04-es>

```
# Añadimos la clave GPG
```

```
(base) carmenlc@localhost:~$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E298A3A825C0
```

```
# Ahora añadimos el repositorio
```

```
(base) carmenlc@localhost:~$ sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu
```

```
# Ejecutamos update para incluir los manifiestos del paquete desde el nuevo repositorio.
```

```
(base) carmenlc@localhost:~$ sudo apt-get update
```

El siguiente paso sería la instalación de R, pero tuvimos unos problemas a la hora de instalarlo. En el repositorio de R no se encontraban dos librerías necesarias para la instalación base, por lo que se tuvo que editar el archivo `/etc/apt/sources.list`.

Se añadieron a la lista los siguientes repositorios:

```
deb http://cz.archive.ubuntu.com/ubuntu bionic main
```

```
deb https://mirrors.nic.cz/R/bin/linux/ubuntu trusty/
```

Una vez que se edita el archivo se hace un `update`

```
(base) carmenlc@localhost:~$ sudo nano /etc/apt/sources.list
```

```
(base) carmenlc@localhost:~$ sudo apt-get update
```

```
(base) carmenlc@localhost:~$ sudo apt install r-base
```

Instalación Anaconda

Seguimos el protocolo de: <https://www.digitalocean.com/community/tutorials/how-to-install-the-anaconda-python-distribution-on-ubuntu-20-04-es>

```
(base) carmenlc@localhost:~$ cd /tmp
```

```
(base) carmenlc@localhost:/tmp$ curl https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.
```

```
(base) carmenlc@localhost:/tmp$ bash anaconda.sh
```

Preparación de directorios

Crearemos un directorio llamado TFM en el que se guardará nuestro archivo `.RData` y los posibles productos creados en R.

```
(base) carmenlc@localhost:/tmp$ cd ..
```

```
(base) carmenlc@localhost:~$ mkdir TFM
```

```
(base) carmenlc@localhost:~$ cd TFM
```

```
(base) carmenlc@localhost:/TFM$ R
```

Procesado de datos en R

R en terminal Ubuntu

Desde el ordenador del alumno siempre hemos estado trabajando desde RStudio, que nos permite realizar las tareas de una manera más gráfica. Sin embargo, desde el servidor tendremos que trabajar con la consola de R y tenemos que habituarnos a cosas tan sencillas como el guardado del entorno de trabajo en archivos `.RData` mediante comandos.

El archivo en el que guardaremos todo el progreso que realizaremos en la sesión se llamará `TFM.RData`.

```
# Para guardar la sesión de R con todos los objetos
```

```
save.image(file = "TFM.RData")
```

```
# Para cargar una sesión de R
```

```
load("TFM.RData")
```

Instalación y carga de paquetes de R

Cargamos los paquetes necesarios:

```
install.packages("BiocManager")
requireNamespace("BiocManager", quietly=TRUE)
BiocManager::install("SummarizedExperiment")
BiocManager::install("TCGAbiolinks")
install.packages('DT')
library(TCGAbiolinks)
library(DT)
library(SummarizedExperiment)
install.packages('rvest')
```

Descarga de datos

Datos de Expresión Génica

```
QueryExpGenRSEM <- GDCquery(project = "TCGA-KIRC", data.category = "Gene expression", data.type = "Gene")
GDCdownload(QueryExpGenRSEM)
ExpGenTCGA_KIRC_RawData <- GDCprepare(QueryExpGenRSEM)
```

Datos de Expresión Proteica

```
Query_ExpProtTCGA_KIRC_RawData <- GDCquery(project = "TCGA-KIRC",
                                             data.category = "Protein expression",
                                             legacy = TRUE)
GDCdownload(Query_ExpProtTCGA_KIRC_RawData)
ExpProtTCGA_KIRC_RawData <- GDCprepare(Query_ExpProtTCGA_KIRC_RawData)
```

Datos de metilación

```
Query_MetTCGA_KIRC <- GDCquery(project = c("TCGA-KIRC"),
                                data.category = "DNA methylation",
                                platform = "Illumina Human Methylation 450",
                                legacy = TRUE)
GDCdownload(Query_MetTCGA_KIRC)
MetTCGA_KIRC_RawData <- GDCprepare(Query_MetTCGA_KIRC)
```

Procesado de datos

Expresión Génica

Filtrado: Genes con sumatorio de conteos > 1

```
keep <- rowSums(assay(ExpGenTCGA_KIRC_RawData)) > 1
ExpGenTCGA_KIRC_RawData <- ExpGenTCGA_KIRC_RawData[keep,]
```

Normalización

```
ExpGenTCGA_KIRC_Norm <- TCGAanalyze_Normalization(tabDF = ExpGenTCGA_KIRC_RawData, geneInfo = geneInfo)
```

Transformación log2

```
ExpGenTCGA_KIRC_Norm_Trans <- log2(ExpGenTCGA_KIRC_Norm+1)
```

Filtrado no específico

```
ExpGenTCGA_KIRC_Norm_Filt75 <- TCGAanalyze_Filtering(tabDF = ExpGenTCGA_KIRC_Norm, method = "quantile",
ExpGenTCGA_KIRC_Norm_Trans_Filt75 <- TCGAanalyze_Filtering(tabDF = ExpGenTCGA_KIRC_Norm_Trans, method =
```

Análisis de expresión diferencial

```
ExpGenTCGA_KIRC_SampleName_DeadStatus <- subset(ExpGenTCGA_KIRC_RawData$barcode, ExpGenTCGA_KIRC_RawData$barcode == "Dead")
ExpGenTCGA_KIRC_SampleName_AliveStatus <- subset(ExpGenTCGA_KIRC_RawData$barcode, ExpGenTCGA_KIRC_RawData$barcode == "Alive")
```

```
ExpGenTCGA_KIRC_Norm_Filt75_DEGs <- TCGAanalyze_DEA(mat1 = ExpGenTCGA_KIRC_Norm_Filt75[,ExpGenTCGA_KIRC_Norm_Filt75_DEGs])
```

```
ExpGenTCGA_KIRC_Norm_Trans_Filt75_DEGs <- TCGAanalyze_DEA(mat1 = ExpGenTCGA_KIRC_Norm_Trans_Filt75[,ExpGenTCGA_KIRC_Norm_Trans_Filt75_DEGs])
```

Vamos a sacar el nombre de los genes que han sido expresados diferencialmente:

```
DEGsNames <- rownames(ExpGenTCGA_KIRC_Norm_Filt75_DEGs)
```

```
# Sacar los índices de los genes diferencialmente expresados
```

```
IndexDEGs_in_ExpGenTCGA_KIRC_Norm_Filt75 <- c()
for (i in DEGsNames){
  IndexDEGs_in_ExpGenTCGA_KIRC_Norm_Filt75 <- c(IndexDEGs_in_ExpGenTCGA_KIRC_Norm_Filt75, which(rownames(ExpGenTCGA_KIRC_Norm_Filt75) == i))
}
```

```
ExpGenTCGA_KIRC_Norm_Filt75_238DEG <- ExpGenTCGA_KIRC_Norm_Filt75[IndexDEGs_in_ExpGenTCGA_KIRC_Norm_Fil
ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG <- ExpGenTCGA_KIRC_Norm_Trans_Filt75[IndexDEGs_in_ExpGenTCGA_K
ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_Transposed <- t(ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG)
ExpGenTCGA_KIRC_Norm_Trans_Filt75_Transposed <- t(ExpGenTCGA_KIRC_Norm_Trans_Filt75)
```

Expresión Proteica

Buscar duplicados

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

Quitar Missing Values

```
ExpProtTCGA_KIRC_RawData_woNA <- na.omit(ExpProtTCGA_KIRC_RawData)
```

Obtener los nombres cortos de las muestras de Expresión Proteica

```
ExpProtTCGA_KIRC_RawData_woNA_ColNames <- colnames(ExpProtTCGA_KIRC_RawData_woNA)[2:479]
```

```
ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort <- c()
for (j in ExpProtTCGA_KIRC_RawData_woNA_ColNames){
ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort <- c(ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort, sub("(.*-
})
```

Metilación

Vamos a crear los objetos:

- MetTCGA_KIRC_ColNames
- MetTCGA_KIRC_ColNames_Short
- MetTCGA_KIRC_RawData_woDupSamples
- MetTCGA_KIRC_RawData_woDupSamples_SNPProbes
- MetTCGA_KIRC_RawData_woDupSamples_woSNP
- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY
- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA
- Differentially_metylated_analysis

Obtener los nombres cortos de las muestras de metilación

```
MetTCGA_KIRC_ColNames <- colnames(MetTCGA_KIRC_RawData)
```

```
MetTCGA_KIRC_ColNames_Short <- c()
for (j in MetTCGA_KIRC_ColNames){
MetTCGA_KIRC_ColNames_Short <- c(MetTCGA_KIRC_ColNames_Short, sub("(.*-.*-.*-.*)-.*-.*-.*", "\\1", j))
}
```

Buscar y quitar duplicados

```
which(duplicated(MetTCGA_KIRC_Colnames_Short))
```

Están duplicadas las muestras de las columnas 308, 459, 474, 658 y 857.

```
MetTCGA_KIRC_RawData_woDupSamples <- MetTCGA_KIRC_RawData[,c(308, 459, 474, 658, 857)]
```

Nos quedamos con 483 muestras.

Quitar Sondas de SNPs

```
# Buscar las sondas de SNP
```

```
MetTCGA_KIRC_RawData_woDupSamples_SNPProbes <- grep("rs", rownames(MetTCGA_KIRC_RawData_woDupSamples))
```

```
# Quitar sondas de SNP
```

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP <- MetTCGA_KIRC_RawData_woDupSamples[-MetTCGA_KIRC_RawData_woDupSamples_SNPProbes]
```

Quitar sondas de cromosomas X e Y

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY <- subset(MetTCGA_KIRC_RawData_woDupSamples_woSNP, subset(MetTCGA_KIRC_RawData_woDupSamples_woSNP[,1:2] %in% c("X", "Y")) == FALSE)
```

Quitar sondas que tienen al menos un missing value

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA <- subset(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY, !is.na(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY[,1:2]))
```

Análisis de expresión diferencial

```
Differentially_metylated_analysis <- TCGAanalyze_DMC(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA,
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,
plot.filename = "survival_metvolcano.png",
diffmean.cut = 0.15,
save = FALSE,
legend = "State",
cores = 1 # if set to 1 there will be a progress bar
)
```

```
SondasDifencialmenteMet <- rownames(Differentially_metylated_analysis)[grep("hypo|hyper", Differentially_metylated_analysis$State)]
```

Igualar muestras entre ómicas

Tenemos que igualar las muestras entre las tres ómicas dado que:

- Expresión Génica: n=606
- Expresión proteica: n=478
- Metilación: n=483

Vamos a igualar todas las ómicas con las de expresión proteica, que es la que menos muestras tiene.

Igualar Expresión proteica y expresión génica

Obtenemos los índices en el vector de muestras de Expresión génica para todas las muestras de expresi

```
Index_ExpProt_ColNamesShort_IN_ExpGen <- c()
for (i in ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort){
  Index_ExpProt_ColNamesShort_IN_ExpGen <- c(Index_ExpProt_ColNamesShort_IN_ExpGen, which(ExpGenTCGA_KIR
}
```

Vemos que el vector `Index_ExpProt_ColNamesShort_IN_ExpGen` tiene 474 elementos, sin embargo el vector `ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort` 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 `ExpProtTCGA_KIRC_RawData_woNA` no se encuentran en `ExpGenTCGA_KIRC_RawData`?

Muestras de Expresión Proteica (índice numérico) que no están en Expresión Génica

```
ExpProtSampleNumb_notIN_ExpGen <- which(match(ExpProtTCGA_KIRC_RawData_woNA_ColNamesShort, ExpGenTCGA_KIR
> ExpProtSampleNumb_notIN_ExpGen
[1] 74 237 330 462
> ExpGenTCGA_KIRC_RawData$sample[c(74,237,330,462)]
[1] "TCGA-CZ-4859-01A" "TCGA-B8-5162-01A" "TCGA-B0-4841-01A" "TCGA-CZ-5986-11A"
```

```
> protein <- ExpProtTCGA_KIRC_RawData_woNA[[1]]
> length(protein)
[1] 177
> rownames(ExpProtTCGA_KIRC_RawData_woNA) <- protein
> ExpProtTCGA_KIRC_RawData_woNA[1] <- NULL
> ExpProtTCGA_KIRC_RawData_woNA[1]
```

	TCGA-A3-3316-01A-03-1737-20
14-3-3_beta-R-V	-0.047050047
14-3-3_epsilon-M-C	-0.138946618
14-3-3_zeta-R-V	-0.109604317
4E-BP1_pS65-R-V	-0.059234358
4E-BP1_pT37_T46-R-V	1.267113607
4E-BP1-R-V	0.062103695
A-Raf_pS299-R-C	-0.023886255
A-Raf-R-V	0.212508268
ACC_pS79-R-V	-0.392884046
ACVRL1-R-C	-0.523735450
ADAR1-M-V	0.257826948
Akt_pS473-R-V	1.174458627
Akt_pT308-R-V	0.514310549
Akt-R-V	0.147430330
AMPK_alpha-R-C	-0.375135494
AMPK_pT172-R-V	-0.310877697
Annexin_VII-M-V	-0.385922344
Annexin-1-M-E	0.120966272
AR-R-V	0.204110056
ASNS-R-V	0.352027812
B-Raf_pS445-R-V	0.166017593
Bad_pS112-R-V	-0.202430476
Bak-R-C	-0.082545172
Bax-R-V	-0.149074989

Quitamos las muestras de Expresión Proteica que no coinciden con muestras de Expresión Génica de los

```
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen <- ExpProtTCGA_KIRC_RawData_woNA[, -ExpProtSampleNumb_notIN]

> dim(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen)
[1] 177 474
```

Ahora en expresión proteica tenemos 474 muestras, pero todas esas muestras están en expresión génica. Ahora tenemos que hacer que todas las muestras de Expresión génica estén en expresión proteica.

- ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt
- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt

```
ExpGenSampleNumb_notIN_ExpProt <- which(match(ExpGenTCGA_KIRC_RawData$sample, ExpProtTCGA_KIRC_RawData$sample))

ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt <- ExpGenTCGA_KIRC_Norm_Trans_Filt75[, -ExpGenSampleNumb_notIN_ExpProt]
> dim(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt)
[1] 4897 474

ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG[, -ExpGenSampleNumb_notIN_ExpProt]
> dim(ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt)
[1] 238 474
```

Igualar Expresión proteica y metilación

```
# Nombre de las columnas del objeto MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA (Muestras de metilación)
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames <- colnames(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA)

# Obtenemos los nombres cortos de las muestras de MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames_Short <- c()
for (j in MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames){
  MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames_Short <- c(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames_Short, j)
}

# Obtenemos todos los nombres de columnas de las muestras de expresión proteica que ya han sido igualadas
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_Colnames <- colnames(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen)

# Obtenemos los nombres cortos de las muestras de ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_Colnames
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort <- c()
for (j in ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_Colnames){
  ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort <- c(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort, j)
}

%in% <- Negate(`in`)

# Muestras de metilación que no están en Expresión proteica
MetSampleNumb_notIN_ExpProt <- which(match(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames_Short, ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort))
```

```

> MetSampleNumb_notIN_ExpProt
[1] 5 7 10 14 16 19 20 21 22 24 29 34 38 40 41 47 48 49 54 57 58 59 65 67
[25] 69 74 80 81 85 86 87 90 91 94 96 97 98 100 101 103 104 110 111 112 113 115 117 121
[49] 122 123 124 126 127 128 131 132 133 135 138 139 140 145 146 147 149 155 157 159 164 166 170 171
[73] 172 173 178 181 185 187 189 192 198 201 206 208 211 213 214 221 224 227 228 231 233 235 237 238
[97] 243 245 250 253 254 257 259 262 265 267 269 271 272 277 279 281 282 285 286 288 289 291 292 298
[121] 299 301 302 303 304 307 308 309 312 313 315 321 323 324 331 332 333 336 337 340 344 347 348 350
[145] 351 353 360 361 362 364 370 371 373 375 379 381 385 390 391 393 394 398 401 404 405 408 410 411
[169] 412 414 420 421 430 433 434 435 437 438 440 441 445 448 450 451 452 461 467 469 470 471 478 479
[193] 480

> length(MetSampleNumb_notIN_ExpProt)
[1] 193

> length(MetTCGA_KIRC_ColNames_Short)
[1] 485

# Muestras de metilación que están en Expresión proteica

> MetSampleNumb_IN_ExpProt <- which(match(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_ColNames_Short,
> MetSampleNumb_IN_ExpProt
[1] 1 2 3 4 6 8 9 11 12 13 15 17 18 23 25 26 27 28 30 31 32 33 35 36
[25] 37 39 42 43 44 45 46 50 51 52 53 55 56 60 61 62 63 64 66 68 70 71 72 73
[49] 75 76 77 78 79 82 83 84 88 89 92 93 95 99 102 105 106 107 108 109 114 116 118 119
[73] 120 125 129 130 134 136 137 141 142 143 144 148 150 151 152 153 154 156 158 160 161 162 163 165
[97] 167 168 169 174 175 176 177 179 180 182 183 184 186 188 190 191 193 194 195 196 197 199 200 202
[121] 203 204 205 207 209 210 212 215 216 217 218 219 220 222 223 225 226 229 230 232 234 236 239 240
[145] 241 242 244 246 247 248 249 251 252 255 256 258 260 261 263 264 266 268 270 273 274 275 276 278
[169] 280 283 284 287 290 293 294 295 296 297 300 305 306 310 311 314 316 317 318 319 320 322 325 326
[193] 327 328 329 330 334 335 338 339 341 342 343 345 346 349 352 354 355 356 357 358 359 363 365 366
[217] 367 368 369 372 374 376 377 378 380 382 383 384 386 387 388 389 392 395 396 397 399 400 402 403
[241] 406 407 409 413 415 416 417 418 419 422 423 424 425 426 427 428 429 431 432 436 439 442 443 444
[265] 446 447 449 453 454 455 456 457 458 459 460 462 463 464 465 466 468 472 473 474 475 476 477 481
[289] 482 483

# Muestras de Expresión proteica que no están en metilación

ExpProtSampleNumb_notIN_Met <- which(match(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort,

> ExpProtSampleNumb_notIN_Met
[1] 1 2 3 5 7 9 11 12 14 18 19 23 31 33 35 37 39 42 48 49 51 52 54 55
[25] 60 63 65 66 70 71 72 73 74 77 80 81 83 86 87 94 96 102 104 116 119 120 122 124
[49] 125 127 132 133 136 138 139 142 143 145 146 148 150 157 162 163 167 169 172 173 180 187 189 192
[73] 193 195 196 201 202 207 210 211 212 214 216 218 221 223 224 225 226 228 232 234 235 236 239 246
[97] 247 249 253 254 255 259 263 267 270 271 275 277 282 289 292 294 295 297 298 299 300 309 315 317
[121] 319 320 323 324 325 326 328 329 333 334 336 337 338 340 341 345 346 348 358 359 360 366 370 371
[145] 376 377 381 391 394 395 396 401 403 407 408 413 414 418 420 422 423 424 430 432 433 435 437 439
[169] 440 441 442 444 446 447 449 455 456 459 460 462 464 465 466 471

> length(ExpProtSampleNumb_notIN_Met)
[1] 184

# Muestras de Expresión proteica que están en metilación

```

```

ExpProtSampleNumb_IN_Met <- which(match(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort, MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColNamesShort))
> ExpProtSampleNumb_IN_Met
[1] 4 6 8 10 13 15 16 17 20 21 22 24 25 26 27 28 29 30 32 34 36 38 40 41
[25] 43 44 45 46 47 50 53 56 57 58 59 61 62 64 67 68 69 75 76 78 79 82 84 85
[49] 88 89 90 91 92 93 95 97 98 99 100 101 103 105 106 107 108 109 110 111 112 113 114 115
[73] 117 118 121 123 126 128 129 130 131 134 135 137 140 141 144 147 149 151 152 153 154 155 156 158
[97] 159 160 161 164 165 166 168 170 171 174 175 176 177 178 179 181 182 183 184 185 186 188 190 191
[121] 194 197 198 199 200 203 204 205 206 208 209 213 215 217 219 220 222 227 229 230 231 233 237 238
[145] 240 241 242 243 244 245 248 250 251 252 256 257 258 260 261 262 264 265 266 268 269 272 273 274
[169] 276 278 279 280 281 283 284 285 286 287 288 290 291 293 296 301 302 303 304 305 306 307 308 310
[193] 311 312 313 314 316 318 321 322 327 330 331 332 335 339 342 343 344 347 349 350 351 352 353 354
[217] 355 356 357 361 362 363 364 365 367 368 369 372 373 374 375 378 379 380 382 383 384 385 386 387
[241] 388 389 390 392 393 397 398 399 400 402 404 405 406 409 410 411 412 415 416 417 419 421 425 426
[265] 427 428 429 431 434 436 438 443 445 448 450 451 452 453 454 457 458 461 463 467 468 469 470 472
[289] 473 474

```

Ambos dataset comparten 290 muestras. Vamos a obtener estas muestras de cada uno:

```

MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt <- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt
> dim(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt)
[1] 373382 290

```

```

ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet <- ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet
> dim(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet)
[1] 177 290

```

Ahora tenemos que igualar los datos de expresión génica para que haya los 290 elementos que también hay en metilación y en expresión proteica.

```

# Obtenemos todos los nombres de columnas de las muestras de expresión génica que ya han sido igualadas

```

```

ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_Colnames <- colnames(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt)

```

```

# Obtenemos los nombres cortos de las muestras de ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_Colnames

```

```

ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_ColNamesShort <- c()
for (j in ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_Colnames){
  ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_ColNamesShort <- c(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_ColNamesShort, j)
}

```

```

# Muestras de Expresión génica que están en metilación

```

```

ExpGenSampleNumb_IN_Met <- which(match(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_ColNamesShort, MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColNamesShort))

```

```

ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet
dim(ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet)
[1] 238 290

```

```

ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet
> dim(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet )
[1] 4897 290

```

Ya tenemos todas las ómicas igualadas con 290 muestras.

Además, vamos a renombrar todas las muestras de los datos siguientes:

- ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet
- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet
- ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet
- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt

```
# Renombrando ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet con muestras cortas:
```

```
ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_Colnames <- colnames(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet)
```

```
ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_ColnamesShort <- c()
for (j in ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_Colnames){
  ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_ColnamesShort <- c(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_ColnamesShort, j)
}
```

```
ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_Renamed <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet
colnames(ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_Renamed) <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_SameSampExpProt_SameSampMet_ColnamesShort
```

```
# Renombrando ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet
```

```
ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet_Renamed <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet
colnames(ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet_Renamed) <- ExpGenTCGA_KIRC_Norm_Trans_Filt75_238DEG_SameSampExpProt_SameSampMet_ColnamesShort
```

```
# Renombrando ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet
```

```
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_Colnames <- colnames(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet)
```

```
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_ColnamesShort <- c()
for (j in ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_Colnames){
  ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_ColnamesShort <- c(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_ColnamesShort, j)
}
```

```
ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_Renamed <- ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet
colnames(ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_Renamed) <- ExpProtTCGA_KIRC_RawData_woNA_SameSampExpGen_SameSampMet_ColnamesShort
```

```
# Renombrando MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt
```

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_Colnames <- colnames(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt)
```

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColnamesShort <- c()
for (j in MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_Colnames){
  MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColnamesShort <- c(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColnamesShort, j)
}
```

```
MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_Renamed <- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt
colnames(MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_Renamed) <- MetTCGA_KIRC_RawData_woDupSamples_woSNP_woXY_woNA_SameSampExpProt_ColnamesShort
```