This is for gene expression.

Downloading the data

For general guide refer to [[Working with TCGA biolinks]] documentation

First load the library:

```
# for downloading tcga data
library(TCGAbiolinks)
# for differential gene expression analysis
library(edgeR)
library(DESeq2)
```

Then we can use this structure to download the expression file. This is a expressionSet object of approx 500mb.

```
wd <- getwd()
library(TCGAbiolinks)
query <- GDCquery(
    project = "TCGA-BRCA",
    data.category = "Transcriptome Profiling",
    data.type = "Gene Expression Quantification",
    workflow.type = "STAR - Counts"
)
GDCdownload(
    query = query
)
dataPrep <- GDCprepare(
    query = query,
    save = TRUE
)
# save the file
saveRDS(dataPrep, "./data/TCGA-BRCA-eset.rds")
```

However, since it takes a lot of time to download the file I have already uploaded the file. Download it and load it:

```
dataPrep <- readRDS("./data/TCGA-BRCA-eset.rds")
```

Explaination of the data structure:

The results is a expresisonSet object with the following qualities:

- `colData()` (or in some cases `phenoData()`): contains information for each column (each patient)
- `rowData()` (or in some cases `featureData()`): contain information for each row (each gene)
- `assay()`: Contains the information obtained from the assay.

In this case, this is a expressionSet object that has RNASeq information curated through STAR program.

STAR provides multiple set of information as you can see here:

```
names(assays(eset))
```

But for DESEQ analysis, we prefer the raw read count which is the `unstraneded assay`. We will get to this shortly.

cleaning the data

To clean it u can use these steps:

As you can see, the

```
table(colData(eset)$sample_type)
```

First we can filter data for primary tumor.

```
eset <- eset[, colData(eset)$sample_type == "Primary Tumor"]
```

And choose protein coding genes:

```
# remove non-protein coding and duplicated gene
eset <- eset[rowData(eset)$gene_type == "protein_coding", ]
```

Then we can clean the patient names. with regex. This is to make the data compatible with the other datasets that use the shorter version without the sample metadata information associated with these codes.

> The regex removes the samples with pattern `-01A-, -01B-, ...` pattern. This is for primary tumors and other ones may need different patterns for cleaning

```
colnames(eset) <- gsub("-01[A-Z]-.*$", "", colnames(eset))
```

I usually just remove duplicated genes. very few exist with these filtering so the effect is highly minimal

```
eset <- eset[!duplicated(rowData(eset)$gene_name), ]
rownames(eset) <- rowData(eset)$gene_name
```

Raw expression data is on the `unstranded` part of the expressionSet as describe in this github issue

```
studyExpr <- assays(eset)[["unstranded"]]
```

patient clinical data is in `colData` of the expressionset object. Usually it has more complete information than cbioportal.

```
clinical <- colData(eset)
```

We then can remove genes with NA values or 0 variance

```
rvar <- apply(studyExpr, 1, var)
studyExpr <- studyExpr[complete.cases(rvar), ]
rvar <- apply(studyExpr, 1, var)
studyExpr <- studyExpr[rvar != 0, ]
```

Then remove patients with NA values or 0 variance.

```
cvar <- apply(studyExpr, 2, var)
studyExpr <- studyExpr[, complete.cases(cvar)]
cvar <- apply(studyExpr, 2, var)
studyExpr <- studyExpr[, cvar != 0]
```

> ***This step should not result in reduction of patient. If it does please investigate further***

Gene expression normalization

Gene expression normalization is usually done for machine learning and visualization applications in bioinformatics. For differential gene expression, we use different methods as described in the other file in this repo.

DESEQ2 normalization

One of the most popular methods is DESeq2 normalization which is quite sophisticated. However, we can fully utilize the normalization technique with simple commands in R

```
dds <- DESeqDataSetFromMatrix(countData = studyExpr, colData =
clinical, design = ~1)
dds <- estimateSizeFactors(dds)
studyExpr <- counts(dds, normalized = TRUE)
```

Other methods

Alternatively, we can do z score normalization on expression data:

```
studyExpr <- apply(studyExpr, 2, function(x) {
    return((x - mean(x)) / sd(x))
})
```

Or scale them using the scale function in R:

```
studyExpr <- scale(studyExpr)
```

MISC:

To download all rnaseq data of the GDC and TCGA dataset, use this script.

Please filter the projects as you see fit, there are many projects and the expression data can get upwards of 100GB if you are not careful.

```
# projects <- getGDCprojects()$id
# # to filter to only include TCGA:
# # projects <- grep("TCGA-", projects, value = TRUE)
#
# wd <- getwd()
# for (project in projects) {
#     tryCatch(
#         {
#             f <- sprintf("./data/%s-rnaseq.rds", project)
#             if (file.exists(f)) next()
#             query <- GDCquery(
#                 project = project,
#               data.category = "Transcriptome Profiling",
#               data.type = "Gene Expression Quantification",
#               workflow.type = "STAR - Counts"
#             )
#             # setwd("~/.cache/TCGAbiolinks/")
#             GDCdownload(
#                 query = query
#             )
#             dataPrep <- GDCprepare(
#                 query = query,
#                 save = TRUE
#             )
#             # setwd(wd)
#             saveRDS(dataPrep, f)
#         },
#         error = function(cond) {
#             message(cond)
#         }
#     )
# }
# print("done")
```