scPOST objective

- scPOST is a simulation framework that estimates a study design's power to detect differentially abundant cell states (e.g. an expansion of a cell state in disease samples compared to healthy).
- scPOST allows users to control the simulated data's characteristics, including: effect size, the number of samples, the number of cells per sample, the batch multiplexing structure, and the magnitude of simulated noise.
- Thus, users can use scPOST to explore how different study design choices might affect power.

Installation

In [55]:

```
library(devtools)
devtools::install_github(repo = "immunogenomics/scpost", force = TRUE)
```

```
Downloading GitHub repo immunogenomics/scpost@HEAD

✔  checking for file ‘/tmp/RtmpI8Uutj/remotes82ee43ec634a/immunogenomic
s-scpost-c574a17/DESCRIPTION’
─  preparing ‘scpost’:
✔  checking DESCRIPTION meta-information
─  checking for LF line-endings in source and make files and shell scri
pts
─  checking for empty or unneeded directories
─  looking to see if a ‘data/datalist’ file should be added
─  building ‘scpost_1.0.tar.gz’ (340ms)
   Warning: invalid uid value replaced by that for user 'nobody'
```

In [56]:

```
library(scpost)
```

Minimum input

To simulate data, scPOST uses an input prototype dataset (such as public or pilot data). scPOST estimates two types of variation often found in multi-sample single-cell data:

1. Variation in a cell state's frequency across samples (Cell state frequency variation)
2. Variation in a cell state's gene expression. We estimate and simulate gene expression with principal components (PCs), because PCs are a summary of gene expression that also takes into account gene covariation. This also reduces computational burden.

scPOST requires the following inputs for each cell:

1. Cell state annotations (in single-cell these are often obtained from clustering algorithms, such as the Louvain method)
2. Sample annotations (the sample each cell comes from)
3. Batch annotations (if no batch information for the data is available, it is sufficient to treat every sample as it's own batch)
4. Principal component values (these are obtained from PCA)

Let's take a look at an example of a prototype dataset that we will apply scPOST to.

In [57]:

```
ra_FibObj$meta %>% str
ra_FibObj$embeddings %>% head(2)
```

```
'data.frame':   1844 obs. of  11 variables:
 $ UMAP1       : num  -4.81 -4.13 -0.23 -4.86 1.44 ...
 $ UMAP2       : num  0.961 0.804 -2.206 0.906 1.474 ...
 $ CellID      : chr  "S006_L1Q1_M01" "S006_L1Q1_M03" "S006_L1Q1_M05"
"S006_L1Q1_M07" ...
 $ sample      : Factor w/ 21 levels "300-0122","300-0153",..: 18 18 18
18 18 18 18 18 18 18 ...
 $ batch       : Factor w/ 24 levels "300-0122","300-0153",..: 18 18 18
18 18 18 18 18 18 18 ...
 $ disease     : chr  "OA" "OA" "OA" "OA" ...
 $ nUMI        : int  14660 12703 10968 16464 13746 14744 15939 7979 19
584 11007 ...
 $ nGenes      : int  4152 3923 3760 4552 3734 4151 4204 2888 4714 3681
...
 $ celltype    : Factor w/ 5 levels "B cell","Empty",..: 3 3 3 3 3 3 3
3 3 3 ...
 $ clusAllCells: Factor w/ 12 levels "0","1","2","3",..: 6 6 7 6 4 6 6
9 6 6 ...
 $ clusOnlyFib : Factor w/ 9 levels "0","1","2","3",..: 3 3 2 3 1 3 3 1
3 3 ...
```

A data.frame: 2 × 20

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<(
S006_L1Q1_M01	19.54967	-1.811336	1.338648	-3.845058	-1.56015	-1.102576	1.6973464	-0.694
S006_L1Q1_M03	16.15553	-1.055468	2.950489	1.226705	-1.64750	1.241454	0.6577186	3.239

In the metadata, we have:

- Cell state annotations (clusOnlyFib)
- Sample annotations (sample)
- Batch annotations (batch)

In the embeddings, we have:

- Principal component values for 20 PCs

Getting started with scPOST

This tutorial will run through the scPOST framework, which comprises 3 steps.

Step 1: Parameter estimation

Step 2: Dataset simulation

Step 3: Association testing for cell state frequency shifts (e.g. expansion or deletion)

Here, we will simulate fibroblast data from the rheumatoid arthritis (RA) dataset described in Zhang F, Wei K, Slowikowski K, Fonseka C, Rao DA, *et al.*, *Nature Immunol* (2020). The input prototype dataset we will use is contained in "ra_FibObj".

Through cytometry, the authors identified an HLA-DRA+ fibroblast cell state that was expanded in inflammatory RA samples compared to osteoarthritis samples. However, they we unable to detect this expansion in their single-cell RNA sequencing (scRNA-seq) data due to sample size (n = 21, 17 case, 4 control).

Let's use scPOST to find study design changes that might increase power.

Step 1: Parameter estimation

Cell state frequency Variation

With the "estimateFreqVar" function, we estimate each cell state's mean frequency and covariance across samples

In [58]:

```
system.time({
    raFib_freqEstimates <- estimateFreqVar(meta = ra_FibObj$meta, clusCol = 'clusOn
lyFib', sampleCol = 'sample', logCov = TRUE)
})
```

```
   user  system elapsed 
  0.178   0.000   0.178 
```

In [59]:

```
raFib_freqEstimates %>% str
```

```
List of 2
 $ meanFreq: Named num [1:9] 0.1871 0.1783 0.1481 0.1038 0.0966 ...
  ..- attr(*, "names")= chr [1:9] "clus0" "clus1" "clus2" "clus3" ...
 $ cfcov   : num [1:9, 1:9] 2.87 -1.225 0.278 0.311 -0.705 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:9] "clus0" "clus1" "clus2" "clus3" ...
  .. ..$ : chr [1:9] "clus0" "clus1" "clus2" "clus3" ...
```

Gene expression variation

With the "estimatePCVar" function, we estimate each cell state's gene expression variation. We use linear mixed effects models to estimate this variance, which we can deconvolute into different sources of variance. From these models, we estimate the following for each cell state:

1. Centroid in PC space
2. Residual variance in PC space
3. Batch-associated variance in PC space
4. Sample-associated variance in PC space

Computation time will depend on the size of the real data. For larger datasets, we recommend saving the aforementioned estimates so that they can be used for future simulations

In [60]:

```
system.time({
    raFib_pcEstimates <- estimatePCVar(pca = ra_FibObj$embeddings, npcs = 20, meta
= ra_FibObj$meta, clusCol = 'clusOnlyFib',
                                       sampleCol = 'sample', batchCol = 'batch')
})
```

```
   user  system elapsed
  6.749   1.186   6.606
```

In [61]:

```
raFib_pcEstimates %>% str(1)
# Save raFib_pcaEstimates for future use
```

```
List of 4
 $ centroids  : num [1:9, 1:20] -1.32 -1.58 14.28 -6.91 -7.64 ...
  ..- attr(*, "dimnames")=List of 2
 $ pc_cov_list:List of 9
 $ batch_vars : num [1:9, 1:20] 0.848 5.89 4.168 5.455 1.944 ...
  ..- attr(*, "dimnames")=List of 2
 $ sample_vars: num [1:9, 1:20] 1.902757 8.78203 0.104539 0.000455 2.81
6822 ...
  ..- attr(*, "dimnames")=List of 2
```

Step 2 and 3: Simulate dataset and perform association testing

Next, we'll generate *in silico* fibroblast datasets and use MASC (Mixed effects association of single-cells) to test whether we test an expansion or depletion of a cluster in our simulated data.

For users that wish to estimate power to detect differential abundance, we recommend users use the "simDataset.withMASC" function to perform both **Step 2 and Step 3** at the same time so that they may make use of streamlined file naming and handling. Users that only wish to simulate a dataset can use the "simDataset.base" function.

Important: Before running simulations, create a save folder where you will save the results of the simulations

- Before running our simulations, we created a "scpostSims/gettingStarted" folder

Simulate datasets with 20 samples: unbalanced study design

In [62]:

```
createParamTable <- function(nreps, clus, fc, ncases = 10, nctrls = 10, nbatches =
4, b_scale = 1, s_scale = 1, cf_scale = 1,
                             res_use = 1.2, cond_induce = "cases", save_path){
    paramTable <- expand.grid(
        rep = seq(nreps),
        ncases = ncases,
        nctrls = nctrls,
        nbatches = nbatches,
        b_scale = b_scale,
        s_scale = s_scale,
        cf_scale = cf_scale,
        clus = clus,
        fc = fc,
        res_use = res_use,
        save_path = save_path
    )
    paramTable$cond_induce = cond_induce
    paramTable$seed <- sample(.Machine$integer.max, size = nrow(paramTable))

    return(paramTable)
}
```

The cohort of the original RA scRNA-seq data consisted of 17 cases and 4 controls. Here, we'll simulate 5 datasets that each have 20 samples (17 cases and 3 controls).

Let's set the number of samples, the number of cells per sample, and the batch structure.

In [63]:

```
set.seed(23)

# Set the number of samples, number of cells per sample, and create batch structure
ncases <- 17
nctrls <- 3
nbatches <- 4
batchStructure <- distribSamples(ncases = ncases, nctrls = nctrls, nbatches = nbatc
hes)
ncells <- rep(250, times = ncases + nctrls)
names(ncells) <- batchStructure$sample_names

batchStructure %>% str
```

```
List of 4
 $ batches     :List of 4
  ..$ batch1:List of 2
  .. ..$ cases: chr [1:5] "sample20" "sample13" "sample11" "sample1"
...
  .. ..$ ctrls: chr "sample5"
  ..$ batch2:List of 2
  .. ..$ cases: chr [1:4] "sample17" "sample2" "sample10" "sample19"
  .. ..$ ctrls: chr "sample9"
  ..$ batch3:List of 2
  .. ..$ cases: chr [1:4] "sample18" "sample12" "sample8" "sample3"
  .. ..$ ctrls: chr "sample14"
  ..$ batch4:List of 2
  .. ..$ cases: chr [1:4] "sample16" "sample7" "sample4" "sample6"
  .. ..$ ctrls: chr(0)
 $ sample_names: chr [1:20] "sample1" "sample2" "sample3" "sample4" ...
 $ cases       : chr [1:17] "sample20" "sample17" "sample18" "sample16"
...
 $ ctrls       : chr [1:3] "sample5" "sample9" "sample14"
```

Next, we'll set up a parameter table with the "createParamTable" function that we'll use to run multiple simulations:

- We'll simulate realistic levels of variation by setting "b_scale", "s_scale", and "cf_scale" equal to 1.
- We'll induce a fold-change of 5 into "clus0", the HLA-DRA+ fibroblast cell state
- We'll set up a folder where we will save our results

In [64]:

```
params <- createParamTable(
    nreps = 5,
    clus = "clus0",
    fc = 5,
    ncases = ncases,
    nctrls = nctrls,
    nbatches = nbatches,
    b_scale = 1,
    s_scale = 1,
    cf_scale = 1,
    res_use = 0.6,
    cond_induce = "cases",
    save_path = file.path(getwd(), "scpostSims/gettingStarted/")
)

params %>% head(2)
```

A data.frame: 2 × 13

rep	ncases	nctrls	nbatches	b_scale	s_scale	cf_scale	clus	fc	res_use	
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<dbl>	<dbl>	
1	17	3	4	1	1	1	clus0	5	0.6	/data/srlab1/nm
2	17	3	4	1	1	1	clus0	5	0.6	/data/srlab1/nm

The "simDataset.MASC" function will simulate datasets and save them as a list containing:

- Metadata information for each simulated cell
- p-values from applying MASC to test for differential abundance. We obtain a p-value for each simulated cell state.
- For larger simulations, we recommend increasing "mc.cores" to make use of multiple cores if possible

In [65]:

```
suppressWarnings({
    lapply(seq(nrow(params)), function(x){
            simDataset.withMASC(
                save_path = params[x, 'save_path'],
                rep = params[x, 'rep'],
                seed = params[x, 'seed'],
                ncases = params[x, 'ncases'],
                nctrls = params[x, 'nctrls'],
                nbatches = params[x, 'nbatches'],
                batchStructure = batchStructure,
                ncells = ncells,
                centroids = raFib_pcEstimates$centroids,
                pc_cov_list = raFib_pcEstimates$pc_cov_list,
                batch_vars = raFib_pcEstimates$batch_vars,
                b_scale = params[x, 'b_scale'],
                sample_vars = raFib_pcEstimates$sample_vars,
                s_scale = params[x, 's_scale'],
                cfcov = raFib_freqEstimates$cfcov,
                cf_scale = params[x, 'cf_scale'],
                meanFreqs = raFib_freqEstimates$meanFreq,
                clus = params[x, 'clus'],
                fc = params[x, 'fc'],
                cond_induce = params[x, 'cond_induce'],
                res_use = params[x, 'res_use'],
                mc.cores = 1,
                clusterData = TRUE,
                returnPCs = FALSE
            )
    })
})
```

```
Simulated dataset at 2020-11-24 12:07:44
Simulated dataset at 2020-11-24 12:07:50
Simulated dataset at 2020-11-24 12:07:58
Simulated dataset at 2020-11-24 12:08:04
Simulated dataset at 2020-11-24 12:08:10
```

1. NULL
2. NULL
3. NULL
4. NULL
5. NULL

Simulate datasets with 20 samples: balanced study design

Now, let's simulate 20-sample datasets with almost the same study design. However, now we'll simulate a more balanced number of cases and controls: 10 cases and 10 controls

In [66]:

```
set.seed(23)

# Set the number of samples, number of cells per sample, and create batch structure
ncases <- 10
nctrls <- 10
nbatches <- 4
batchStructure <- distribSamples(ncases = ncases, nctrls = nctrls, nbatches = nbatc
hes)
ncells <- rep(250, times = ncases + nctrls)
names(ncells) <- batchStructure$sample_names

params <- createParamTable(
    nreps = 5,
    clus = "clus0",
    fc = 5,
    ncases = ncases,
    nctrls = nctrls,
    nbatches = nbatches,
    b_scale = 1,
    s_scale = 1,
    cf_scale = 1,
    res_use = 0.6,
    cond_induce = "cases",
    save_path = file.path(getwd(), "scpostSims/gettingStarted/")
)

params %>% head(2)
```

A data.frame: 2 × 13

rep	ncases	nctrls	nbatches	b_scale	s_scale	cf_scale	clus	fc	res_use	
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<dbl>	<dbl>	
1	10	10	4	1	1	1	clus0	5	0.6	/data/srlab1/nm
2	10	10	4	1	1	1	clus0	5	0.6	/data/srlab1/nm

In [67]:

```
suppressWarnings({
    lapply(seq(nrow(params)), function(x){
            simDataset.withMASC(
                save_path = params[x, 'save_path'],
                rep = params[x, 'rep'],
                seed = params[x, 'seed'],
                ncases = params[x, 'ncases'],
                nctrls = params[x, 'nctrls'],
                nbatches = params[x, 'nbatches'],
                batchStructure = batchStructure,
                ncells = ncells,
                centroids = raFib_pcEstimates$centroids,
                pc_cov_list = raFib_pcEstimates$pc_cov_list,
                batch_vars = raFib_pcEstimates$batch_vars,
                b_scale = params[x, 'b_scale'],
                sample_vars = raFib_pcEstimates$sample_vars,
                s_scale = params[x, 's_scale'],
                cfcov = raFib_freqEstimates$cfcov,
                cf_scale = params[x, 'cf_scale'],
                meanFreqs = raFib_freqEstimates$meanFreq,
                clus = params[x, 'clus'],
                fc = params[x, 'fc'],
                cond_induce = params[x, 'cond_induce'],
                res_use = params[x, 'res_use'],
                mc.cores = 1,
                clusterData = TRUE,
                returnPCs = FALSE
            )
    })
})
```

```
Simulated dataset at 2020-11-24 12:08:16
Simulated dataset at 2020-11-24 12:08:23
Simulated dataset at 2020-11-24 12:08:29
Simulated dataset at 2020-11-24 12:08:37
Simulated dataset at 2020-11-24 12:08:44
```

1. NULL
2. NULL
3. NULL
4. NULL
5. NULL

Get Power

Now that we've simulated data and performed association testing, we can retrieve what our estimated power is.

We only simulated 5 replicates for each study design; to achieve a more accurate estimate, we recommend running many replicates. Here, we load the results tables from MASC analysis (named "res" in the saved list that was created by the "simDataset.MASC" function)

In [68]:

```
dir <- file.path(getwd(), "scpostSims/gettingStarted")
filenames <- list.files(path = dir,
                        full.names = T,
                        pattern = '*res') %>% basename
resTables <- lapply(filenames, function(x){
        readRDS(file.path(dir, x))[["res"]]
})
```

If you induced a fold change in multiple cell states, you can stratify the power results by cell state (instead of aggregate power over all cell states). Here, we only induced a fold change in the HLA-DRA+ fibroblast cell state, so that is not necessary.

In [69]:

```
getPowerFromRes(
    resFiles = filenames,
    resTables = resTables,
    threshold = 0.05,
    z = 1.96,
    stratByClus = FALSE
)
```

A data.frame: 2 × 11

ind_fc	ncases	nctrls	nsamples	ncells	bscale	sscale	cfscale	trials	masc_power	masc_pow
<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<int>	<dbl>	<
5	10	10	20	250	1	1	1	5	0.8	0.350
5	17	3	20	250	1	1	1	5	0.0	0.000

From these small number of simulations, we estimated that the unbalanced study design would have 0% power. However, by changing the balance of cases and controls so that the study design is more balanced, we see an estimated power of 80%. This example showcases how scPOST can be used to evaluate how changing a study design might affect power.

For more accurate estimates, we recommend running higher numbers of simulations. In Millard *et al.*, we ran 500 simulations for each of these study designs, and estimated 12% power for the unbalanced study design and 60% power for the balanced study design.

End