Data visualization: tSNE

05 November 2021

Practical Session tSNE

Please, complete all the code proposed and answer to questions.

You will find these on the slides as (#) and (Q). You are not expected to include your answer to the check questions (C) or (K) knowledge thoughts. These are only to guide your progress.

Change the code appropriately or add new lines if necessary.

Load data into variables:

```r
# First of all, set your working directory if necessary.
setwd("C:/Users/aless/Desktop/Bioinformatics/Erasmus BCN/Data Visualisation/p2_tSNE")

# Read data & metadata
P1_data = read.delim('P1.tsv.gz', row.names = 1)
P2_data = read.delim('P2.tsv.gz', row.names = 1)

treat = as.factor(readLines('treatment.txt'))

# Check the data and that we read in the data correctly:
print(head(P1_data)[,1:5])
```

```
##                    P2449_N716.S502 P2449_N718.S502 P2449_N719.S502
## ENSG00000000003.14               0               0            0.00
## ENSG00000000005.5                0               0            0.00
## ENSG00000000419.12              43              53           91.00
## ENSG00000000457.13               1               0           20.33
## ENSG00000000460.16               2              89           26.67
## ENSG00000000938.12              15               0           16.00
##                    P2449_N720.S502 P2449_N721.S502
## ENSG00000000003.14               0               0
## ENSG00000000005.5                0               0
## ENSG00000000419.12              80              61
## ENSG00000000457.13               0               0
## ENSG00000000460.16               0             138
## ENSG00000000938.12               0               0
```

```r
print(head(P2_data)[,1:5])
```

```
##                    P2458_N701.S513 P2458_N702.S513 P2458_N703.S513
## ENSG00000000003.14               0            0.00            0.00
## ENSG00000000005.5                0            0.00            0.00
## ENSG00000000419.12               5           13.00           18.00
## ENSG00000000457.13              0          0.01          9.15
## ENSG00000000460.16             20         45.00         16.85
## ENSG00000000938.12              0          0.00          0.00
##                    P2458_N704.S513 P2458_N705.S513
## ENSG00000000003.14              0          0.00
## ENSG00000000005.5               0          0.00
## ENSG00000000419.12             26         10.00
## ENSG00000000457.13              0         33.99
## ENSG00000000460.16             82         10.01
## ENSG00000000938.12              0          0.00
```

```r
print(treat)
```

```
##  [1] N N N N N N U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U T T
## [39] T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T
## [77] T T T T T T T T T T T T T T T T T T T T
## Levels: N T U
```

```r
print(dim(P1_data))
```

```
## [1] 58441    96
```

```r
print(dim(P2_data))
```

```
## [1] 58441    96
```

```r
print(table(treat))
```

```
## treat
##  N  T  U
##  6 60 30
```

Adapt the data accordingly and merge both tables of observations. Since we need columns of the tables to represent variables and rows to be samples, individuals, we need to transpose the data of the matrices. Then, we merged experiments into a single data.frame.

```r
# Modify the matrices.
X1 = t(P1_data)
X2 = t(P2_data)

# Merge replicates
X = rbind(X1, X2)
```

Check that we transformed the data correctly:

```r
print(dim(X1))
```

```
## [1]    96 58441
```

```r
print(dim(X2))
```

```
## [1]    96 58441
```

```r
print(dim(X))
```

```
## [1]   192 58441
```

Get metadata associated:

```r
meta_data <- data.frame(row.names = rownames(X),
                        experiment= as.factor(rep(c("A","B"), each=96)),
                        treatment= treat)

head(meta_data)
```

```
##                  experiment treatment
## P2449_N716.S502          A         N
## P2449_N718.S502          A         N
## P2449_N719.S502          A         N
## P2449_N720.S502          A         N
## P2449_N721.S502          A         N
## P2449_N722.S502          A         N
```

Load package `Rtsne` and check the help. Run `install.packages("Rtsne")` if required.

```r
library(Rtsne)
help(Rtsne)
```

```
## starting httpd help server ... done
```

Q1. What are default values for perplexity, initial dimensions and iterations?

A1. The default values for perplexity, initial dimensions and iterations are 30, 50, and 1000 respectively.

A simple example of tSNE plot.

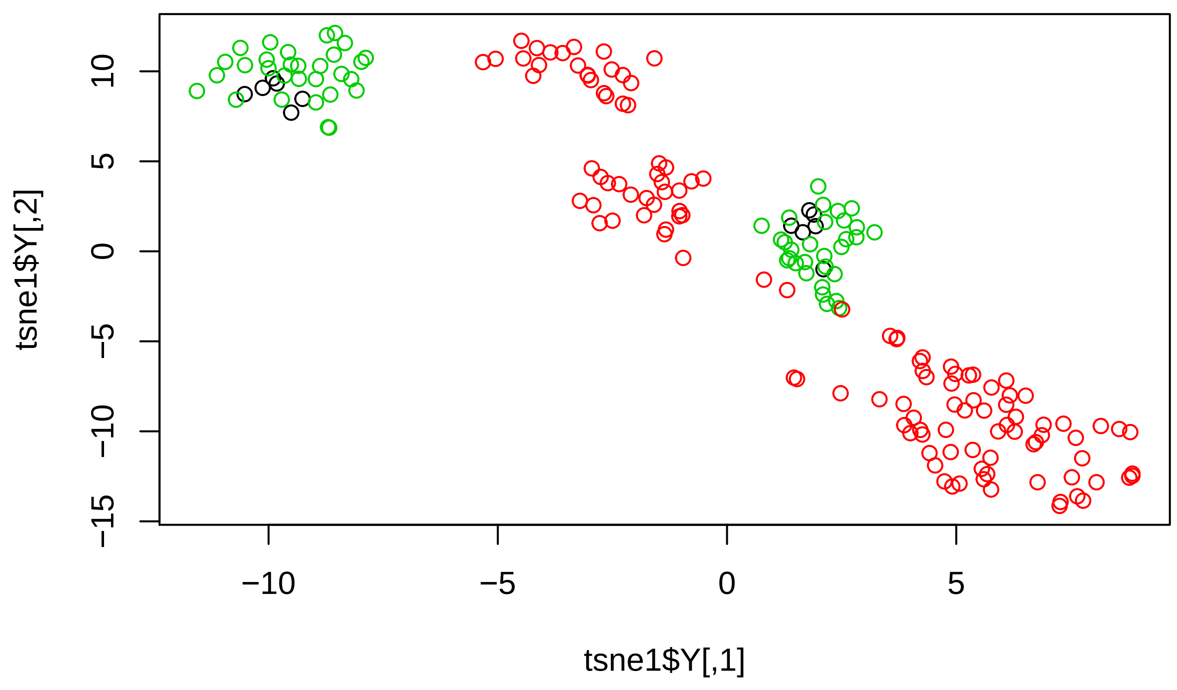
```r
## Default parametnes for Rtsne
tsne1 <- Rtsne(X)

## Check results generated
names(tsne1)
```

```
##  [1] "N"                   "Y"                   "costs"
##  [4] "itercosts"           "origD"               "perplexity"
##  [7] "theta"               "max_iter"            "stop_lying_iter"
## [10] "mom_switch_iter"     "momentum"            "final_momentum"
## [13] "eta"                 "exaggeration_factor"
```

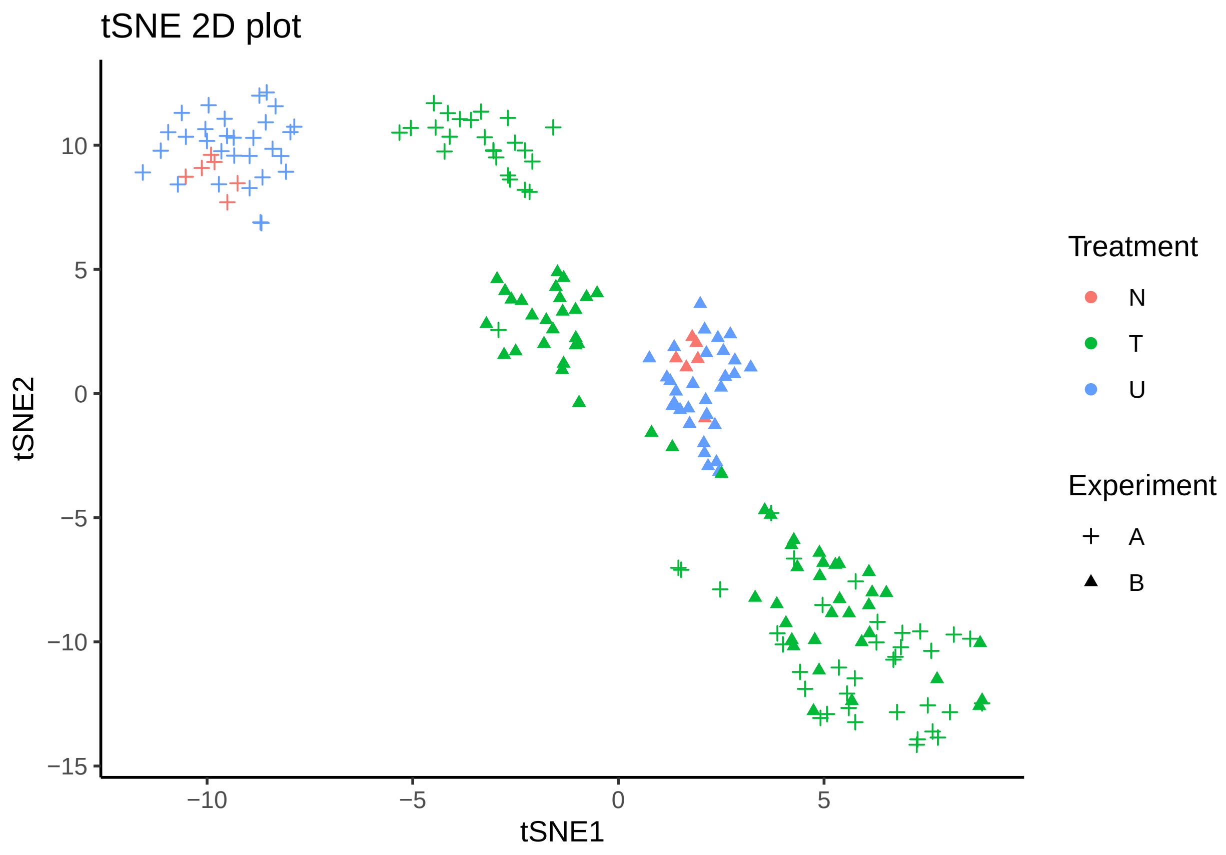
```r
head(tsne1$Y)
```

```
##             [,1]     [,2]
## [1,] -10.521189 8.732627
## [2,]  -9.902713 9.611694
## [3,]  -9.506814 7.703964
## [4,]  -9.259948 8.469074
## [5,]  -9.820462 9.323392
## [6,] -10.128411 9.081040
```

```r
## Basic R plot
plot(tsne1$Y, col=treat)

tsne_data <- as.data.frame(tsne1$Y)
tsne_data$Experiment <- meta_data$experiment
tsne_data$Treatment <- meta_data$treatment

## ggplot
library(ggplot2)
```



```r
ggplot(tsne_data, aes(x=V1, y=V2)) +
  geom_point(aes(color=Treatment, shape=Experiment)) +
  scale_shape_manual(values = c(3,17)) +
  ggtitle("tSNE 2D plot") + xlab("tSNE1") + ylab("tSNE2") +
  theme_classic()
```



Explore reproducibility of results

Run a second tSNE with the same dataset and parameters:

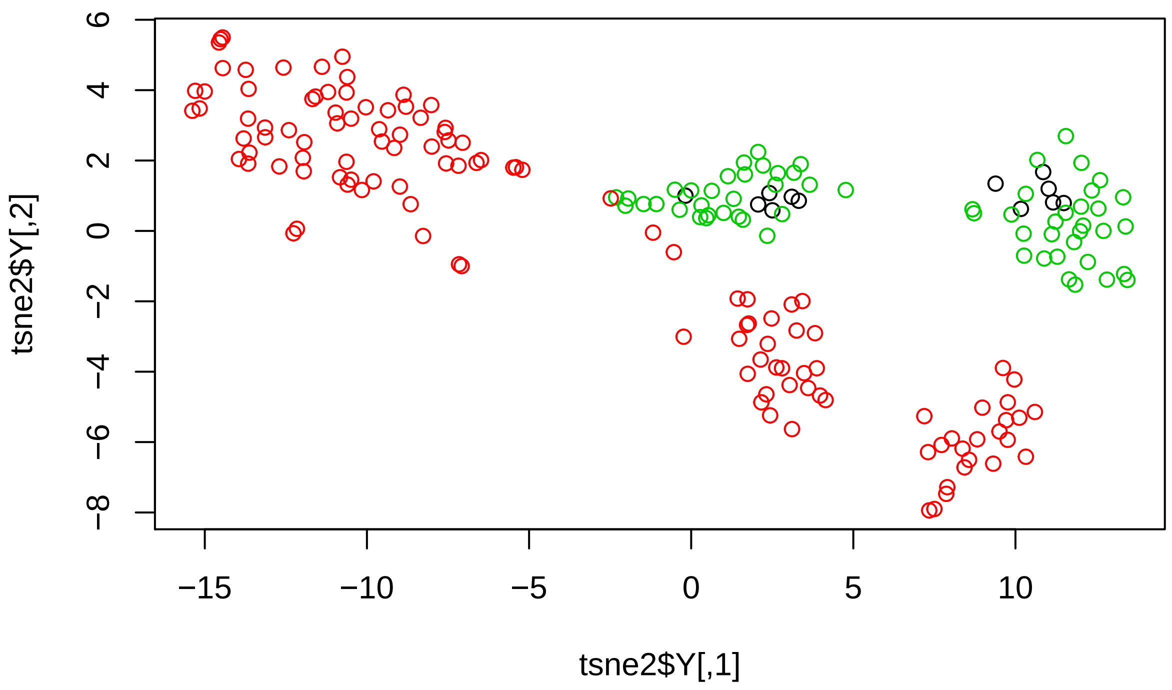
```
tsne2 <- Rtsne(X)

names(tsne2)
```

```
##  [1] "N"                    "Y"                    "costs"
##  [4] "itercosts"            "origD"                "perplexity"
##  [7] "theta"                "max_iter"             "stop_lying_iter"
## [10] "mom_switch_iter"      "momentum"             "final_momentum"
## [13] "eta"                  "exaggeration_factor"
```

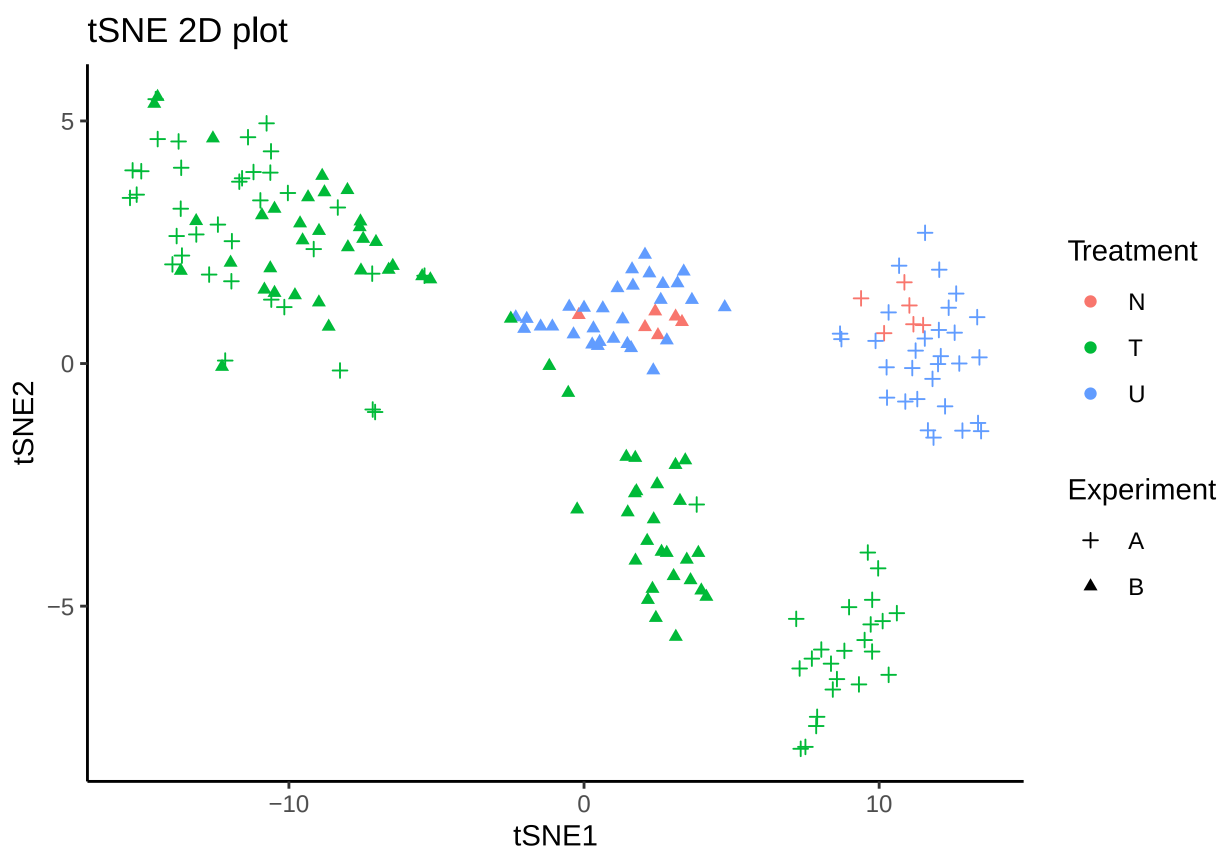
```
head(tsne2$Y)
```

```
##           [,1]      [,2]
## [1,] 10.855761 1.6739849
## [2,] 11.489761 0.7927729
## [3,]  9.387378 1.3435162
## [4,] 10.168390 0.6255299
## [5,] 11.161283 0.8102897
## [6,] 11.024912 1.1974697
```

```r
plot(tsne2$Y, col=treat)
```



```r
tsne_data <- as.data.frame(tsne2$Y)
tsne_data$Experiment <- meta_data$experiment
tsne_data$Treatment <- meta_data$treatment

library(ggplot2)
ggplot(tsne_data, aes(x=V1, y=V2)) +
  geom_point(aes(color=Treatment, shape=Experiment)) +
  scale_shape_manual(values = c(3,17)) +
  ggtitle("tSNE 2D plot") + xlab("tSNE1") + ylab("tSNE2") +
  theme_classic()
```



Q2 · Are `tsne1` and `tsne2` the same? Why?

A2. No, `tsne1` and `tsne2` are not identical, because tSNE generation is stochastic, the starting point in chosen randomly each time.

Explore `pca=TRUE`:

Q3. In the `Rtsne` function, what does the `pca` option mean?

A1. The `pca` option in the `tSNE` function determines whether an initial PCA step should be performed.

Run tSNE again and test the effect of setting `pca` to `TRUE/FALSE`.

Use `system.time()` to check time for each process.

```
t1 <- system.time(Rtsne(X, pca= TRUE)) # default
t2 <- system.time(Rtsne(X, pca= FALSE))
print(t1)
```

```
##    user  system elapsed
##   16.28    0.36   17.09
```

```
print(t2)
```

```
##    user  system elapsed
##   13.15    5.02   18.31
```

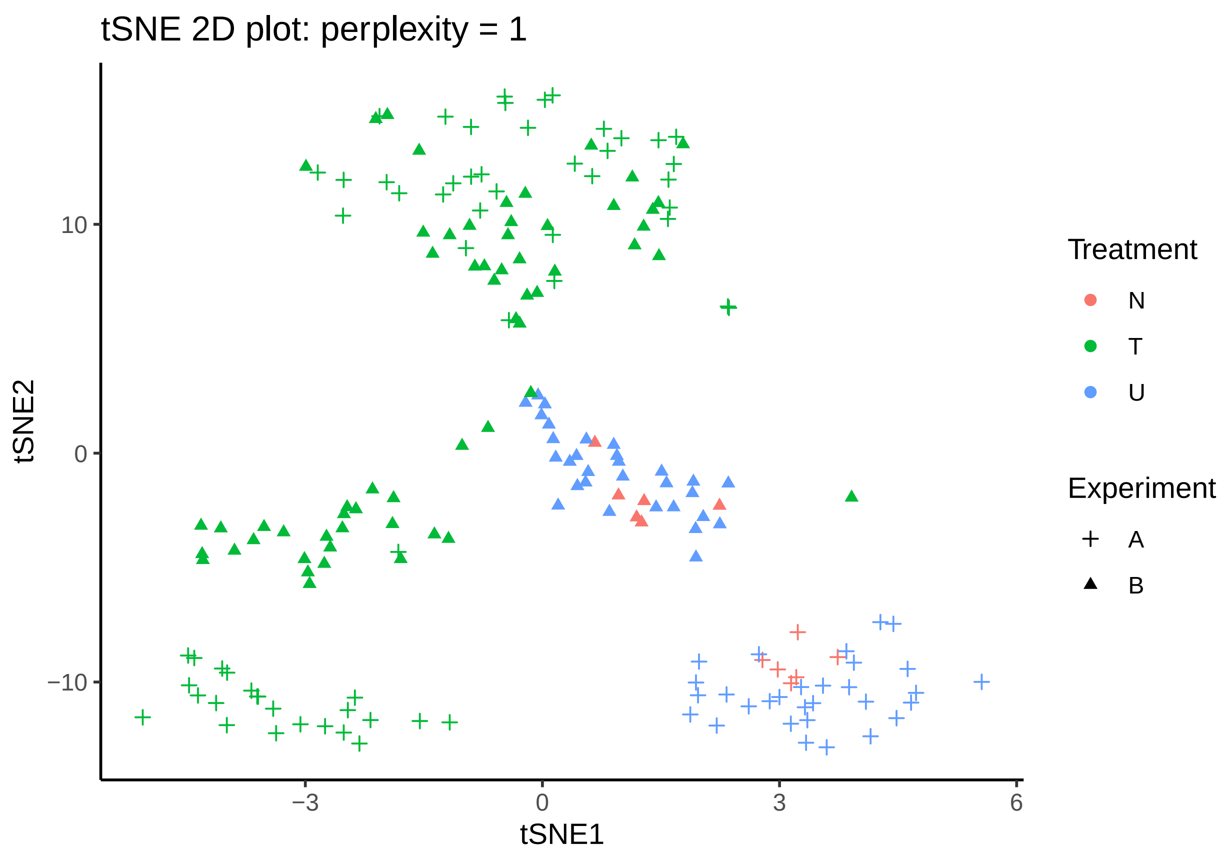
```
print(t1-t2)
```

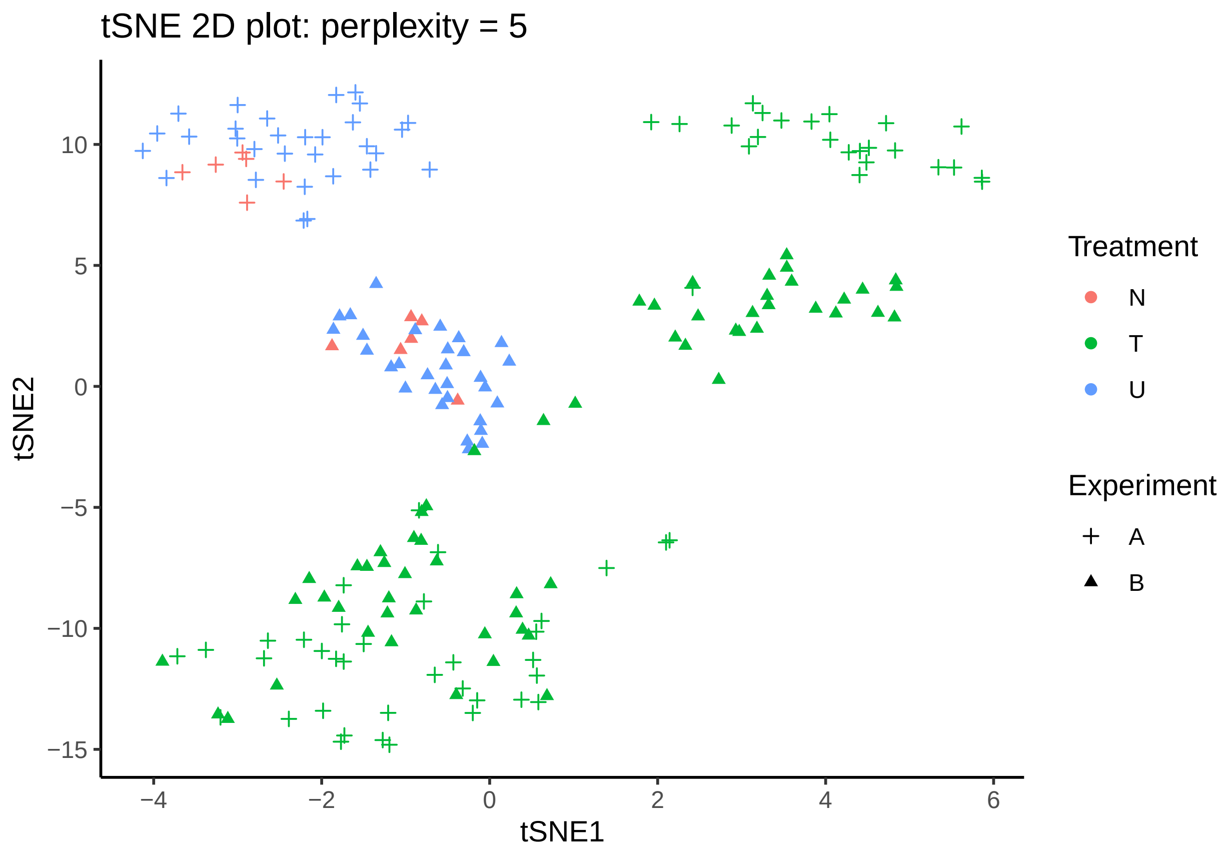
```
##    user  system elapsed
##    3.13   -4.66   -1.22
```

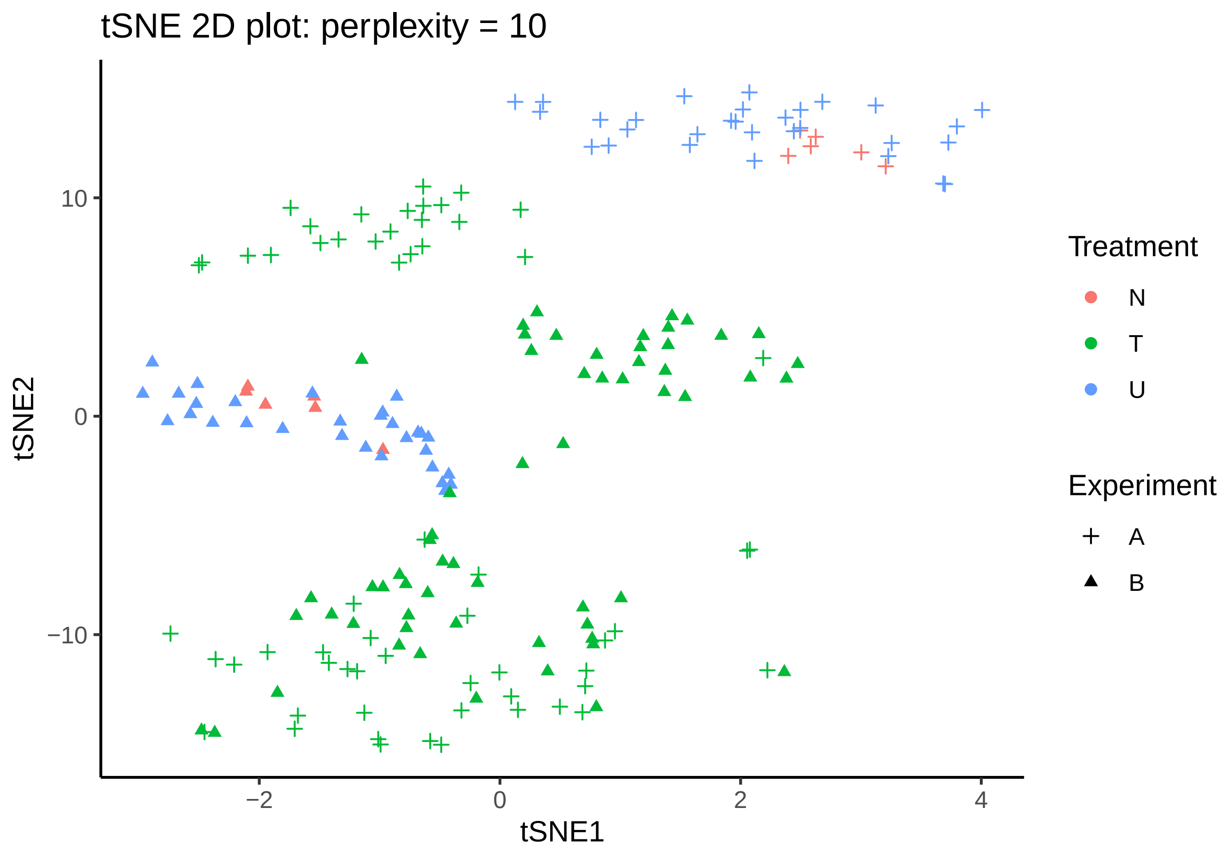
Explore `perplexity`:

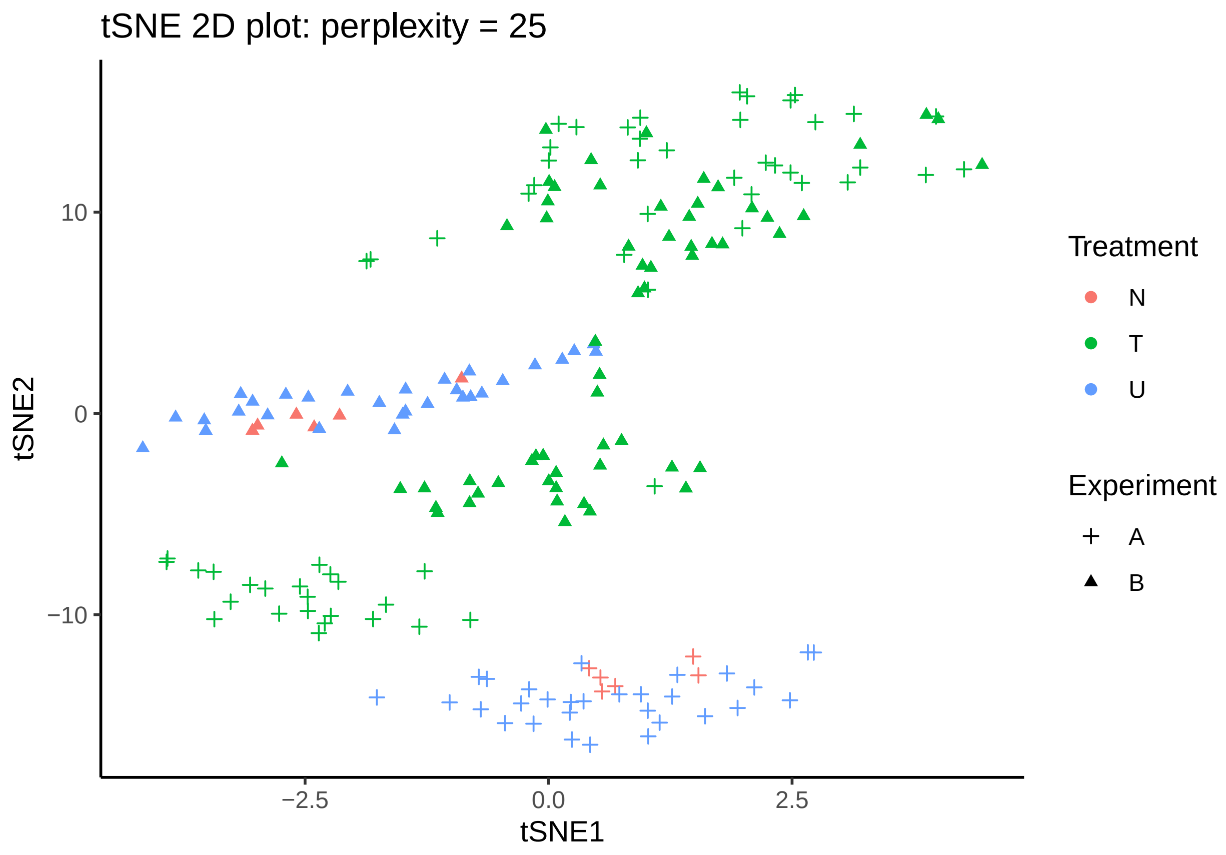
Create different tSNE with different perplexity values.

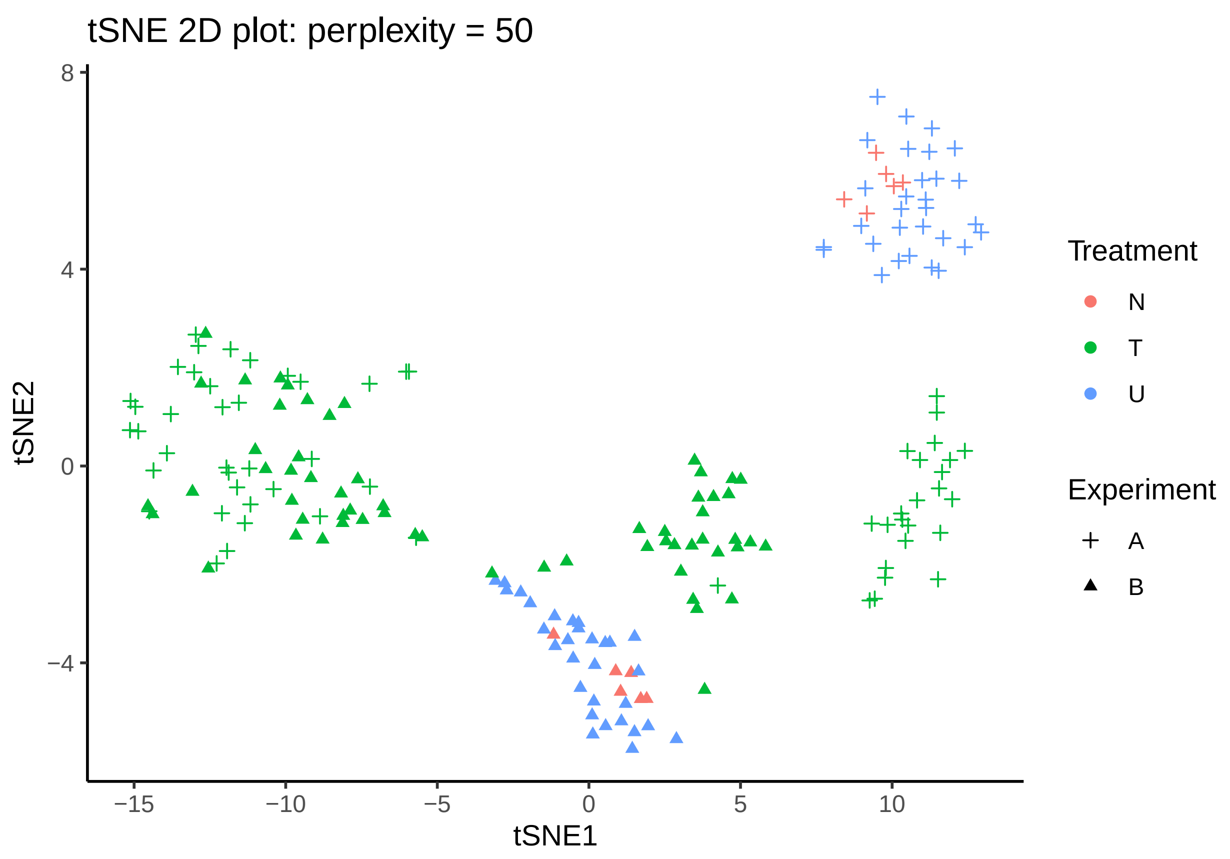
```
set.seed(123) ## allows us to reproduce results

for (p_value in c(1, 5, 10, 25, 50)) {
  tsne <- Rtsne(X)

  tsne_data <- as.data.frame(tsne$Y)
  tsne_data$Experiment <- meta_data$experiment
  tsne_data$Treatment <- meta_data$treatment

  library(ggplot2)
  p <- ggplot(tsne_data, aes(x=V1, y=V2)) +
    geom_point(aes(color=Treatment, shape=Experiment)) +
    scale_shape_manual(values = c(3,17)) +
    ggtitle(paste("tSNE 2D plot: perplexity =", p_value)) + xlab("tSNE1") + ylab("tSNE2") +
    theme_classic()
  print(p)
}
```




tSNE 2D plot: perplexity = 5
tSNE2
tSNE1
10
5
0
−5
−10
−15
−4
−2
0
2
4
6
Treatment
N
T
U
Experiment
A
B


tSNE 2D plot: perplexity = 10
tSNE2
tSNE1
10
0
−10
−2
0
2
4
Treatment
N
T
U
Experiment
A
B


tSNE 2D plot: perplexity = 25
tSNE2
tSNE1
10
0
−10
−2.5
0.0
2.5
Treatment
N
T
U
Experiment
A
B



Q4. What is the effect of perplexity?

A4. The perplexity value controls how many nearest neighbours are taken into account when constructing the embedding in the low-dimensional space: with high perplexity values, the clusters will be more dense and defined.

Try to set a very big perplexity value.

```
set.seed(123) ## allows us to reproduce results
tsne_test <- Rtsne(X1, perplexity = 100)
```

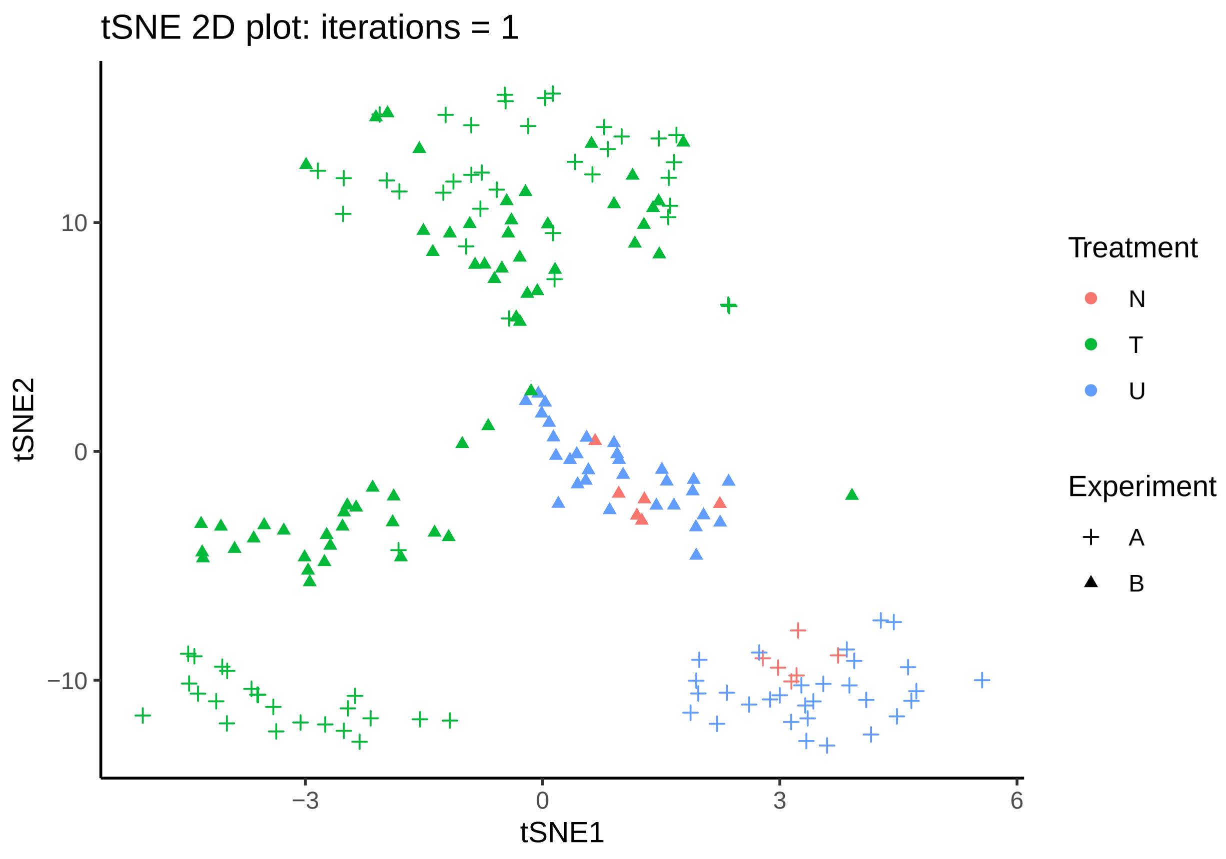
```
## Error in .check_tsne_params(nrow(X), dims = dims, perplexity = perplexity, : perplexity is too large
```

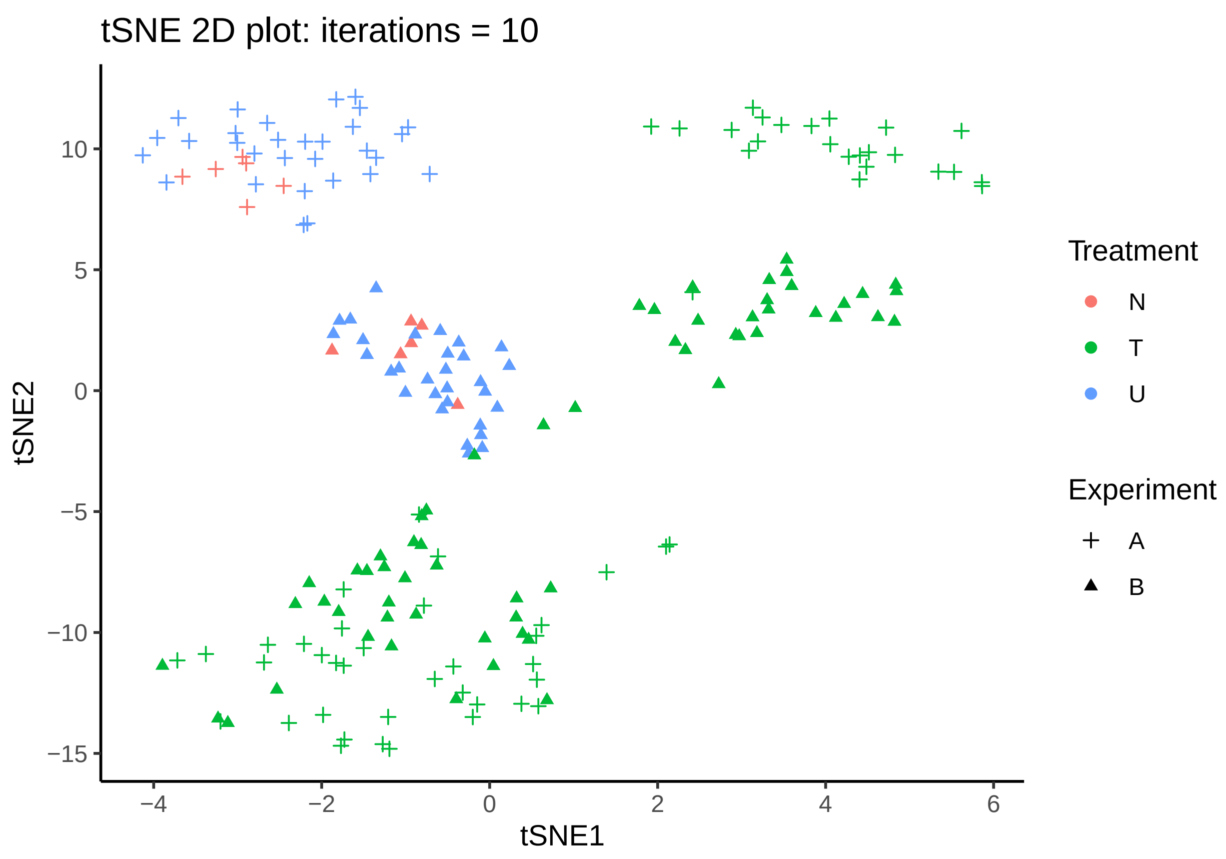
Q5. What is the effect of the number of iterations?

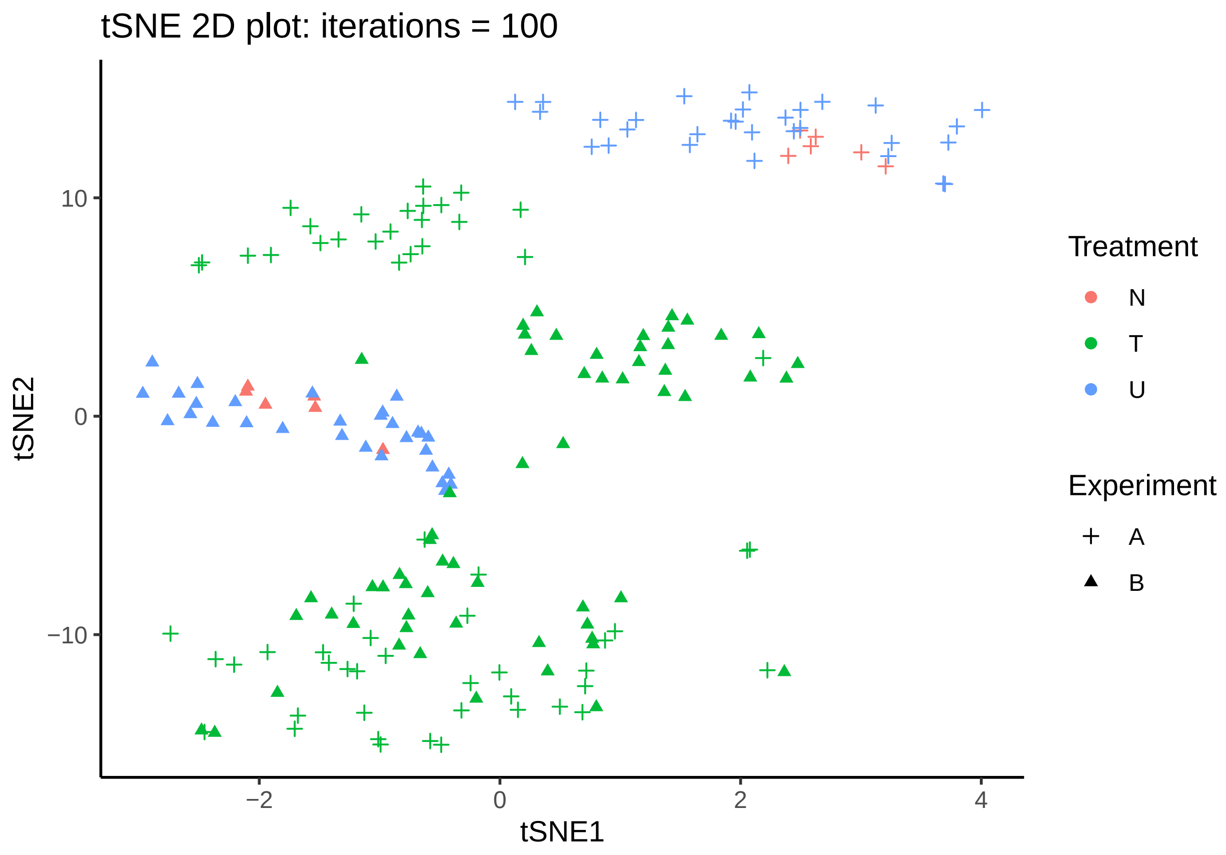
A5. With more and more iterations, the distance between clusters becomes bigger, so with higher values of `max_iter`, we get tSNE plots with more distinct and defined clusters

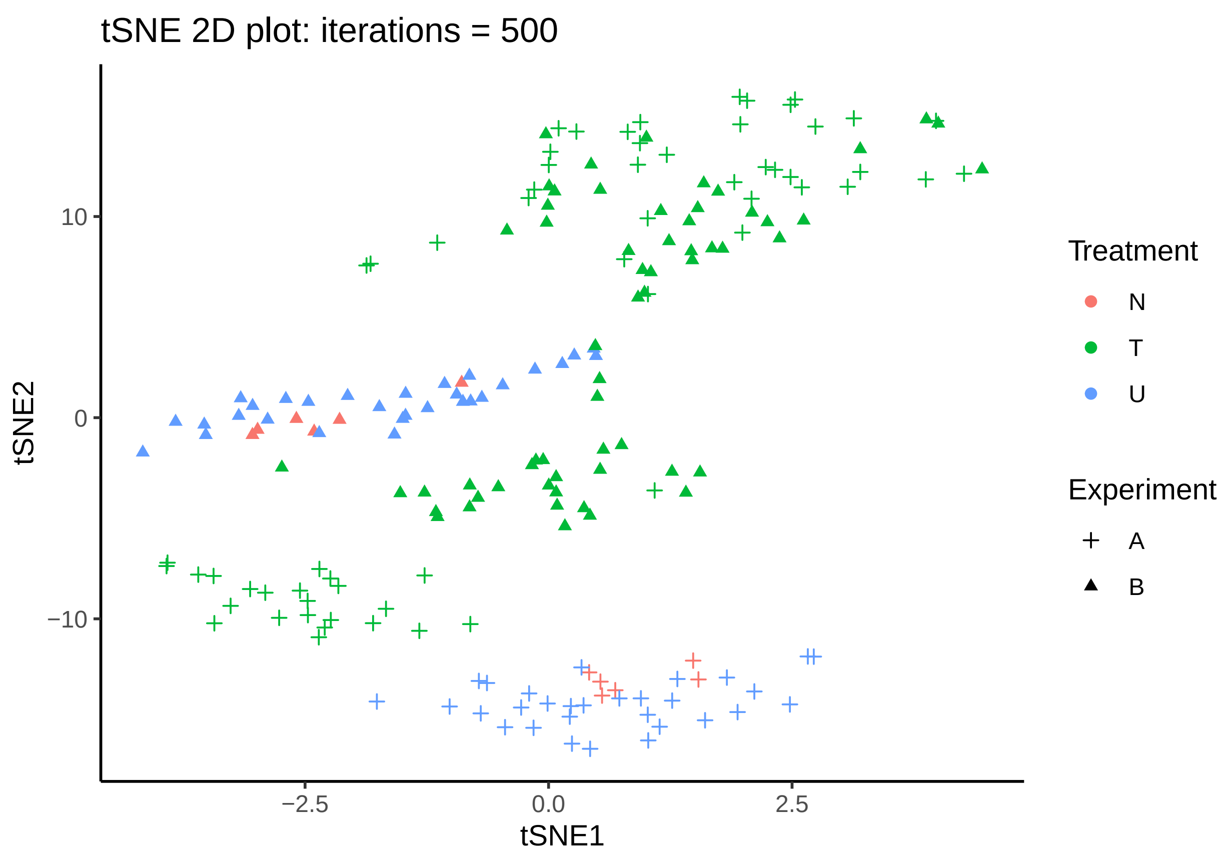
Create different tSNE with different iterations values.

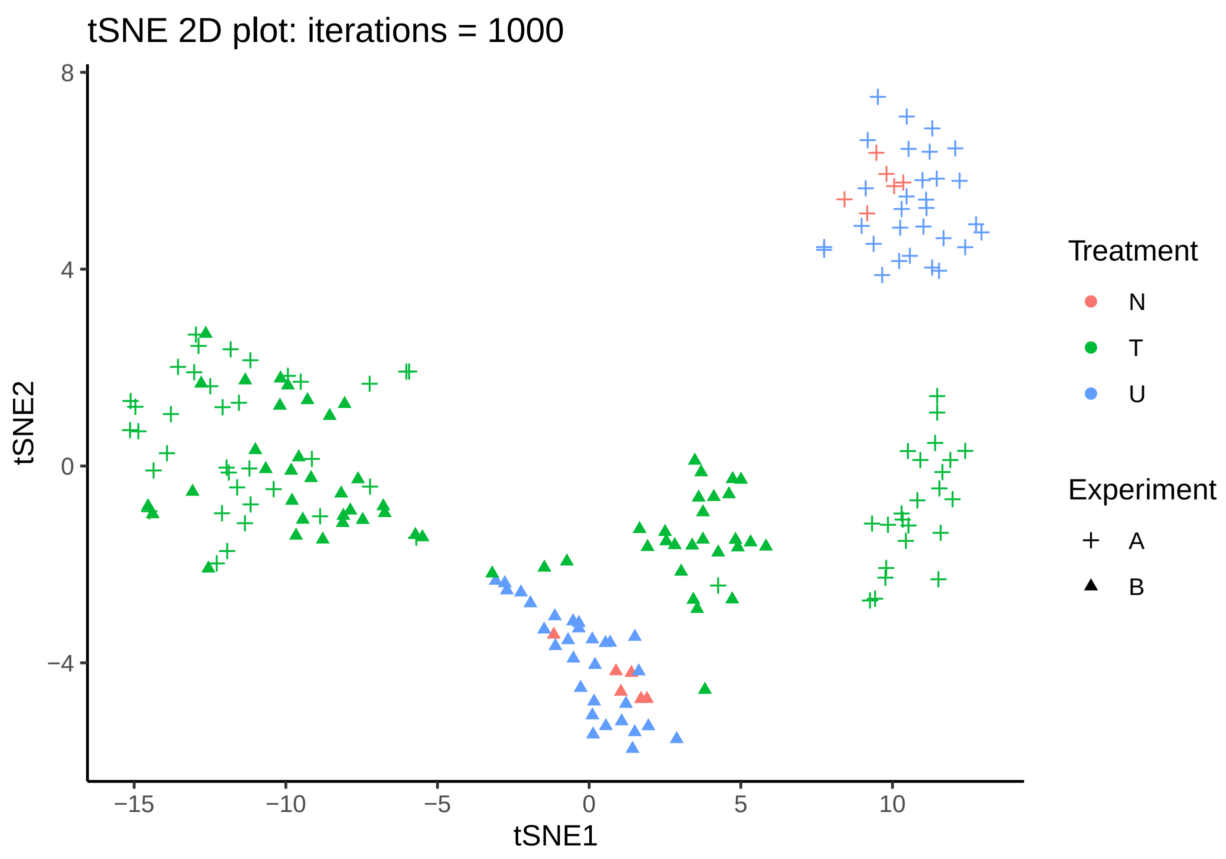
```
set.seed(123) ## allows us to reproduce results

for (max_iter in c(1, 10, 100, 500, 1000)) {
  tsne <- Rtsne(X)

  tsne_data <- as.data.frame(tsne$Y)
  tsne_data$Experiment <- meta_data$experiment
  tsne_data$Treatment <- meta_data$treatment

  library(ggplot2)
  p <- ggplot(tsne_data, aes(x=V1, y=V2)) +
   geom_point(aes(color=Treatment, shape=Experiment)) +
   scale_shape_manual(values = c(3,17)) +
   ggtitle(paste("tSNE 2D plot: iterations =", max_iter)) + xlab("tSNE1") + ylab("tSNE2") +
   theme_classic()
  print(p)
}
```




tSNE 2D plot: iterations = 10
tSNE2
tSNE1
10
5
0
−5
−10
−15
−4
−2
0
2
4
6
Treatment
N
T
U
Experiment
A
B


tSNE 2D plot: iterations = 100
tSNE2
tSNE1
10
0
−10
−2
0
2
4
Treatment
N
T
U
Experiment
A
B


tSNE 2D plot: iterations = 500
tSNE2
tSNE1
10
0
−10
−2.5
0.0
2.5
Treatment
N
T
U
Experiment
A
B



Q6. Identify and explain if there is any batch effect.

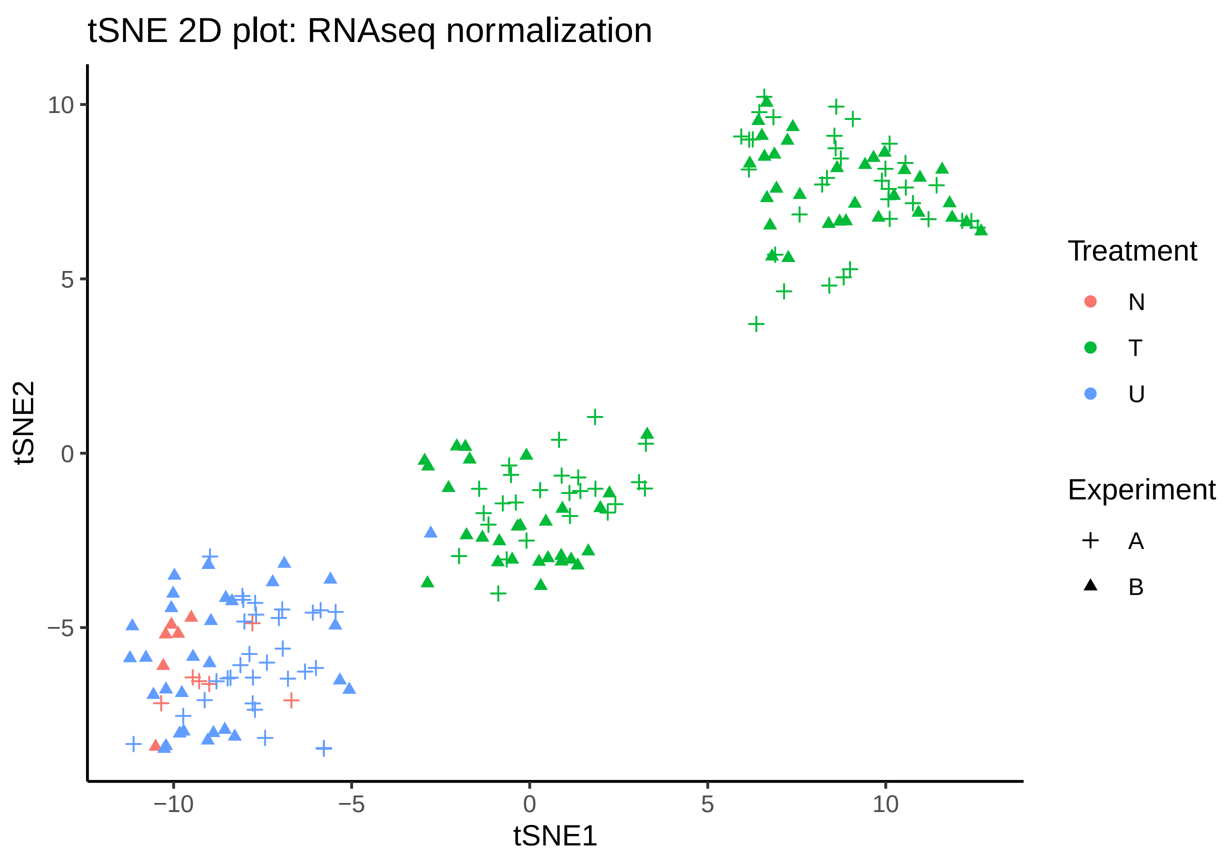
A6. Yes, there is evident batch effect as we get cdifferent clusters depending on the experiment from which the sample are coming from.

...

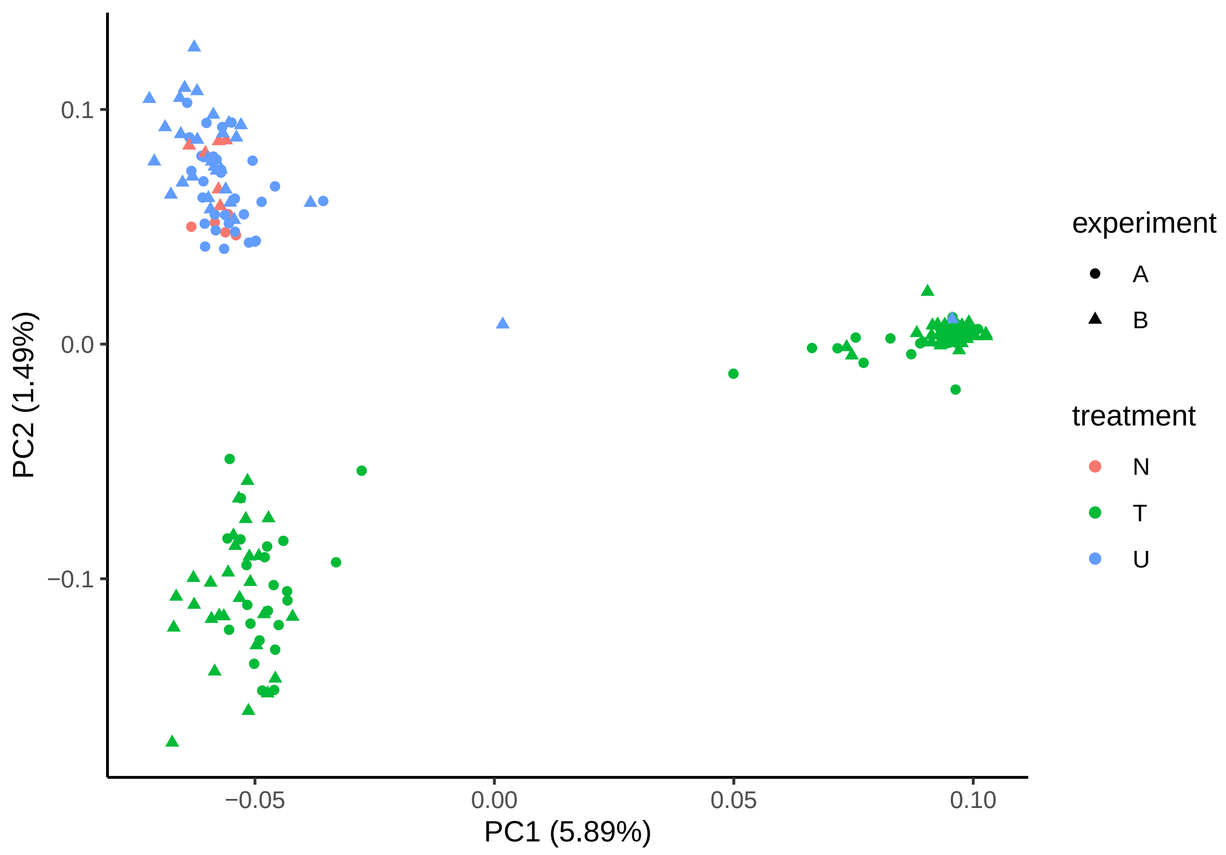
Final plots

Normalize and create PCA and tSNE:

```
X=X[, colSums(X)>0]
Y = X/rowSums(X)
tSNE_norm <- Rtsne(Y, pca=TRUE)

tsne_data3 <- as.data.frame(tSNE_norm$Y)
tsne_data3$Experiment <- meta_data$experiment
tsne_data3$Treatment <- meta_data$treatment

## ggplot
plot_p2 <- ggplot(tsne_data3, aes(x=V1, y=V2)) +
  geom_point(aes(color=Treatment, shape=Experiment)) +
  scale_shape_manual(values = c(3,17)) +
  ggtitle("tSNE 2D plot: RNAseq normalization") + xlab("tSNE1") + ylab("tSNE2") +
  theme_classic()

print(plot_p2)
```



```r
## create PCA
pca_norm <- stats::prcomp(Y, scale=TRUE)

library(ggfortify)
ggplot2::autoplot(pca_norm, data=meta_data, colour="treatment", shape="experiment") + theme_classic()
```



0.1
0.0
−0.1
PC2 (1.49%)
−0.05
0.00
0.05
0.10
PC1 (5.89%)
experiment
A
B
treatment
N
T
U