# Census notebook

```
library(Seurat)
```

```
## Attaching SeuratObject
```

```
library(data.table)
library(Matrix)
library(ggplot2)
source("census.R")
```

## Load Data

```
#mat <- Read10X("/home/data/CITEseq/RNA_3p/")
#meta <- fread("/home/data/CITEseq/GSE164378_sc.meta.data_3P.csv")
#mat <- Read10X("/home/Data/Hao (CITEseq-PBMC)/hto_5p/")
mat_r <- readMM("/home/Data/Hao (CITEseq-PBMC)/RNA_subset.mtx")
```

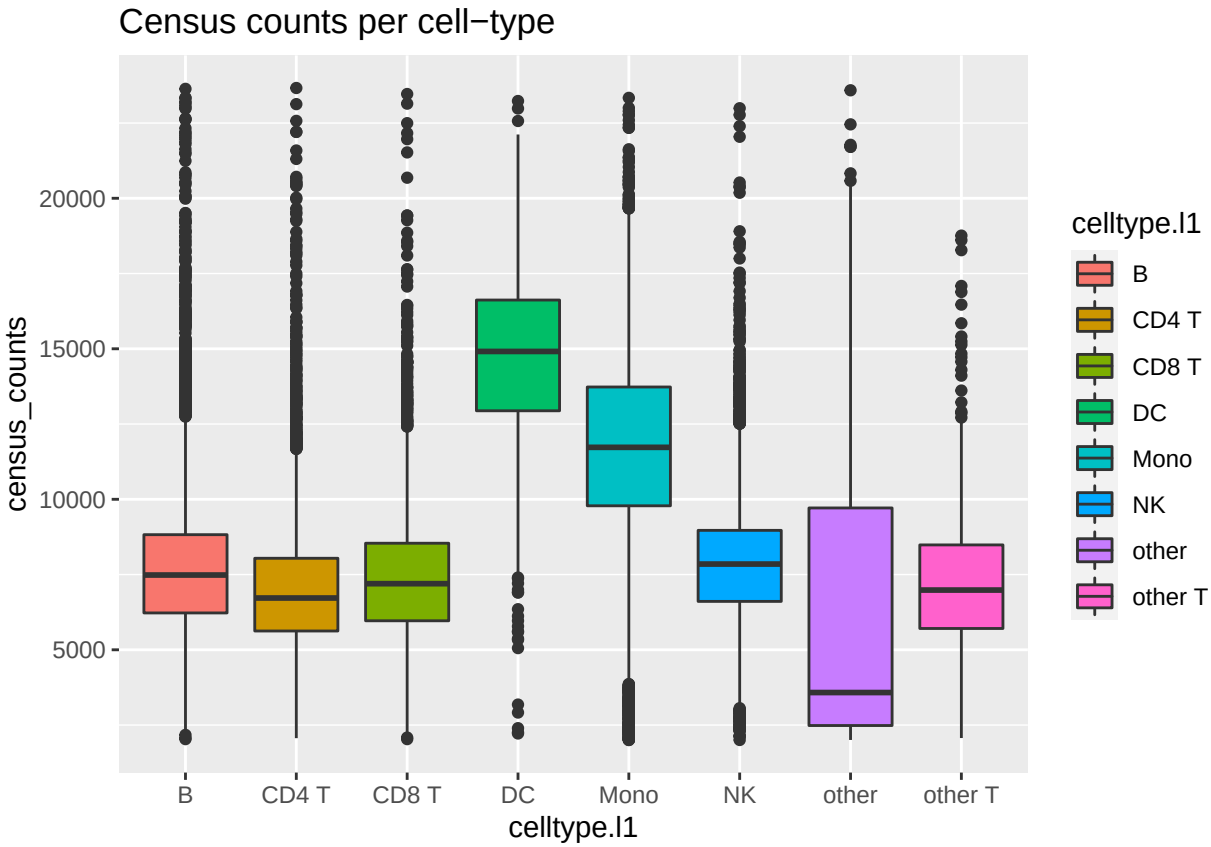
## Run Census (monocle version) on complete data

```
#only works on norm
#out_monocle <- as.data.frame(census(matrix = mat, ncores = 2, method = "monocle"))
#colnames(out_monocle) <- c("census_counts")
#df <- merge(out_monocle, meta, by.x=0, by.y="V1")
df <- fread("../../Data/Hao (CITEseq-PBMC)/census_meta.tsv")
```

```
## Warning in fread("../../Data/Hao (CITEseq-PBMC)/census_meta.tsv"): Detected
## 15 column names but the data has 16 columns (i.e. invalid file). Added 1 extra
## default column name for the first column which is guessed to be row names or an
## index. Use setnames() afterwards if this guess is not correct, or fix the file
## write command that created the file to create a valid file.
```

## Census counts per cell-type

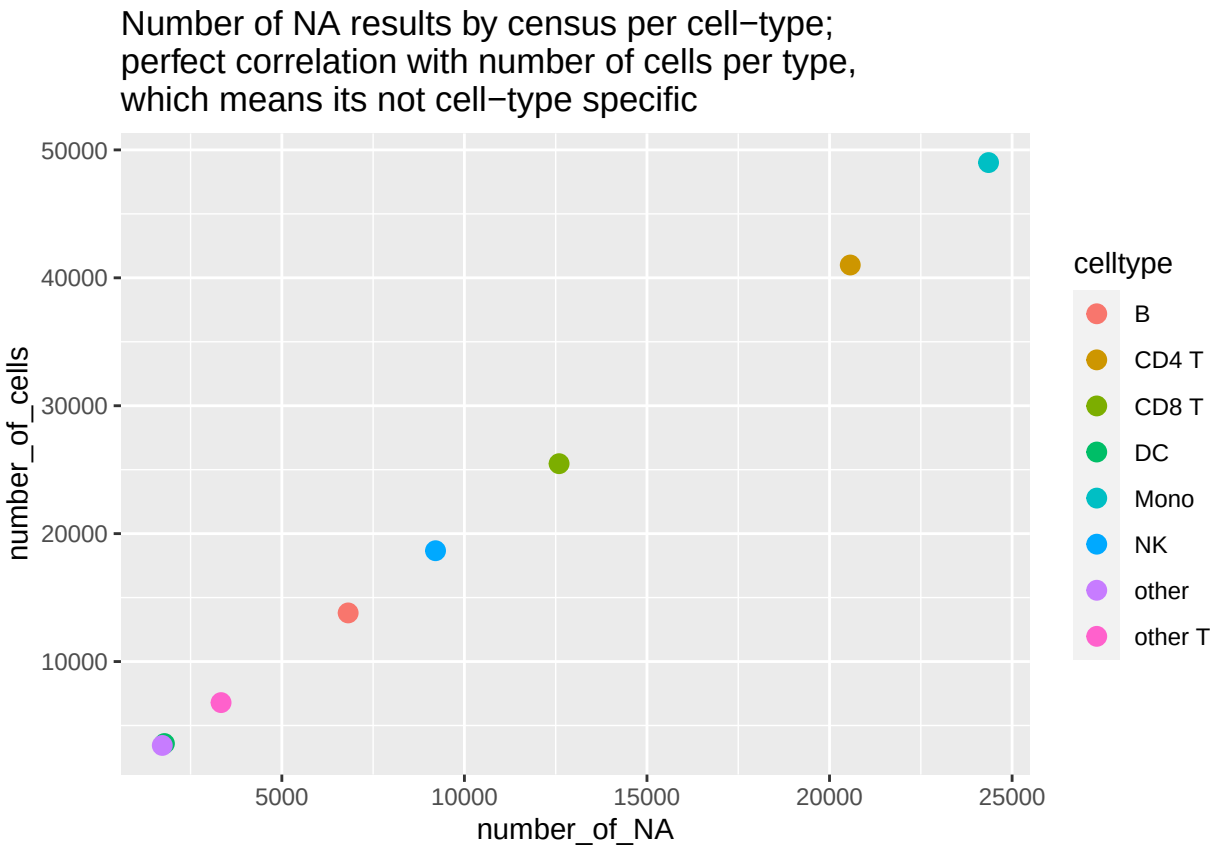
```
ggplot(df[!is.na(df$census_counts),], aes(y=census_counts,x=celltype.l1,fill=celltype.l1))+
  geom_boxplot()+
  ggtitle("Census counts per cell-type")
```




### Census produced ~50% NA values

```
df$isNA <- is.na(df$census_counts)
na_df <- df[,sum(isNA==T),by=celltype.l1]
colnames(na_df) <- c("celltype","number_of_NA")
total_df <- df[,.N,by=celltype.l1]
colnames(total_df) <- c("celltype","number_of_cells")
plot_df <- merge(na_df, total_df, by="celltype")

ggplot(plot_df, aes(x=number_of_NA,y=number_of_cells,color=celltype))+
  geom_point(size=3)+
  ggtitle("Number of NA results by census per cell-type; \nperfect correlation with number of cells per
```




## How are the NAs produced?

(all shown on a 10k cells subset of the dataset) When calculating **x*__i** (they call it *t_estimate* in their code), they log10 transform the expression values (counts in this case). The range of **x*__i** can be seen below (for 5k random cells). For the next steps, they only consider the genes with an expression value $> 0.1$. Later on, they calculate the number of single mRNA cells as all theses cells, which have a lower expression value than **x*__i**. This means if **x*__i** falls below 1, the formula finds no single mRNA cells (because they already removed all non-zero expression values from the cell) and we would have 0 as the numerator in the M_i fraction. This cannot work and we get NaN as result.

I think because TPMs have smaller values, the function might make more sense on then.

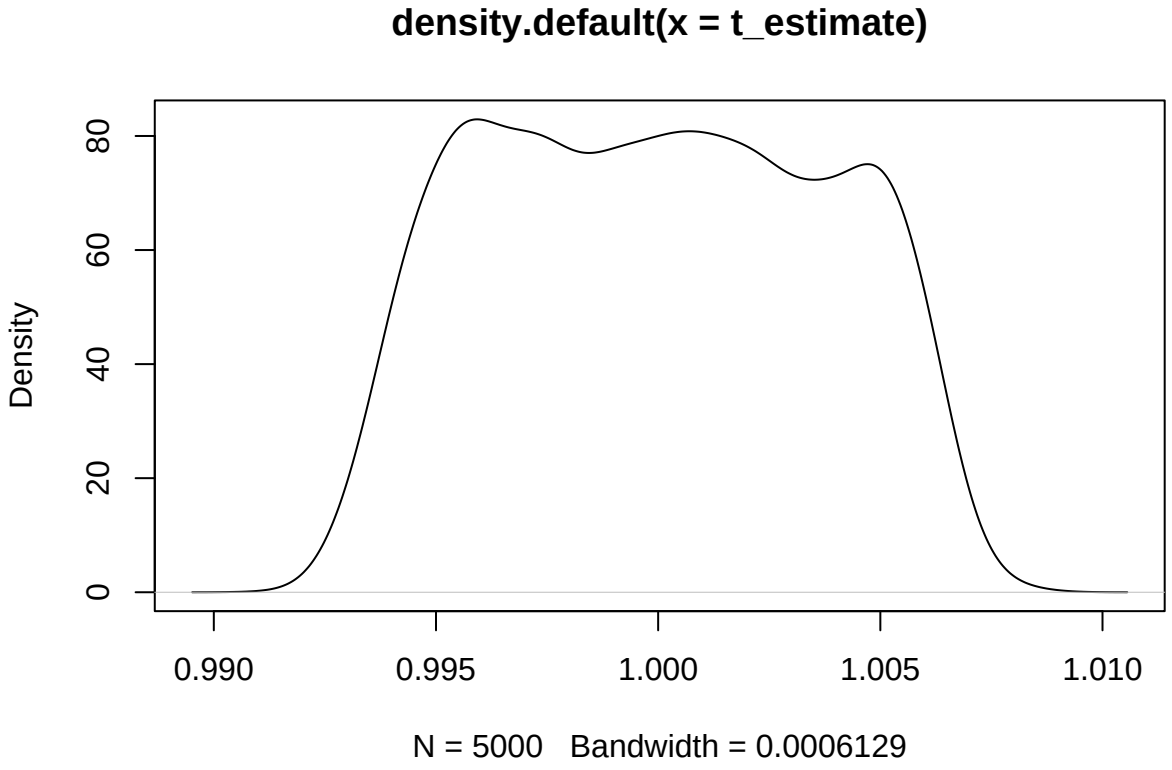
```
t_estimate <- census(matrix=mat_r, method="t_estimate", ncores=1)
```

```
## [1] "20 %"
## [1] "40 %"
## [1] "60 %"
## [1] "80 %"
## [1] "100 %"
```

```
summary(t_estimate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9914  0.9967  0.9999  0.9999  1.0031  1.0087
```

```
plot(density(t_estimate))
```


density.default(x = t_estimate)
Density
0
20
40
60
80
0.990
0.995
1.000
1.005
1.010
N = 5000 Bandwidth = 0.0006129