Use case: Benchmarking method parameters

Shian Su

2018-11-29

Contents

1 Introduction

This use case shows how to test a range of parameters for a given method. We will use the CelSeq2 mRNA mixture data and apply knn-smooth with various `k` parameter to see its effects on the output.

2 Setting up benchmark

```
library(CellBench)
library(dplyr)
library(purrr)
library(ggplot2)
```

We load in the data and create a list of 1 SingleCellExperiment object.

```
cellbench_mrna_mix_data <- load_mrna_mix_data()

data <- list(
    mrna_mix_celseq = cellbench_mrna_mix_data$mrna_mix_celseq
)

str(data, 1)
## List of 1
##  $ mrna_mix_celseq:Formal class 'SingleCellExperiment' [package "SingleCellExperiment"] with 10 slots
```

We need to write some small wrappers to help run pipelines and make methods uniform in input and output. This is necessary because each step of analysis should take in the same type of data and output the same type of data, however different methods may differ in how they are called, how many steps need to run and what they output. Wrappers help manage this, in this example we want our normalisation step to take in a SingleCellExperiment and output a normalised count matrix. The imputation step should take a count matrix and return an imputed counts matrix.

```
# take in a SingleCellExperiment and return a scran normalised
# expression matrix
scran_norm_expr <- function(x) {
    stopifnot(is(x, "SingleCellExperiment"))

    x <- scran::computeSumFactors(x)
    x <- scater::normalize(x, return_log = FALSE)

    SingleCellExperiment::normcounts(x)
}

# take in an expression matrix and return the imputed expression matrix
impute_knn_smooth <- function(expr, k) {
    source("https://raw.github.com/yanailab/knn-smoothing/master/knn_smooth.R")
    smoothed_mat <- knn_smoothing(mat = expr, k = k)
    smoothed_mat
}
```

We then create the lists of functions to use with CellBench. We only have one normalisation method, but for imputation we can create a series of partially applied functions with different `k` parameters. Here assuming we have `f(x, y)`, `partial(f, y = 1)` is equivalent to `function(x) f(x, y = 1)`, partial application "fills in" parameter values and returns a function that can be called.

```
norm_method <- list(
    scran = scran_norm_expr
)

# identity simply returns its argument, here it's used to represent
# no imputation
impute_method <- list(
    "none" = identity,
    "partial(k = 2)" = partial(impute_knn_smooth, k = 2),
    "partial(k = 4)" = partial(impute_knn_smooth, k = 4),
    "partial(k = 8)" = partial(impute_knn_smooth, k = 8),
    "partial(k = 16)" = partial(impute_knn_smooth, k = 16),
    "partial(k = 32)" = partial(impute_knn_smooth, k = 32)
)
```

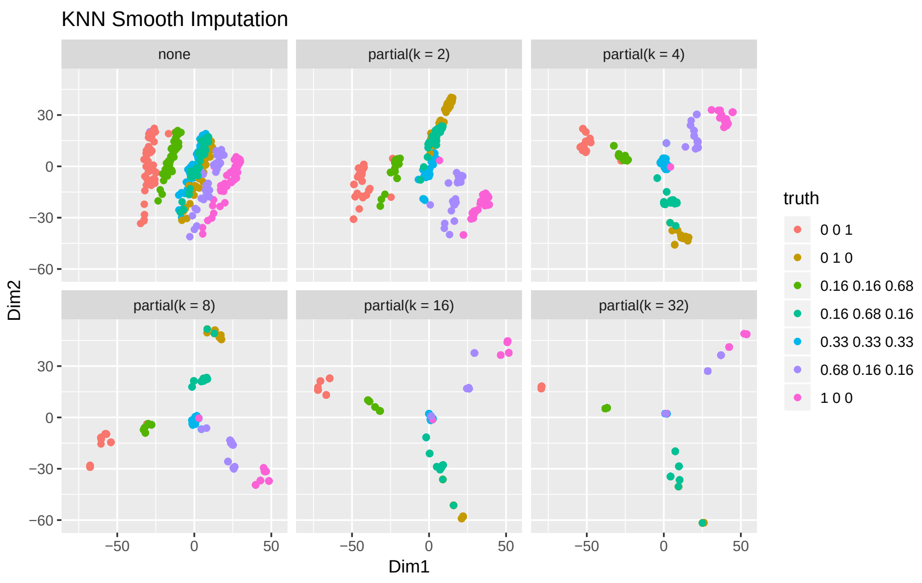
```
res_norm <- data %>%
    apply_methods(norm_method)
## Warning in .get_all_sf_sets(object): spike-in set 'ERCC' should have its own
## size factors

res_norm
## # A tibble: 1 x 3
##   data            norm_method result
##   <fct>           <fct>       <list>
## 1 mrna_mix_celseq scran       <dbl [14,804 x 340]>
```

```
res_impute <- res_norm %>%
    apply_methods(impute_method)

res_impute
## # A tibble: 6 x 4
##   data            norm_method impute_method   result
##   <fct>           <fct>       <fct>           <list>
## 1 mrna_mix_celseq scran       none            <dbl [14,804 x 340]>
## 2 mrna_mix_celseq scran       partial(k = 2)  <dbl [14,804 x 340]>
## 3 mrna_mix_celseq scran       partial(k = 4)  <dbl [14,804 x 340]>
## 4 mrna_mix_celseq scran       partial(k = 8)  <dbl [14,804 x 340]>
## 5 mrna_mix_celseq scran       partial(k = 16) <dbl [14,804 x 340]>
## 6 mrna_mix_celseq scran       partial(k = 32) <dbl [14,804 x 340]>
```

```
dim_red <- list(
    pca = compute_pca
)

# log-transform the counts
res_impute$result <- lapply(res_impute$result, function(x) log2(x + 1))

res <- res_impute %>%
    apply_methods(dim_red)

res
## # A tibble: 6 x 5
##   data             norm_method impute_method   dim_red result
##   <fct>            <fct>       <fct>           <fct>   <list>
## 1 mrna_mix_celseq scran       none            pca     <data.frame [340 x 2]>
## 2 mrna_mix_celseq scran       partial(k = 2)  pca     <data.frame [340 x 2]>
## 3 mrna_mix_celseq scran       partial(k = 4)  pca     <data.frame [340 x 2]>
## 4 mrna_mix_celseq scran       partial(k = 8)  pca     <data.frame [340 x 2]>
## 5 mrna_mix_celseq scran       partial(k = 16) pca     <data.frame [340 x 2]>
## 6 mrna_mix_celseq scran       partial(k = 32) pca     <data.frame [340 x 2]>
```

```
append_anno <- function(data_key, result) {
    # get mRNA amount
    col_data <- colData(cellbench_mrna_mix_data$mrna_mix_celseq)
    mRNA_amount <- col_data$mRNA_amount

    truth <- with(
        col_data,
        paste(H2228_prop, H1975_prop, HCC827_prop)
    )

    result %>%
        tibble::add_column(mRNA_amount, .before = TRUE) %>%
        tibble::add_column(truth, .before = TRUE)
}
```

```
annotated_res <- res %>%
    dplyr::mutate(data_key = paste(data)) %>%
    dplyr::mutate(result = map2(data_key, result, append_anno)) %>%
    dplyr::select(-data_key)

plot_df <- tidyr::unnest(annotated_res)

plot_df %>%
    ggplot(aes(x = Dim1, y = Dim2, col = truth)) +
    geom_point() +
    facet_wrap(~impute_method, nrow = 2) +
    ggtitle("KNN Smooth Imputation")
```



3 Conclusion