Compression Strategy Along inferred Pseudotime

ScMaSigPro Supplementary Material-I

Priyansh Srivastava, Stefan Götz, María José Nueda, Ana Conesa

2023-10-18

Contents

Introduction

Pseudotime Binning

scMaSigPro works by binning the raw expression data along the inferred Pseudotime. Previous studies have shown that the pseudo-bulk approaches for finding DE genes achieve the highest fidelity to the ground truth compared to approaches that modelled cells individually [Murphy and Skene, Junttila et al., Squair et al.]. Specifically, a cumulative sum of raw counts at the cell level (e.g. per cell type) followed by normalization yields superior performance in detecting DE genes (Junttila et al.).

Unlike clustering, where each bin or pseudo-bulk cluster contains cells of the same type, in trajectory inference (TI) related datasets, cellular states are arranged according to pseudotime. Therefore, it becomes not only crucial to group similar cells within a bin but also to preserve the original sequence of the cell state as dictated by the pseudotime.

scMaSigPro introduces a unique approach for binning that is tailored to data analysed in the context of pseudotime. It employs a three-step strategy to determine the optimal number of bins and the size of the bin width, based on the original range of the inferred pseudotime keeping the original sequence of the cell states unchanged.

Binning Steps

Histogram Binning

scMaSigPro offers a range of binning methods for pseudo-bulking, with "Sturges" set as the default. Here we will demosntrate the binning with "Sturges".

```
# Short Pseudotime with repetitions
short.rep.pseudotime <- data.frame(
  real_pseudotime = create_random_repeated_vector(1, 100, 1, 3)
)

# Print the time series
kable(head(short.rep.pseudotime))
```

Create a sample Time series data

real_pseudotime
1
2
2
2
3
4

Get the optimal number of bins

To get the optimal number of the bins with Sturges binning

- Let the inferred Pseudotime be represented as:

$$T_{Pseudotime} = \{t_1, t_2, t_3, t_4, ..., t_N\}$$

- Let the number of bins equal

$$N_{Bins} = (log2(N) + 1) * k$$

where, $k \in [0.3, \infty]$

- The interval width, I is then calculated by:

$$I = \frac{max(T_{Pseudotime}) - min(T_{Pseudotime})}{N_{Bins}}$$

- Finally,the index range of each bin, spanning from lowest to highest:

$$T^{Binned}_{Pseudotime} = \{indexofbin(I_{min} - I_{max})\}$$

```
# Extract the time series
time_series <- short.rep.pseudotime$real_pseudotime

# Number of Time points
time_points <- length(time_series)

# Number of Bins "Sturges Method"
estBins <- log2(time_points) + 1

# Multiply by drop.factor
estBins <- round(estBins * 0.7) # drop.fac

# Print the number of bins
estBins
```

```
## [1] 6
```

Bin Intervals

Using these calculated bin intervals I, the raw expression counts corresponding to each $T^{Binned}_{Pseudotime}$ value are aggregated into N_{Bins}.

Step-4: Calculate Bin intervals

```
bin_intervals <- as.data.frame(discretize(time_series,
  numBins = estBins,
  r = range(time_series)
))

colnames(bin_intervals) <- c("bin", "bin_size")
bin_intervals$customTime <- rownames(bin_intervals)

bin_table <- as.data.frame(t(as.data.frame(apply(bin_intervals, 1, create_range))))
colnames(bin_table) <- c("from", "to", "bin_size", "binnedTime")

short.rep.pseudotime.pooled <- as.data.frame(
  left_join(
    short.rep.pseudotime, bin_table,
    by = join_by(
      closest(real_pseudotime >= from),
      closest(real_pseudotime <= to)
    )
  )
)
kable(short.rep.pseudotime.pooled[c(c(1:3), c(29:33), c(55:58)), ])
```

	real_pseudotime	from	to	bin_size	binnedTime
1	1	1.0	17.5	37	1
2	2	1.0	17.5	37	1
3	2	1.0	17.5	37	1
29	15	1.0	17.5	37	1
30	15	1.0	17.5	37	1
31	15	1.0	17.5	37	1
32	16	1.0	17.5	37	1
33	16	1.0	17.5	37	1
55	28	17.5	34.0	30	2
56	28	17.5	34.0	30	2
57	29	17.5	34.0	30	2
58	29	17.5	34.0	30	2

Custom Function

Following function has been used to create a time-seires with repetitions, to mimic a pseudotime series.

```
# Define a function 'create_random_repeated_vector'
create_random_repeated_vector <- function(start, end, min_repetitions, max_repetitions) {
  repetitions <- sample(min_repetitions:max_repetitions,
    size = end - start + 1, replace = TRUE
  )
  result_vector <- rep(seq(from = start, to = end, by = 1),
    times = repetitions
  )
  return(result_vector)
}

# Define a function 'discretize'
discretize <- function(x, numBins, r = range(x)) {
  b <- seq(from = r[1], to = r[2], length.out = numBins + 1)
```

```
  cut_x <- cut(x, breaks = b, include.lowest = TRUE)
  y <- table(cut_x)
  return(y)
}

# Define a function 'create_range'
create_range <- function(x) {
  y <- as.character(x[["bin"]])
  y <- y %>% stringr::str_remove_all(pattern = "\\[|\\]|\\(|\\)")
  y1 <- as.numeric(sapply(strsplit(y, ","), "[", 1))
  y2 <- as.numeric(sapply(strsplit(y, ","), "[", 2))
  rangeVec <- c(y1, y2, x[["bin_size"]], x[["customTime"]])
  return(as.numeric(rangeVec))
}
```

References

Sini Junttila, Johannes Smolander, and Laura L Elo. Benchmarking methods for detecting differential states between conditions from multi-subject single-cell RNA-seq data. 23(5):bbac286. ISSN 1467-5463, 1477-4054. doi: 10.1093/bib/bbac286. URL https://academic.oup.com/bib/article/doi/10.1093/bib/bbac286/6649780.

Alan E. Murphy and Nathan G. Skene. A balanced measure shows superior performance of pseudobulk methods in single-cell RNA-sequencing analysis. 13(1):7851. ISSN 2041-1723. doi: 10.1038/s41467-022-35519-4. URL https://www.nature.com/articles/s41467-022-35519-4.

Jordan W. Squair, Matthieu Gautier, Claudia Kathe, Mark A. Anderson, Nicholas D. James, Thomas H. Hutson, Rémi Hudelle, Taha Qaiser, Kaya J. E. Matson, Quentin Barraud, Ariel J. Levine, Gioele La Manno, Michael A. Skinnider, and Grégoire Courtine. Confronting false discoveries in single-cell differential expression. 12(1):5692. ISSN 2041-1723. doi: 10.1038/s41467-021-25960-2. URL https://www.nature.com/articles/s41467-021-25960-2.