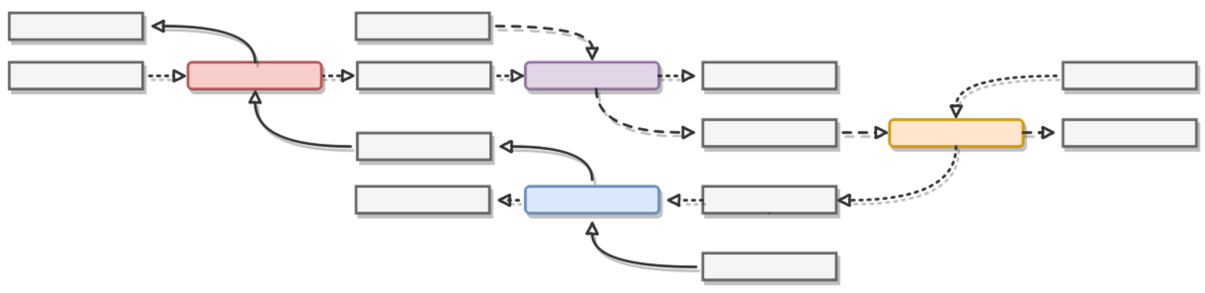
Assembly graphs, streaming partitioning, RNA-seq

Row 1

Characterizing RNA-seq assembly graphs: when is enough, enough?



Camille Scottand C. Titus Brown

Background: With sequencing experiments regularly reaching into the billions of fragments, assembly graphs have become a core feature of most extant assemblers. Traversals of these graphs yield images of the underlying sequence, and the assembled sequences that result are studied in downstream analyses. However, less studied are features of the assembly graph itself. Motivated by the observation that assembly graphs succinctly encode a universe of possible assemblies, we explore some fundamental features of assembly graphs from RNA-seq. Our work is guided by the question: how much sequence is needed to build a reliable and useful image of the underlying transcripts? Using a large body of RNA-seq experiments available through the Marine Microbial Eukaryotic Sequencing Project (MMETSP) and a streaming assembly graph partitioner, we describe transcriptome assembly graphs though their component size and coverage distributions.

Row 2

Methods

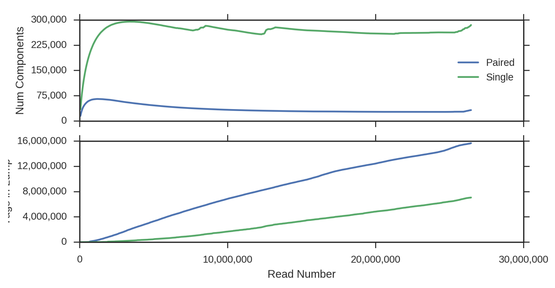
Assembly Graph Construction We use a typical de Bruijn graph of order \(k\) for our assembly graphs. The graph is succinctly encoded using either a bloom filter or Count-min sketch as implemented in the khmer package (Crusoe et al. 2015), depending on whether \(k\)-mer presence only or \(k\)-mer counting is desired. As reads are parsed from a sample, they are broken down into \(k\)-mers and inserted into the filter; we also keep an auxillary sparse map of \(k\)-mers at maximum distance \(d=(k*2)-1\) to act as indices into the graph, which we refer to as tags.

Streaming Partitioning Components are tracked online using a streaming partitioning algorithm. Briefly, when a read with \(k\)-mers \(R\) is inserted in the graph \(G\), the tags it intersects and any newly created tags are gathered in a set \(T\). A (user-selectable) partitioning function \(F(T, G) \rightarrow \mathbb{R}\) scores the insert, triggering partitioning if the score exceeds a threshold. The tags are partitioned (ie, assigned to a component) by breadth-first search, starting from those \(k\)-mers in \(R\) from which there is no path through \(R\cap G\) to an already-partitioned tag, and ending when all nearest tags in \(G/R\) have been found. All components associated with that set are then merged into its largest component.

Graph Artifacts

Early work showed that assembly graphs from RNA-seq data tend to become dominated by one highly connected component, even when error trimming and adapter removal is performed. This component, which we call the "lump," is composed of low complexity sequence and high-degree nodes.

Fig. 1: "Lump" formation as reads are added to the graph.



Another Row

Component Size and Coverage Dynamics

Fig. 2: Quantiles of component coverage and size as reads are inserted.

Here, our partitioning function favors mean tags counts between 5 and 10. The bottom decile is thus dominated by low-coverage, small components. Coverage in higher quantiles grows slowly, after an initial peak: the highest-coverage graph regions, comprising mostly low-complexity sequence and high-degree nodes, are consumed by the lump. Note the relatively linear growth of the largest components, compared to the stability of the smaller ones.

Samples for both figures taken from the MMETSP project (Keeling 2014, Cohen, Alexander, and Brown (2017)), SRR1300451.

Discussion

On RNA-seq Assembly Graphs How to approach the connectivity problem is an ongoing challenge faced by practitioners using assembly graphs. Traditional methods tend to break up the graph at chains of high-degree nodes, or more recently, use community-detection algorithms (Kannan et al. 2016). An improved solution might prevent the lump from forming in the first place by biasing assembly graph contruction.

These results confirm that the basic structure of the assembly graph is sketched out by only a small subset of a sample; by the time the graph has filled out enough to resemble its final component structure, coverage reaches relatively stable growth, which suggests that coverage information can be estimated early on in a streaming capacity as well.

On the Methods Current work is progressing toward better partition scoring by harnessing the detailed information on local and global component-wise coverage that we gather. The current streaming partitioning implementation is both information-rich and fast, exceeding the speed of a decent network connection. Streaming partitioning (and by extension, assembly) could operate as a filter for receiving sequence data, delivering components (or transcripts) as they become available, or acting as a persistent service for applications like resequencing experiments.

Row 3

Humans Were Involved

Contact

- Camille Scott
  University of California Davis
- cswel@ucdavis.edu

- https://github.com/camillescott

Thanks to the DIB Lab, the Python community, the MSU HPCC, and the various friends Camille has talked assembly graphs with.

And Software

These analyses are implemented in the `boink` package (https://github.com/camillescott/boink), which is built on the `khmer/oxli` package from the Lab for Data Intensive Biology (http://ivory.idyll.org/lab/) at UCD.

References

Cohen, Lisa, Harriet Alexander, and C. Titus Brown. 2017. "Marine Microbial Eukaryotic Transcriptome Sequencing Project, re-assemblies," January. doi:10.6084/m9.figshare.3840153.v6.

Crusoe, MR, HF Alameldin, S Awad, E Boucher, A Caldwell, R Cartwright, A Charbonneau, et al. 2015. "The Khmer Software Package: Enabling Efficient Nucleotide Sequence Analysis Version 1; Referees: 2 Approved, 1 Approved with Reservations." *F1000Research* 4 (900). doi:10.12688/f1000research.6924.1.

Kannan, Sreeram, Joseph Hui, Kayvon Mazooji, Lior Pachter, and David Tse. 2016. "Shannon: An Information-Optimal de Novo Rna-Seq Assembler." *BioRxiv*. Cold Spring Harbor Labs Journals. doi:10.1101/039230.

Keeling, Fabien AND Wilcox, Patrick J. AND Burki. 2014. "The Marine Microbial Eukaryote Transcriptome Sequencing Project (Mmetsp): Illuminating the Functional Diversity of Eukaryotic Life in the Oceans Through Transcriptome Sequencing." *PLOS Biology* 12 (6). Public Library of Science: 1–6. doi:10.1371/journal.pbio.1001889.

```
---
title: "Assembly graphs, streaming partitioning, RNA-seq"
output:
  flexdashboard::flex_dashboard:
    self_contained: false
    orientation: rows
    source: embed
    social: menu
geometry: -paperwidth=36in
          -paperheight=48in
bibliography: ../references/references.bib
---

body {
   font-size: 13px;
}


```{r setup, include=FALSE}
library("flexdashboard")
library("tidyverse")

# Setup knitr
knitr::opts_chunk$set(
  echo = FALSE, message = FALSE, warning = FALSE,
  # Save all figures in the output dir, you have to include them explicitly
  # with an  tag
  fig.path = "../output/img/", fig.show = "hide"
)
```
```

Row 1 {data-height=20}

###

Characterizing RNA-seq assembly graphs: when is enough, enough?

Camille Scott
and C. Titus Brown

Background: With sequencing experiments regularly reaching into the billions of fragments,
assembly graphs have become a core feature of most extant assemblers. Traversals of these
graphs yield images of the underlying sequence, and the assembled sequences that
result are studied in downstream analyses. However, less studied are features of the assembly graph itself. Motivated by the observation that assembly graphs succinctly encode a universe of possible assemblies, we explore some fundamental features of assembly graphs from RNA-seq. Our work is guided by the question: how much sequence is needed to build a reliable and useful image of the underlying transcripts? Using a

Row 2 {data-height=45}

Methods

Assembly Graph Construction

We use a typical de Bruijn graph of order k for our assembly graphs. The graph is succinctly encoded using
either a bloom filter or Count-min sketch as implemented in the khmer package [@khmer], depending on whether
k-mer presence only or k-mer counting is desired. As reads are parsed from a sample, they are broken down
into k-mers and inserted into the filter; we also keep an auxillary sparse map of k-mers at maximum distance
$d=(k*2)-1$ to act as indices into the graph, which we refer to as tags.

Streaming Partitioning

Components are tracked online using a streaming partitioning algorithm. Briefly, when a read with k-mers R is inserted in the graph G, the tags
it intersects and any newly created tags are gathered in a set T. A (user-selectable) partitioning function
$F(T, G) \rightarrow \mathbb{R}$ scores the insert, triggering partitioning if the score exceeds a threshold. The tags
are partitioned (ie, assigned to a component) by breadth-first search, starting from those k-mers in R from which there is no path through $R\cap G$ to an already-partitioned tag, and ending when all nearest tags in G/R have been found. All components associated with that set are then merged into its largest component.

Graph Artifacts

Early work showed that assembly graphs from RNA-seq data tend to become dominated by one highly connected
component, even when error trimming and adapter removal is performed. This component, which we call
the "lump," is composed of low complexity sequence and high-degree nodes.

Fig. 1: "Lump" formation as reads are added to the graph.

Another Row {data-height=50}

Component Size and Coverage Dynamics

Fig. 2: Quantiles of component coverage and size as reads are inserted.

Here, our partitioning function favors mean tags counts between 5 and 10. The bottom decile is thus dominated by
low-coverage, small components. Coverage in higher quantiles grows slowly, after an initial peak: the highest-coverage
graph regions, comprising mostly low-complexity sequence and high-degree nodes, are consumed by the lump. Note the
relatively linear growth of the largest components, compared to the stability of the smaller ones.

Samples for both figures taken from the MMETSP project [@plos_mmetsp, @Cohen2017], SRR1300451.

Discussion

On RNA-seq Assembly Graphs

How to approach the connectivity problem is an ongoing challenge faced by practitioners using assembly graphs.
Traditional methods tend to break up the graph at chains of high-degree nodes, or more recently, use
community-detection algorithms [@Kannan]. An improved solution might prevent the lump from forming in the first place
by biasing assembly graph contruction.

These results confirm that the basic structure of the assembly graph is sketched out by only a small subset
of a sample; by the time the graph has filled out enough to resemble its final component structure, coverage
reaches relatively stable growth, which suggests that coverage information can be estimated early on in a streaming
capacity as well.

On the Methods

Current work is progressing toward better partition scoring by harnessing the detailed information on local
and global component-wise coverage that we gather. The current streaming partitioning implementation is both
information-rich and fast, exceeding the speed of a decent network connection. Streaming
partitioning (and by extension, assembly) could operate as a filter for receiving sequence data,
delivering components (or transcripts) as they become available, or acting as a persistent service for applications
like resequencing experiments.

Row 3 {data-height=30}

Humans Were Involved {data-width=400}

Contact

- Camille Scott

  University of California Davis
- cswel@ucdavis.edu
- https://github.com/camillescott

Thanks to the DIB Lab, the Python community, the MSU HPCC, and the various friends Camille has talked
assembly graphs with.

And Software

These analyses are implemented in the `boink` package (https://github.com/camillescott/boink), which is
built on the `khmer/oxli` package from the Lab for Data Intensive Biology
(http://ivory.idyll.org/lab/) at UCD.

{.small}

References