Introduction to the pageRank Package

Hongxu Ding
Department of Biomolecular Engineering and Genomics Institute,
University of California, Santa Cruz, Santa Cruz, CA, USA

May 1, 2024

Contents

1 Introduction 1
  1.1 Background ....................................................... 1
  1.2 Installation ..................................................... 2

2 PageRank Analysis 2
  2.1 Temporal PageRank ............................................. 2
  2.2 Multiplex PageRank ............................................ 3
  2.3 Adjusting PageRank Calculations ............................ 4

3 Prioritizing TFs in GRNs 5
  3.1 Generating GRNs from Multi-Omics Profiles ................. 5
  3.2 Filter GRNs with Expression Profiles ........................ 7
  3.3 Session Information ........................................... 27

4 References 30

1 Introduction

1.1 Background

The pageRank package provides implementations of temporal PageRank as defined by [1], as well as multiplex PageRank as defined by [2]. As the extension of original steady-state PageRank [3,4] in temporal networks, temporal PageRank ranks nodes based on their connections that change over time. Multiplex PageRank, on the other hand, extends PageRank analysis to multiplex networks. In such networks, the same nodes might interact with one another in different layers. Multiplex PageRank is calculated according to the topology of a predefined
PageRank-related approaches can be applied to prioritize key transcriptional factors (TFs) in gene regulatory networks (GRNs). Specifically, the pageRank package provides functions for generating temporal GRNs from corresponding static counterparts. The pageRank package also provides functions for converting multi-omics, e.g. gene expression, chromatin accessibility and chromosome conformation profiles to multiplex GRNs. Such temporal and multiplex GRNs can thus be used for temporal and multiplex PageRank-based TF prioritization, respectively.

1.2 Installation

pageRank requires the R version 4.0 or later, packages BSgenome.Hsapiens.UCSC.hg19, TxDb.Hsapiens.UCSC.hg19, org.Hs.eg.db, annotate, GenomicFeatures, JASPAR2018, TFBSTools and bcellViper, to run the examples. After installing R, all required components can be obtained with:

```r
if (!requireNamespace("BiocManager", quietly=TRUE)) install.packages("BiocManager")
BiocManager::install("BSgenome.Hsapiens.UCSC.hg19")
BiocManager::install("TxDb.Hsapiens.UCSC.hg19.knownGene")
BiocManager::install("org.Hs.eg.db")
BiocManager::install("annotate")
BiocManager::install("GenomicFeatures")
BiocManager::install("JASPAR2018")
BiocManager::install("TFBSTools")
BiocManager::install("bcellViper")
```

2 PageRank Analysis

2.1 Temporal PageRank

We applied `diff_graph()` to calculate temporal PageRank. This is a simplified version of temporal PageRank described by [1] by only analyzing temporally adjacent graph pairs.

```r
> library(pageRank)
> set.seed(1)
> graph1 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph1)$name <- 1:100
> # the 1st graph with name as vertex attributes
> set.seed(2)
> graph2 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph2)$name <- 1:100
> # the 2nd graph with name as vertex attributes
> diff_graph(graph1, graph2)
```
igraph 8077800 DN-- 98 190 --
+ attr: name (v/c), pagerank (v/n), moi (e/n)
+ edges from 8077800 (vertex names):
[1] 1 ->60 2 ->15 2 ->57 3 ->10 3 ->16 3 ->84 4 ->43 5 ->20 5 ->66
[10] 5 ->72 5 ->81 6 ->25 6 ->50 7 ->37 7 ->67 7 ->73 7 ->85
[55] 27->43 28->98 29->95 30->84 32->49 33->10 34->99 35->81
[64] 36->17 36->33 36->45 36->53 36->77 37->33 37->54 38->6 38->17
+ ... omitted several edges

Differential graph graph1-graph2 will be outputed. The Differential graph has "moi (mode of interaction, 1 and -1 for interactions gained and losted in graph1, respectively)" as edge attribute. The Differential graph has "pagerank" and "name" as vertex attributes.

2.2 Multiplex PageRank

We applied multiplex_page_rank() to calculate multiplex PageRank following definition by [2].

> set.seed(1)
> graph1 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph1)$name <- 1:100
> igraph::V(graph1)$pagerank <- igraph::page_rank(graph1)$vector
> #the base graph with pagerank and name as vertex attributes.
> set.seed(2)
> graph2 <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph2)$name <- 1:100
> igraph::V(graph2)$pagerank <- igraph::page_rank(graph2)$vector
> #the supplemental graph with pagerank and name as vertex attributes.
> multiplex_page_rank(graph1, graph2)

1 2 3 4 5 6
0.024486930 0.003587882 0.003269234 0.025062625 0.002517812 0.014031152
7 8 9 10 11 12
0.019560780 0.002517812 0.010657975 0.024750578 0.003587882 0.002517812
13 14 15 16 17 18
0.002517812 0.002517812 0.012543315 0.011993811 0.011752012 0.002517812
19 20 21 22 23 24
0.002517812 0.005019851 0.005073934 0.019579420 0.010917862 0.006654581
25 26 27 28 29 30
Multiplex PageRank values corresponded to nodes in graph1 (base network) will be outputed.

2.3 Adjusting PageRank Calculations

The `clean_graph()` can remove nodes by residing subgraph sizes, vertex names and PageRank values. We thus can adjust graphs for PageRank calculation.

```r
> set.seed(1)
> graph <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph)$name <- 1:100
> igraph::V(graph)$pagerank <- igraph::page_rank(graph)$vector
> #the graph to be cleaned, with pagerank and name as vertex attributes.
> clean_graph(graph, size=5)

IGRAPH 7077b87 DN-- 82 96 -- Erdos-Renyi (gnp) graph
+ attr: name (g/c), type (g/c), loops (g/l), p (g/n), name (v/n),
  | pagerank (v/n)
```
+ edges from 7077b87 (vertex names):
[1]  72->  1  88->  3  22->  4  11->  6  65->  6  87->  6  60->  7  85->  7
[9]  84->  9  33-> 10  100->10  11->100  2-> 15  40-> 15  3->  6  34->  6
[17]  19-> 17  46-> 17  5-> 20  69-> 20  100->20  92-> 21  27-> 22  83-> 22
[41]  21-> 41  71-> 41  49-> 42  65-> 42  77-> 42  87-> 43  100->43  52-> 44
+ ... omitted several edges

Adjusted graph will be outputed, with "pagerank" and "name" as vertex attributes.
The adjust_graph() can re-calculate PageRank with updated damping factor, personalized vector and edge weights.

> set.seed(1)
> graph <- igraph::erdos.renyi.game(100, 0.01, directed = TRUE)
> igraph::V(graph)$name <- 1:100
> igraph::V(graph)$pagerank <- igraph::page_rank(graph, damping=0.85)$vector
> #the graph to be adjusted, with pagerank and name as vertex attributes.
> adjust_graph(graph, damping=0.1)

IGRAPH a808b6f DN-- 100 98 -- Erdos-Renyi (gnp) graph
+ attr: name (g/c), type (g/c), loops (g/l), p (g/n), name (v/n), pagerank (v/n)
+ edges from a808b6f (vertex names):
  [1]  72->  1  88->  3  22->  4  11->  6  65->  6  87->  6  60->  7  85->  7
  [9]  84->  9  33-> 10  100->10  11->100  2-> 15  40-> 15  3->  6  34->  6
  [17]  19-> 17  46-> 17  5-> 20  69-> 20  100->20  92-> 21  27-> 22  83-> 22
  [41]  21-> 41  71-> 41  49-> 42  65-> 42  77-> 42  87-> 43  100->43  52-> 44
+ ... omitted several edges

Adjusted graph will be outputed, with updated "pagerank" and "name" as vertex attributes.
Please note diff_graph(), multiplex_page_rank(), clean_graph() and adjust_graph() can be used in combination for customized PageRank analysis tasks.

3 Prioritizing TFs in GRNs

3.1 Generating GRNs from Multi-Omics Profiles

The aracne_network() can re-format ARACNe network in regulon object for PageRank analysis. It can also handle GRNs reverse engineered using other algorithms, as long as such
GRNs are written in regulon object.

```r
> library(bcellViper)
> data(bcellViper)
> head(aracne_network(regulon[1:10]))

          reg  target direction
1       AATF    SAMM50       1
2       AATF     DRG1       1
3       AATF      ATIC       1
4       AATF   SMARCC1       1
5       AATF       AHCY       1
6       AATF  HSD17B10       1
```

The `accessibility_network()` can build network from accessibility, e.g. ATAC-Seq peaks.

```r
> table <- data.frame(Chr=c("chr1", "chr1"), Start=c(713689, 856337), End=c(714685, 862152), + row.names=c("A", "B"), stringsAsFactors=FALSE)
> regulators=c("FOXF2", "MZF1")
> # peaks and regulators to be analyzed
> library(GenomicRanges)
> library(GenomicFeatures)
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> library(annotate)
> promoter <- promoters(genes(TxDb.Hsapiens.UCSC.hg19.knownGene))
> names(promoter) <- getSYMBOL(names(promoter), data="org.Hs.eg")
> promoter <- promoter[!is.na(names(promoter))]
> # get promoter regions
> library(JASPAR2018)
> library(TFBSTools)
> library(motifmatchr)
> pfm <- getMatrixSet(JASPAR2018, list(species="Homo sapiens"))
> pfm <- pfm[unlist(lapply(pfm, function(x) name(x))) %in% regulators]
> # get regulator position frequency matrix (PFM) list
> library(BSgenome.Hsapiens.UCSC.hg19)
> accessibility_network(table, promoter, pfm, "BSgenome.Hsapiens.UCSC.hg19")
```

```r
  target   reg
1 LOC100288069 FOXF2
```
The `conformation_network()` can build network from conformation, e.g. HiChIP records.

```r
> table <- data.frame(Chr1=c("chr1", "chr1"), Position1=c(569265, 713603),
+                      Chr2=c("chr4", "chr1"), Position2=c(206628, 715110),
+                      row.names=c("A", "B"), stringsAsFactors=FALSE)
> regulators=c("FOXF2", "MZF1")
> #peaks and regulators to be analyzed
>
> promoter <- promoters(genes(TxDb.Hsapiens.UCSC.hg19.knownGene))
> names(promoter) <- getSYMBOL(names(promoter), data="org.Hs.eg")
> promoter <- promoter[!is.na(names(promoter))]
> #get promoter regions
>
> pfm <- getMatrixSet(JASPAR2018, list(species="Homo sapiens"))
> pfm <- pfm[unlist(lapply(pfm, function(x) name(x))) %in% regulators]
> #get regulator position frequency matrix (PFM) list
>
> conformation_network(table, promoter, pfm, "BSgenome.Hsapiens.UCSC.hg19")

<table>
<thead>
<tr>
<th>target</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZNF876P</td>
<td>MZF1</td>
</tr>
<tr>
<td>LOC100288069</td>
<td>FOXF2</td>
</tr>
<tr>
<td>LOC100288069</td>
<td>MZF1</td>
</tr>
</tbody>
</table>
```

3.2 Filter GRNs with Expression Profiles

The `P_graph()` can filter GRNs by quantifying joint and margin probability distributions of regulator-target pairs. Statistically significant non-random regulator-target pairs will be kept.

```r
> dset <- exprs(dset)
> net <- do.call(rbind, lapply(1:10, function(i, regulon){
+    data.frame(reg=rep(names(regulon)[i], 10),
+              target=names(regulon[[i]][[1]][1:10]),
+              stringsAsFactors = FALSE), regulon=regulon))
> P_graph(dset, net, method="difference", null=NULL, threshold=0.05)
```
3.3 Session Information

> sessionInfo()

R version 4.4.0 beta (2024-04-15 r86425)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.4 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
[1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8 LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8 LC_NAME=C
[9] LC_ADDRESS=C LC_TELEPHONE=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats4  stats  graphics  grDevices  utils  datasets  methods
[8] base
other attached packages:
[1] BSgenome.Hsapiens.UCSC.hg19_1.4.3
[2] BSgenome_1.72.0
[3] rtracklayer_1.64.0
[5] Biostrings_2.72.0
[6] XVector_0.44.0
[7] motifmatchr_1.26.0
[8] TFBSTools_1.42.0
[9] JASPAR2018_1.1.1
[10] annotate_1.82.0
[12] org.Hs.eg.db_3.19.1
[14] GenomicFeatures_1.56.0
[15] AnnotationDbi_1.66.0
[16] GenomicRanges_1.56.0
[17] GenomeInfoDb_1.40.0
[18] IRanges_2.38.0
[19] S4Vectors_0.42.0
[20] bcellViper_1.39.0
[21] Biobase_2.64.0
[22] BiocGenerics_0.50.0
[23] pageRank_1.14.0

loaded via a namespace (and not attached):
[1] DBI_1.2.2               bitops_1.0-7
[3] rlang_1.1.3             magrittr_2.0.3
[5] matrixStats_1.3.0       compiler_4.4.0
[7] RSQLite_2.3.6           png_0.1-8
[9] vctrs_0.6.5             reshape2_1.4.4
[11] stringr_1.5.1          pwalignment_1.0.0
[13] pkgconfig_2.0.3         crayon_1.5.2
[15] fastmap_1.1.1           caTools_1.18.2
[17] utf8_1.2.4              Rsamtools_2.20.0
[19] tzdb_0.4.0              pracma_2.4.4
[21] UCSC.utils_1.0.0        DirichletMultinomial_1.46.0
[23] bit_4.0.5               zlibbioc_1.50.0
[25] cachem_1.0.8            CNEr_1.40.0
[27] jsonlite_1.8.8          blob_1.2.4
[29] DelayedArray_0.30.0    BiocParallel_1.38.0
[31] parallel_4.4.0         R6_2.5.1
[33] stringi_1.8.3  Rcpp_1.0.12
[35] SummarizedExperiment_1.34.0 R.utils_2.12.3
[37] readr_2.1.5  Matrix_1.7-0
[39] igraph_2.0.3  tidyselect_1.2.1
[41] abind_1.4-5  yaml_2.3.8
[43] codetools_0.2-20 curl_5.2.1
[45] lattice_0.22-6 tibble_3.2.1
[47] plyr_1.8.9  KEGGREST_1.44.0
[49] pillar_1.9.0  MatrixGenerics_1.16.0
[51] generics_0.1.3  RCurl_1.98-1.14
[53] hms_1.1.3  ggplot2_3.5.1
[55] munsell_0.5.1  scales_1.3.0
[57] gtools_3.9.5  xtable_1.8-4
[59] glue_1.7.0  seqLogo_1.70.0
[61] tools_4.4.0  TFMPvalue_0.0.9
[63] GenomicAlignments_1.40.0  poweRlaw_0.80.0
[65] grid_4.4.0  colorspace_2.1-0
[67] GenomeInfoDbData_1.2.12  restfulr_0.0.15
[69] cli_3.6.2  fansi_1.0.6
[71] S4Arrays_1.4.0  dplyr_1.1.4
[73] gtable_0.3.5  R.methodsS3_1.8.2
[75] SparseArray_1.4.0  rjson_0.2.21
[77] memoise_2.0.1  R.oo_1.26.0
[79] lifecycle_1.0.4  http_1.4.7
[81] GO.db_3.19.1  bit64_4.0.5
4 References