

Introduction to the pageRank Package

Hongxu Ding

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

October 27, 2020

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	6
3.1	Generating GRNs from Multi-Omics Profiles	6
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 base network, with regular PageRank of other supplemental networks as edge weights and personalization vector.

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.knownGene*, *org.Hs.eg.db*, *annotate*, *GenomicFeatures*, *JASPAR2018*, *TFBSTools* and *bcellViper*, to run the examples. After installing R, all required components can be obtained with:

```
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.

```
> 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 c83add8 DN-- 98 190 --
+ attr: name (v/c), pagerank (v/n), moi (e/n)
+ edges from c83add8 (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 5 ->91 6 ->25 6 ->50 7 ->37 7 ->67 7 ->73 7 ->85
 [19] 8 ->80 9 ->90 10->26 11->6 11->100 12->70 12->82 12->92 13->30
 [28] 13->48 13->51 13->61 15->74 15->77 16->85 17->31 17->32 17->34
 [37] 17->50 17->58 19->17 19->96 20->23 20->79 20->87 21->41 21->45
 [46] 22->4 22->41 23->57 24->61 25->66 26->34 26->39 26->72 27->22
 [55] 27->43 28->98 29->95 30->84 32->49 33->10 34->16 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 losed 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
0.022431793	0.004376548	0.005101288	0.021806420	0.004038261	0.014009056
7	8	9	10	11	12
0.017033776	0.004038261	0.010897818	0.022024129	0.004376548	0.004038261
13	14	15	16	17	18

0.004038261	0.004038261	0.011984174	0.010662594	0.012861595	0.004038261
19	20	21	22	23	24
0.004038261	0.006973192	0.007883792	0.018708510	0.009449760	0.008580800
25	26	27	28	29	30
0.009992110	0.023175991	0.016861132	0.007081820	0.004038261	0.007696307
31	32	33	34	35	36
0.017884438	0.004465826	0.019332930	0.005497036	0.026386118	0.014886821
37	38	39	40	41	42
0.004473382	0.012774746	0.008259414	0.008244753	0.010494265	0.015930714
43	44	45	46	47	48
0.017448552	0.007901711	0.008117970	0.007680122	0.004038261	0.004263786
49	50	51	52	53	54
0.011679744	0.020924465	0.008291128	0.009090471	0.008316500	0.008569083
55	56	57	58	59	60
0.004038261	0.011492897	0.013013291	0.006154228	0.009112068	0.005053121
61	62	63	64	65	66
0.011557465	0.008125871	0.004038261	0.009595523	0.004038261	0.013377737
67	68	69	70	71	72
0.004473382	0.005786216	0.006188052	0.011348562	0.004807863	0.021639449
73	74	75	76	77	78
0.005681911	0.011938368	0.004664091	0.028329952	0.005204011	0.008601557
79	80	81	82	83	84
0.007514945	0.004714834	0.037833945	0.005251514	0.004376548	0.007672082
85	86	87	88	89	90
0.017889151	0.004376548	0.013828925	0.005346400	0.004038261	0.010681597
91	92	93	94	95	96
0.011139970	0.009048308	0.015566866	0.009116546	0.006681434	0.004714834
97	98	99	100		
0.012194456	0.008225016	0.005368584	0.004931364		

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.

```
> 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 c83fa24 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 c83fa24 (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-> 16 34-> 16
[17] 19-> 17 46-> 17 5-> 20 69-> 20 100-> 20 92-> 21 27-> 22 83-> 23
[25] 42-> 24 6-> 25 10-> 26 74-> 27 94-> 27 43-> 31 36-> 33 38-> 35
[33] 59-> 35 90-> 35 60-> 36 70-> 36 53-> 38 26-> 39 46-> 40 88-> 40
[41] 21-> 41 71-> 41 49-> 42 65-> 42 77-> 42 87-> 43 100-> 43 52-> 44
[49] 21-> 45 54-> 46 32-> 49 92-> 49 6-> 50 17-> 50 43-> 52 54-> 52
+ ... omitted several edges
```

Adjusted graph will be outputted, 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 c846c94 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 c846c94 (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-> 16 34-> 16
[17] 19-> 17 46-> 17 5-> 20 69-> 20 100-> 20 92-> 21 27-> 22 83-> 23
[25] 42-> 24 6-> 25 10-> 26 74-> 27 94-> 27 63-> 30 43-> 31 36-> 33
[33] 38-> 35 59-> 35 90-> 35 60-> 36 70-> 36 53-> 38 26-> 39 46-> 40
[41] 88-> 40 21-> 41 71-> 41 49-> 42 65-> 42 77-> 42 87-> 43 100-> 43
[49] 52-> 44 21-> 45 54-> 46 32-> 49 92-> 49 6-> 50 17-> 50 13-> 51
+ ... omitted several edges
```

Adjusted graph will be outputted, 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.

```
> 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.

```
> table <- data.frame(Chr=c("chr1", "chr1"), Start=c(713689, 856337), End=c(714685, 856337),
+                      row.names=c("A", "B"), stringsAsFactors=FALSE)
> regulators=c("FOXF2", "MZFI")
> #peaks and regulators to be analyzed
>
> library(GenomicRanges)
> library(GenomicFeatures)
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> library(org.Hs.eg.db)
> 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")
```

	target	reg
1	LOC100288069	FOXF2
2	LOC100288069	MZF1
3	LINC02593	FOXF2
4	SAMD11	FOXF2
5	LINC02593	MZF1
6	SAMD11	MZF1

The `conformation_network()` can build network from conformation, e.g. HiChIP records.

```
> table <- data.frame(Chr1=c("chr1", "chr1"), Position1=c(569265, 713603), Strand1=c(
+                               Chr2=c("chr4", "chr1"), Position2=c(206628, 715110), Strand2=c(
+                               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")
```

	target	reg
1	ZNF876P	MZF1
2	LOC100288069	FOXF2
3	LOC100288069	MZF1

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.

```

> 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)

```

		0%
=		1%
=		2%
==		3%
===		4%
===		5%
====		6%
=====		7%
=====		8%
=====		9%
=====		10%
=====		11%
=====		12%
=====		13%
=====		14%
=====		15%
=====		16%
=====		17%

=====			18%
=====			19%
=====			20%
=====			21%
=====			21%
=====			22%
=====			23%
=====			24%
=====			25%
=====			26%
=====			27%
=====			28%
=====			29%
=====			30%
=====			31%
=====			32%
=====			33%
=====			34%
=====			35%
=====			36%
=====			36%

	=====		37%
	=====		38%
	=====		39%
	=====		40%
	=====		41%
	=====		42%
	=====		43%
	=====		44%
	=====		45%
	=====		46%
	=====		47%
	=====		48%
	=====		49%
	=====		50%
	=====		51%
	=====		52%
	=====		53%
	=====		54%
	=====		55%
	=====		56%
	=====		57%

			58%
	=====		
			59%
	=====		
			60%
	=====		
			61%
	=====		
			62%
	=====		
			63%
	=====		
			64%
	=====		
			64%
	=====		
			65%
	=====		
			66%
	=====		
			67%
	=====		
			68%
	=====		
			69%
	=====		
			70%
	=====		
			71%
	=====		
			72%
	=====		
			73%
	=====		
			74%
	=====		
			75%
	=====		
			76%
	=====		
			77%

	=====	78%
	=====	79%
	=====	79%
	=====	80%
	=====	81%
	=====	82%
	=====	83%
	=====	84%
	=====	85%
	=====	86%
	=====	87%
	=====	88%
	=====	89%
	=====	90%
	=====	91%
	=====	92%
	=====	93%
	=====	94%
	=====	95%
	=====	96%
	=====	97%

=====		98%
=====		99%
=====		100%
		0%
=		1%
=		2%
==		3%
===		4%
====		5%
=====		6%
=====		7%
=====		8%
=====		9%
=====		10%
=====		11%
=====		12%
=====		13%
=====		14%
=====		15%
=====		16%
=====		17%

			18%
	=====		19%
	=====		20%
	=====		21%
	=====		22%
	=====		23%
	=====		24%
	=====		25%
	=====		26%
	=====		27%
	=====		28%
	=====		29%
	=====		30%
	=====		31%
	=====		32%
	=====		33%
	=====		34%
	=====		35%
	=====		36%
	=====		37%
	=====		38%

	=====		39%
	=====		40%
	=====		41%
	=====		42%
	=====		43%
	=====		44%
	=====		45%
	=====		46%
	=====		47%
	=====		48%
	=====		49%
	=====		51%
	=====		52%
	=====		53%
	=====		54%
	=====		55%
	=====		56%
	=====		57%
	=====		58%
	=====		59%
	=====		60%

		61%
=====		62%
		63%
=====		64%
		65%
=====		66%
		67%
=====		68%
		69%
=====		70%
		71%
=====		72%
		73%
=====		74%
		75%
=====		76%
		77%
=====		78%
		79%
=====		80%
		81%
=====		

=====		82%
=====		83%
=====		84%
=====		85%
=====		86%
=====		87%
=====		88%
=====		89%
=====		90%
=====		91%
=====		92%
=====		93%
=====		94%
=====		95%
=====		96%
=====		97%
=====		98%
=====		99%
=====		100%
		0%
=		1%

=		2%
==		3%
===		4%
====		5%
=====		6%
=====		7%
=====		8%
=====		9%
=====		10%
=====		11%
=====		12%
=====		13%
=====		14%
=====		15%
=====		16%
=====		17%
=====		18%
=====		19%
=====		20%
=====		21%
=====		22%

			23%
	=====		
			24%
	=====		
			25%
	=====		
			26%
	=====		
			27%
	=====		
			28%
	=====		
			29%
	=====		
			30%
	=====		
			31%
	=====		
			32%
	=====		
			33%
	=====		
			34%
	=====		
			35%
	=====		
			36%
	=====		
			37%
	=====		
			38%
	=====		
			39%
	=====		
			40%
	=====		
			41%
	=====		
			42%
	=====		
			43%

	=====		44%
	=====		45%
	=====		46%
	=====		47%
	=====		48%
	=====		49%
	=====		51%
	=====		52%
	=====		53%
	=====		54%
	=====		55%
	=====		56%
	=====		57%
	=====		58%
	=====		59%
	=====		60%
	=====		61%
	=====		62%
	=====		63%
	=====		64%
	=====		65%

			66%
	=====		
			67%
	=====		
			68%
	=====		
			69%
	=====		
			70%
	=====		
			71%
	=====		
			72%
	=====		
			73%
	=====		
			74%
	=====		
			75%
	=====		
			76%
	=====		
			77%
	=====		
			78%
	=====		
			79%
	=====		
			80%
	=====		
			81%
	=====		
			82%
	=====		
			83%
	=====		
			84%
	=====		
			85%
	=====		
			86%

=====		87%
=====		88%
=====		89%
=====		90%
=====		91%
=====		92%
=====		93%
=====		94%
=====		95%
=====		96%
=====		97%
=====		98%
=====		99%
=====		100%
		0%
=		1%
=		2%
==		3%
===		4%
====		5%
====		6%

=====		7%
=====		8%
=====		9%
=====		10%
=====		11%
=====		12%
=====		13%
=====		14%
=====		15%
=====		16%
=====		17%
=====		18%
=====		19%
=====		20%
=====		21%
=====		22%
=====		23%
=====		24%
=====		25%
=====		26%
=====		27%

	=====		28%
	=====		29%
	=====		30%
	=====		31%
	=====		32%
	=====		33%
	=====		34%
	=====		35%
	=====		36%
	=====		37%
	=====		38%
	=====		39%
	=====		40%
	=====		41%
	=====		42%
	=====		43%
	=====		44%
	=====		45%
	=====		46%
	=====		47%
	=====		48%

	=====		49%
	=====		51%
	=====		52%
	=====		53%
	=====		54%
	=====		55%
	=====		56%
	=====		57%
	=====		58%
	=====		59%
	=====		60%
	=====		61%
	=====		62%
	=====		63%
	=====		64%
	=====		65%
	=====		66%
	=====		67%
	=====		68%
	=====		69%
	=====		70%

			71%
	=====		
	=====		72%
	=====		73%
	=====		74%
	=====		75%
	=====		76%
	=====		77%
	=====		78%
	=====		79%
	=====		80%
	=====		81%
	=====		82%
	=====		83%
	=====		84%
	=====		85%
	=====		86%
	=====		87%
	=====		88%
	=====		89%
	=====		90%
	=====		91%

```

|
|=====| 92%
|
|=====| 93%
|
|=====| 94%
|
|=====| 95%
|
|=====| 96%
|
|=====| 97%
|
|=====| 98%
|
|=====| 99%
|
|=====| 100%IGRAPH e
+ attr: name (v/c), pagerank (v/n), pvalue (e/n)
+ edges from ec6bdbd (vertex names):
[1] PPM1G ->AATF    CTBP2 ->APP    TAGLN ->APP    MTSS1 ->APP    JMJD1C->ARID4A

```

3.3 Session Information

```

> sessionInfo()

R version 4.0.3 (2020-10-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 R2 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats4    parallel  stats      graphics  grDevices  utils      datasets
[8] methods  base

```

other attached packages:

- [1] BSgenome.Hsapiens.UCSC.hg19_1.4.3
- [2] BSgenome_1.58.0
- [3] rtracklayer_1.50.0
- [4] Biostrings_2.58.0
- [5] XVector_0.30.0
- [6] motifmatchr_1.12.0
- [7] TFBSTools_1.28.0
- [8] JASPAR2018_1.1.1
- [9] annotate_1.68.0
- [10] XML_3.99-0.5
- [11] org.Hs.eg.db_3.12.0
- [12] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
- [13] GenomicFeatures_1.42.0
- [14] AnnotationDbi_1.52.0
- [15] GenomicRanges_1.42.0
- [16] GenomeInfoDb_1.26.0
- [17] IRanges_2.24.0
- [18] S4Vectors_0.28.0
- [19] bcellViper_1.25.0
- [20] Biobase_2.50.0
- [21] BiocGenerics_0.36.0
- [22] pageRank_1.0.0

loaded via a namespace (and not attached):

[1] bitops_1.0-6	matrixStats_0.57.0
[3] DirichletMultinomial_1.32.0	bit64_4.0.5
[5] progress_1.2.2	httr_1.4.2
[7] tools_4.0.3	R6_2.4.1
[9] seqLogo_1.56.0	DBI_1.1.0
[11] colorspace_1.4-1	tidyselect_1.1.0
[13] prettyunits_1.1.1	curl_4.3
[15] bit_4.0.4	compiler_4.0.3
[17] xml2_1.3.2	DelayedArray_0.16.0
[19] caTools_1.18.0	scales_1.1.1
[21] readr_1.4.0	askpass_1.1
[23] rappdirs_0.3.1	stringr_1.4.0
[25] digest_0.6.27	Rsamtools_2.6.0
[27] R.utils_2.10.1	pkgconfig_2.0.3
[29] MatrixGenerics_1.2.0	dbplyr_1.4.4
[31] rlang_0.4.8	RSQLite_2.2.1
[33] generics_0.0.2	BiocParallel_1.24.0

[35] gtools_3.8.2	dplyr_1.0.2
[37] R.oo_1.24.0	RCurl_1.98-1.2
[39] magrittr_1.5	GO.db_3.12.0
[41] GenomeInfoDbData_1.2.4	Matrix_1.2-18
[43] Rcpp_1.0.5	munsell_0.5.0
[45] lifecycle_0.2.0	R.methodsS3_1.8.1
[47] stringi_1.5.3	SummarizedExperiment_1.20.0
[49] zlibbioc_1.36.0	plyr_1.8.6
[51] BiocFileCache_1.14.0	grid_4.0.3
[53] blob_1.2.1	crayon_1.3.4
[55] CNEr_1.26.0	lattice_0.20-41
[57] hms_0.5.3	KEGGREST_1.30.0
[59] pillar_1.4.6	igraph_1.2.6
[61] reshape2_1.4.4	biomaRt_2.46.0
[63] TFMPvalue_0.0.8	glue_1.4.2
[65] vctrs_0.3.4	png_0.1-7
[67] gtable_0.3.0	powerLaw_0.70.6
[69] openssl_1.4.3	purrr_0.3.4
[71] assertthat_0.2.1	ggplot2_3.3.2
[73] xtable_1.8-4	pracma_2.2.9
[75] tibble_3.0.4	GenomicAlignments_1.26.0
[77] memoise_1.1.0	ellipsis_0.3.1

4 References

1. Rozenstein, Polina, and Aristides Gionis. "Temporal pagerank." Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2016.
2. Halu, Arda, et al. "Multiplex pagerank." PloS one 8.10 (2013).
3. Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." (1998).
4. Page, Lawrence, et al. The pagerank citation ranking: Bringing order to the web. Stanford InfoLab, 1999.