Supplementary Information

RTNduals case studies: exploring dual regulons in breast cancer regulatory networks.

Vinicius S. Chagas[1,*], Clarice S. Groeneveld[1,*], Kelin G. de Oliveira[1,2], Sheyla Trefflich[3], Rodrigo C. de Almeida[4], Bruce A. J. Ponder[5], Kerstin B. Meyer[5,6], Steven J. M. Jones[7], A. Gordon Robertson[7,**], Mauro A. A. Castro[1,**].

[1]Bioinformatics and Systems Biology Lab, Federal University of Paraná, Curitiba, 81520-260, Brazil. [2]Department of Clinical Sciences, Lund University, Lund, 221 85, Sweden. [3]Bioinformatics Department, Federal University of Minas Gerais, Belo Horizonte, 31270-901, Brazil. [4]Department of Medical Statistics and Bioinformatics, Section Molecular Epidemiology, Leiden University Medical Center, Leiden, The Netherlands. [5]Department of Oncology and Cancer Research UK Cambridge Institute, University of Cambridge, Li Ka Shing Centre, Cambridge, United Kingdom. [6]Wellcome Sanger Institute, Hinxton, CB10 1SA, United Kingdom. [7]Canada's Michael Smith Genome Sciences Center, BC Cancer Agency, Vancouver, V5Z 4S6, Canada.

*These authors contributed equally to this work.
**Corresponding authors: mauro.castro@ufpr.br, grobertson@bcgsc.ca

Contents

1. METABRIC breast cancer cohort 1

1.1 Context

Fletcher *et al.* (2013) reconstructed regulons for 809 transcription factors (TFs) using microarray transcriptomic data from the METABRIC breast cancer cohort (Curtis *et al.*, 2012). Castro *et al.* (2016) found that 36 of these TF regulons were associated with genetic risk of breast cancer. The risk TFs were in two distinct clusters. The "cluster 1" risk TFs were associated with estrogen receptor-positive (ER+) breast cancer risk and comprise TFs such as ESR1, FOXA1, and GATA3, whereas the "cluster 2" risk TFs were associated with estrogen receptor-negative (ER-), basal-like breast cancer. **Our goals here are (1) to explore associations between the regulons reconstructed by Fletcher *et al.* (2013) and (2) to identify *dual regulons*.**

1.2 Package installation and data sets

The *RTNduals* package is available from the R/Bioconductor repository, together with other required packages. Installing and then loading the *Fletcher2013b* data package will make available all data required for this case study.

```
#-- Set the Bioconductor repository
#-- Please make sure to use bioc version >= 3.8 (R >= 3.5)
source("https://bioconductor.org/biocLite.R")
biocVersion()
```

```
#-- Install RTNduals and other required packages
#-- RTN(>=2.6.3); RTNduals(>=1.6.2); Fletcher2013b(>=1.16.0)
biocLite(c("RTNduals","Fletcher2013b"))
install.packages("pheatmap")
```

```
#-- Call packages
library(RTNduals)
library(Fletcher2013b)
```

```
#-- Load 'rtni1st' data object, which includes regulons and expression profiles
data("rtni1st")
```

```
#-- A list of transcription factors of interest (here 36 risk-associated TFs)
risk.tfs <- c("AFF3", "AR", "ARNT2", "BRD8", "CBFB", "CEBPB", "E2F2", "E2F3", "ENO1",
              "ESR1", "FOSL1", "FOXA1", "GATA3", "GATAD2A", "LZTFL1", "MTA2", "MYB",
              "MZF1", "NFIB", "PPARD", "RARA", "RB1", "RUNX3", "SNAPC2", "SOX10",
              "SPDEF", "TBX19", "TCEAL1", "TRIM29", "XBP1", "YBX1", "YPEL3", "ZNF24",
              "ZNF434", "ZNF552", "ZNF587")
```

1.3 Preparing data for input to *RTNduals*

To prepare the input data for *RTNduals* we run two commands. The first filters an `TNI-class` object with the `tni.dpi.filter` function, and the second creates an `MBR-class` object with the `tni2mbrPreprocess` function. The `tni.dpi.filter` function will update the `TNI-class` object called `rtni1st`, for `eps = NA` (see below). The `tni2mbrPreprocess` function will then create the `MBR-class` object called `mbr1st`, which will require (1) a transcriptional regulatory network computed by the RTN package (the `mbr1st` loaded above), and (2) a list of regulators.

As we explain in **section 2.3**, the `eps` argument sets the ARACNe algorithm's mutual information (MI) threshold. When we want to remove all dependencies between regulons, *e.g.* to enrich regulons with direct

targets, we recommend setting `eps = 0.0`, which is the default option. The regulatory network that we've loaded would have been calculated with this setting. For *RTNduals*, however, we want to assess the overlap between pairs of target gene sets, so we need to re-run this step on the `rtni1st` object. We recommend setting `eps = NA`, which will estimate a nonzero MI threshold from the empirical null distribution computed in the permutation and bootstrap steps.

```
#-- Update the 'rtni1st' object
rtni1st <- tni.dpi.filter(rtni1st, eps = NA)
```

```
#-- Check consistency of the input data and build an MBR-class object
mbr1st <- tni2mbrPreprocess(tni = rtni1st, regulatoryElements = risk.tfs)
```

1.4 A single step infers *dual regulons*

The `mbrAssociation` function tests the association between pairs of regulons, using Fisher's exact test and permutation analysis to assess the statistical significance of the overlap and the correlation between regulons, respectively.

```
#-- Run 'mbrAssociation' pipeline
mbr1st <- mbrAssociation(mbr1st)
```

```
#-- Get a list of dual regulons ranked by correlation statistics
#-- (Supplementary Table 1)
mbr1st_results <- mbrGet(mbr1st, "dualsCorrelation")
mbr1st_results[1:10,]
```

Supplementary Table 1. Top 10 *dual regulons* ranked by P-value.

Dual Regulon	MI Regulators	R Regulons	Pvalue	Adjusted Pvalue*
ESR1~FOXA1	0.34	0.84	2.32e−117	1.46e−114
ESR1~GATA3	0.55	0.84	2.02e−112	1.27e−109
FOXA1~GATA3	0.43	0.81	5.95e−111	3.75e−108
AFF3~ESR1	0.32	0.72	2.06e−85	1.30e−82
MYB~FOXA1	0.30	0.73	7.47e−85	4.71e−82
RUNX3~FOXA1	0.20	−0.73	1.81e−82	1.14e−79
ESR1~MYB	0.38	0.73	8.10e−82	5.10e−79
XBP1~FOXA1	0.54	0.73	7.17e−81	4.52e−78
XBP1~ESR1	0.33	0.68	2.70e−80	1.70e−77
AFF3~GATA3	0.33	0.69	9.11e−78	5.74e−75

*Benjamini-Hochberg (BH) adjusted p-values

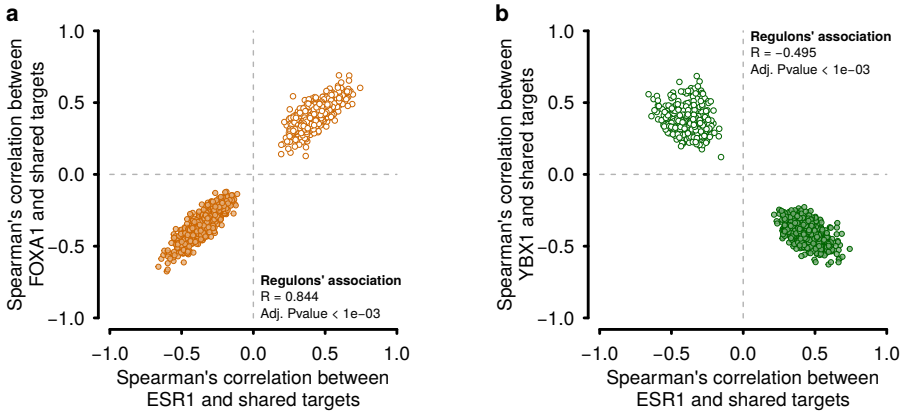
Supplementary Table 1 shows the top 10 dual regulons in the *mbr1st_results* object. All but one of them have positive Spearman correlation coefficients (R *Regulons* > 0), meaning that both TFs co-operate by influencing shared target genes in the same direction (*i.e.* either co-activating or co-repressing the shared targets, see below).

1.5 Representing *dual regulons* with scatter plots

The `mbrPlotDuals` function allows us to represent how the expression of the shared targets of a dual regulon is correlated with the expression of each regulator. In **Supplementary Figure 1** we show *ESR1~FOXA1* and *ESR1~YBX1* as examples generated by this function.

```
#-- Scatter plots of shared targets
#-- (Supplementary Figure 1)
```

```
mbrPlotDuals(mbr1st, "ESR1~FOXA1")
mbrPlotDuals(mbr1st, "ESR1~YBX1")
```



Supplementary Figure 1: Relationship between the expression of shared targets in a dual regulon, computed from the expression profiles of METABRIC breast cancer data, n=997 (Fletcher *et al.*, 2013). (**a**) Transcription factors ESR1 and FOXA1 co-operate in influencing shared target genes, while (**b**) ESR1 and YBX1 have opposite-signed correlations (see **Figure 1c,d** for ESR1∼GATA3 and ESR1∼NFIB dual regulons, respectively).

1.6 A heatmap of correlations for regulon pairs

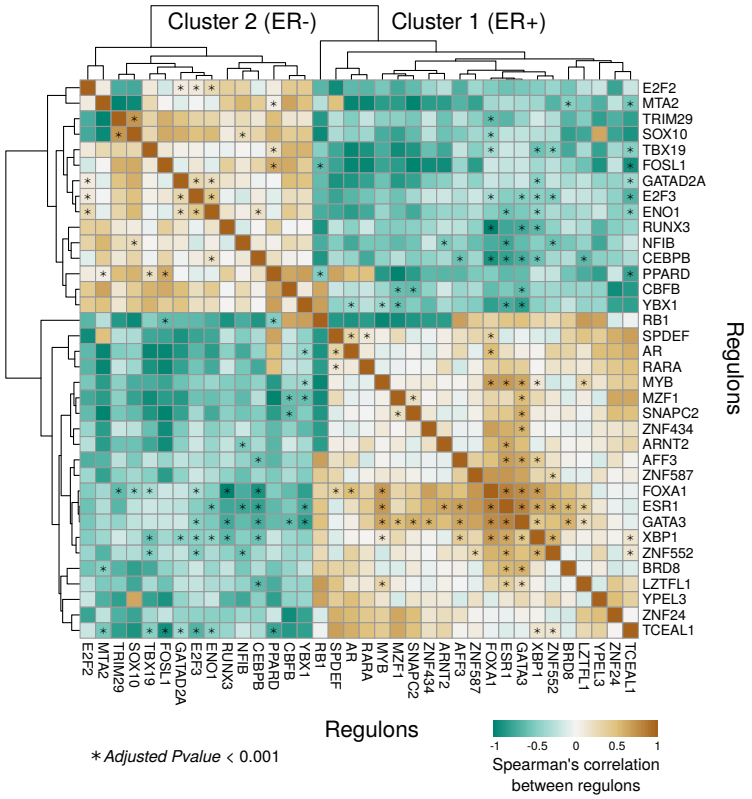
We can now visualize a heatmap to summarize the relationships between all regulon pairs assessed in the analysis pipeline. For this, we need to call the `mbrGet` function to obtain a correlation matrix with all Spearman correlation coefficients, and then plot a `heatmap`.

```
#-- Get the correlation matrix between regulons
dmat <- mbrGet(mbr1st, what="dualsCorMatrix")
```

```
#-- Plot the correlation matrix between regulons
#-- (Supplementary Figure 2)
library(pheatmap)
colorpal <- c("#018571","#80CDC1","#F5F5F5","#DFC27D","#A6611A")
pheatmap(mat = dmat$cormat, display_numbers = dmat$sigmat,
         color = colorRampPalette(colorpal)(100),
         clustering_distance_rows = "correlation",
         clustering_distance_cols = "correlation")
```

Each square in the heatmap of the **Supplementary Figure 2** represents a Spearman correlation coefficient estimated for a regulon pair, and summarizes the more detailed information shown in the scatter plots of **Supplementary Figure 1** for individual dual regulons. Since the heatmap represents a correlation matrix, values are mirrored across the diagonal. All significant associations ($P < 0.001$, BH adjusted) are marked with asterisks.

The lower right corner, in "Cluster 1", contains a region with highly correlated regulons that is enriched with significant predictions, particularly among GATA3, ESR1 and FOXA1, all of which are highly influential in ER+ breast cancer (Theodorou *et al.*, 2013). Duals *ESR1~YBX1* (**Supplementary Figure 1b**) and *ESR1~NFIB* (**Figure 1d**) are also of note: both NFIB and YBX1 interact with the ESR1-FOXA1 complex and inhibit the transactivational potential of ESR1, and these interactions further repress ESR1 target gene expression when in association with induced FGFR2 signalling (Campbell *et al.*, 2018). There is also evidence linking SOX10 and TRIM29 (Panaccione *et al.*, 2017) (as indicated in **Supplementary Figure 2**), both of which are associated with a neural stem cell-like signature in ER- tumours.



Supplementary Figure 2: Heatmap showing the correlation matrix between regulons for 36 transcription factors. Each point in the heatmap summarizes the relationship between a regulon's shared targets as shown in the scatter plots of **Supplementary Figure 1**. Significant associations ($P < 0.001$, BH adjusted) are indicated with asterisks. "Cluster 1" and "Cluster 2", as named in Castro et al. (2016), represent regulons associated with ER+ and ER- tumours, respectively.

1.8 Other tools for inferring co-regulation

Several algorithms have been developed that support exploring regulation in biological networks. *Bionet* (Beisser *et al.*, 2010), *Minet* (Meyer *et al.*, 2008), and *CoRegNet* (Nicolle *et al.*, 2015) are examples of R/Bioconductor packages that implement routines and hypothesis testing methods for functional analysis of biological networks. Nicolle *et al.* (2015) compiled features of a large number of tools to compare state of the art methods and concluded that, at that time, only *CoRegNet* provided methods to infer co-regulation. The same authors assessed *RTN* as able to do "network inference", "genomic data integration", and "differential analysis", but not "co-regulation". Now, given *RTNduals*, the *RTN* toolset addresses co-regulation. Next we run the *CoRegNet* (version 1.20.0) with the same gene expression data and 36 risk TFs that we used in *RTNduals* (version 1.6.2). *RTNduals* returned 86 dual regulons; *CoRegNet* returns 76 co-regulator pairs.

```
#-- Run CoRegNet with the same input data used in RTNduals
library(CoRegNet)
gexp1st <- tni.get(rtni1st, what="gexp", idkey = "SYMBOL")
coreg1st <- hLICORN(gexp1st, TFlist=risk.tfs)
```

```
#-- Get co-regulators ranked by CoRegNet 'support' statistics
#-- (Supplementary Table 2)
coreg1st_results <- coregulators(coreg1st, adjustMethod = "BH")
coreg1st_results[1:10,]
```

Supplementary Table 2. Top 10 co-regulators reported by *CoRegNet*.

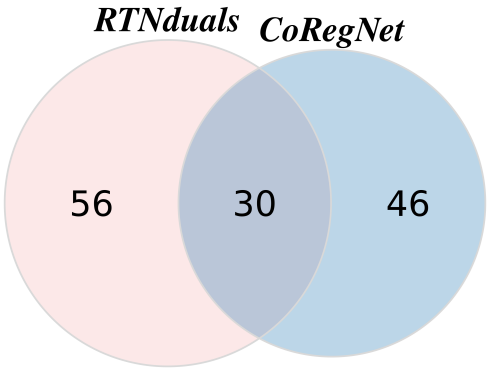
Co-regulator	Reg1	Reg2	Support	Fisher Test	Adjusted Pvalue*
ESR1~GATA3**	ESR1	GATA3	0.053	0.00e+00	0.00e+00
ESR1~AFF3**	ESR1	AFF3	0.041	0.00e+00	0.00e+00
AFF3~GATA3**	AFF3	GATA3	0.037	0.00e+00	0.00e+00
ESR1~FOXA1**	ESR1	FOXA1	0.032	6.87e-124	6.38e-123
FOXA1~GATA3**	FOXA1	GATA3	0.029	1.42e-41	5.61e-41
ESR1~MYB**	ESR1	MYB	0.024	2.38e-86	1.50e-85
FOXA1~AFF3	FOXA1	AFF3	0.019	2.57e-46	1.10e-45
MYB~GATA3**	MYB	GATA3	0.018	8.85e-11	2.59e-10
ESR1~XBP1**	ESR1	XBP1	0.016	8.36e-257	1.47e-255
AFF3~XBP1**	AFF3	XBP1	0.014	7.76e-266	1.53e-264

*Benjamini-Hochberg (BH) adjusted p-values
**Co-regulators also listed as dual regulons in Supplementary Figure 2

Supplementary Table 2 shows that the top 10 co-regulators reported by *CoRegNet* are consistent with the regulons identified by *RTNduals*, with nine of them listed in **Supplementary Figure 2**. Next we use a Venn diagram to extend the comparison with *RTNduals* to all co-regulators.

```
#-- Align labels between CoRegNet and RTNduals
lab <- paste(coreg1st_results$Reg1, coreg1st_results$Reg2, sep="~")
idx <- lab %in% rownames(mbr1st_results)
lab[!idx] <- paste(coreg1st_results$Reg2, coreg1st_results$Reg1, sep="~")[!idx]
rownames(coreg1st_results) <- lab

#-- Plot a Venn diagram (Supplementary Figure 3)
library(VennDiagram)
res <- list(RTNduals = rownames(mbr1st_results),
            CoRegNet = rownames(coreg1st_results))
venn.diagram(res, fill=c("#e41a1c","#1f78b4"), col="grey85",
             alpha=c(0.1,0.3), cex=2.5, cat.cex=2.5, cat.pos=0,
             cat.fontface=4, lty=1, fontfamily=3, filename="venn.tiff")
```



Supplementary Figure 3: Venn diagram showing the overlap between dual regulons and co-regulators (inferred by *RTNduals* and *CoRegNet*, respectively) for the same input gene expression data.

Supplementary Figure 3 shows that 30 (39%) of 76 of the CoRegNet co-regulators were also identified as dual regulons. Differences in reported co-regulation may be due to differences in the input regulatory networks. These packages use different algorithms to compute the regulatory networks; *CoRegNet* detects TF-gene interactions with the LICORN algorithm (Rouveirol *et al.*, 2007), while *RTN* reconstructs regulons with the ARACNe algorithm (Margolin *et al.*, 2006) (additional comments in **section 2.3**). After computing the regulatory network, both packages use similar approaches to access the number of targets shared between a pair of regulators. The main conceptual difference relies on what is considered a regulatory unit, which shapes the analysis workflows. For *RTNduals*, the regulatory unit is the regulon (*e.g.* group of genes and a given regulator) and regulation is investigated between regulons. For *CoRegNet*, the regulatory unit is formed by one gene and two regulators (*e.g.* cooperative regulations are enumerated in the first place) and regulation is investigated between regulators. These packages take complementary directions: *RTNduals* implements a top-down approach, breaking down regulons to gain insight into the subsystems, while *CoRegNet* implements a bottom-up strategy, linking together individual co-regulatory elements to form larger subsystems. Users may benefit from exploring regulatory associations with both packages to check the overall consistency of the results.

2 TCGA breast invasive carcinoma cohort (TCGA-BRCA)

2.1 Context

In **section 1**, we used a precalculated transcriptional network for the METABRIC breast cancer cohort, which we made available as the `Fletcher2013b` data package. In **section 2**, we will show how to prepare input data for the *RTN* and *RTNduals* packages using the publicly available mRNA-seq data for the TCGA-BRCA cohort. We will show how to download harmonized GRCh38/hg38 data from the Genomic Data Commons (GDC) using the TCGAbiolinks package (Colaprico *et al.*, 2016). The preprocessing will generate generate a `SummarizedExperiment` object that contains gene expression data, which we will then use to generate the transcriptional network. The subsequent steps will infer *dual regulons*, following exactly the same steps as described in **section 1.4**.

2.2 Using TCGAbiolinks to download data from GDC

Please ensure you have installed all libraries before proceeding.

```
library(SummarizedExperiment)
library(TCGAbiolinks)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
```

We'll use the Bioconductor package `TCGAbiolinks` to query and download from the GDC. We are looking for the harmonized, normalized RNA-seq data for the TCGA-BRCA cohort.

`TCGAbiolinks` will create a directory called GDCdata in your working directory and will save into it the files downloaded from the GDC. The download can take a while. The files for each patient will be downloaded in a separate file. Then, the `GDCprepare` function will compile the files into an R object of class `RangedSummarizedExperiment`.

The `RangedSummarizedExperiment` has 6 slots. The most important are `rowRanges` (gene metadata), `colData` (patient metadata), and `assays`, which contains the gene expression matrix.

```
#-- Subset BRCA cohort for a quicker demonstration
subsample <- TCGAquery_subtype(tumor = "BRCA")
subsample <- subsample$patient[sample(nrow(subsample),500)]
```

```
#-- Download mRNA TCGA-BRCA data
query <- GDCquery(project= "TCGA-BRCA",
                  data.category = "Transcriptome Profiling",
                  data.type = "Gene Expression Quantification",
                  experimental.strategy = "RNA-Seq",
                  workflow.type = "HTSeq - FPKM",
                  sample.type = c("Primary solid Tumor"),
                  barcode = subsample)

GDCdownload(query)
tcgaBRCA_mRNA_data <- GDCprepare(query)
```

The object downloaded from GDC contains gene-level expression data that includes both coding and noncoding genes (*e.g.* lncRNAs). We will filter these, retaining only genes annotated in the UCSC hg38 known gene list.

```
#-- Subset by known gene locations
geneRanges <- genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
tcgaBRCA_mRNA_data <- subsetByOverlaps(tcgaBRCA_mRNA_data, geneRanges)
```

Finally, we'll change column names for better internal pre-processing in RTN's `tni.constructor` function. Having the `SYMBOL` column will enable genes with the same symbol to be preprocessed by this function. When this step has been run, the `tcgaBRCA_mRNA_data` object is ready for the RTN pipeline.

```
#-- Change column names for best 'tni.constructor' summarizations
#-- and save the preprocessed data for subsequent analyses
colnames(rowData(tcgaBRCA_mRNA_data)) <- c("ENSEMBL", "SYMBOL", "OG_ENSEMBL")
save(tcgaBRCA_mRNA_data, file = "tcgaBRCA_mRNA_data_preprocessed.RData")
```

2.3 Inferring the regulatory network with *RTN*

The RTN pipeline starts with the construction of a `TNI-class` object, using the `tni.constructor` method. This method takes in a matrix of gene expression and metadata on the samples and genes, as well as a list of the regulators to be evaluated. Here, the expression matrix and metadata are available as a `SummarizedExperiment` object, and the list of regulators will be extracted from the `rtni1st` object with the `tni.get` accessory function. The `tni.constructor` method will check the consistency of all the given arguments. The inference pipeline is then executed in three subsequent steps: (i) compute mutual information (MI) between a regulator and all potential targets, removing non-significant associations by permutation analysis, (ii) remove unstable interactions by bootstrapping, and (iii) apply the ARACNe algorithm, which uses the data processing inequality (DPI) theorem to remove indirect interactions(for additional details, please refer to Margolin *et al.* (2006) and Fletcher *et al.* (2013)). Briefly, consider three random varibles, `X`, `Y` and `Z` forming a network triplet, with `X` interacting with `Z` only through `Y` (*i.e.*, the interaction network is `X->Y->Z`), and no alternative path exists between `X` and `Z`). The DPI theorem states that the information transferred between `Y` and `Z` is always larger than the information transferred between `X` and `Z`. Based on this assumption, the ARACNe algorithm scans all triplets formed by two regulators and one target and removes the edge with the smallest MI value of each triplet, which is regarded as a redundant association. As the DPI filter eliminates shared targets between regulators, the overlap between regulons is observed in the interactions not removed by the ARACNe algorithm (see below).

```
#-- Get the list of regulatoryElements available from the 'rtni1st' object
regulatoryElements <- names(tni.get(rtni1st, what="regulatoryElements"))
```

```
#-- TNI constructor
rtni_tcgaBRCA <- tni.constructor(tcgaBRCA_mRNA_data,
                                 regulatoryElements = regulatoryElements)
```

To compute a large regulatory network we recommend using a multithreaded mode with the **snow** package. As minimum computational resources, we suggest a processor with >= 4 cores and RAM >= 8 GB per core (specific routines should be adjusted for the available resources). The `makeCluster` function will set the number of nodes to create on the local machine, making a `cluster` object available for the `TNI-class` methods. This example (29885 rows vs. 500 columns gene expression matrix and 809 regulators) should take 2h to conclude when running in a 2.9 GHz Core i9-8950H workstation with 32GB DDR4 RAM.

```
#-- Compute the reference regulatory network by permutation and
#-- bootstrap analyses. For RNA-seq data we recommend using the
#-- non-parametric estimator of the mutual information
#-- (estimator = "spearman").
library(snow)
options(cluster=makeCluster(4, "SOCK"))
rtni_tcgaBRCA <- tni.permutation(rtni_tcgaBRCA, pValueCutoff = 10^-7,
                                 estimator = "spearman")
rtni_tcgaBRCA <- tni.bootstrap(rtni_tcgaBRCA, nBootstraps = 200)
stopCluster(getOption("cluster"))
```

Next we run the ARACNe algorithm. Note that the overlap between regulons is affected by the `eps` argument, which sets the threshold for removing the edge with the smallest MI value of each triplet (see comments above and the `aracne` function documentation). In order to access the overlap between regulons in *RTNduals*, we recommend setting `eps = NA`, which will result in the MI threshold being estimated from the empirical null distribution computed in the permutation and bootstrap steps.

```
#-- Compute the DPI-filtered regulatory network
rtni_tcgaBRCA <- tni.dpi.filter(rtni_tcgaBRCA, eps = NA)
```

```
#-- Save the TNI object for subsequent analyses
save(rtni_tcgaBRCA, file="rtni_tcgaBRCA.RData")
```

Please note that some level of missing annotation is expected, as not all gene symbols listed in the `regulatoryElements` might be available in the TCGA-BRCA preprocessed data. Also, data that are inconsistent with the calculation may be removed in the `tni.constructor` preprocess; for example, it is not possible to test associations for a gene whose expression does not vary across samples, so such genes are not included in the analysis (in this example, genes "SOX10", "XBP1" and "ZNF434" were missed or removed, so we tested fewer regulons than those tested in **section 1**). For a summary of the resulting regulatory network we recommend using the `tni.regulon.summary` function.

```
#-- Summary
tni.regulon.summary(rtni_tcgaBRCA)
```

```
## This regulatory network comprised of 771 regulons.

## -- DPI-filtered network:

## regulatoryElements            Targets              Edges
##                771              21389              81866
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0      28      71     106     136    1600

## -- Reference network:

## regulatoryElements            Targets              Edges
##                771              21389            1724922
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     326    1802    2237    3674    8013
## ---
```

Additionally, all parameters, input data, and results available in the final `TNI-class` object can be retrieved by the `tni.get` accessory function.

```
#-- For example, to retrieve DPI-filtered regulons with a given annotation
regulons <- tni.get(rtni_tcgaBRCA, what="regulons", idkey="SYMBOL")
```

Note that the MI statistics are based on a gene's expression varying across a cohort. If a gene's expression does not vary across a cohort, it is not possible to associate this gene's expression with the expression of other genes in the cohort. As an extreme case (noted above), genes that exhibit no variability (*e.g.* that are not expressed in all samples) are excluded from the analysis. Large cohorts of tumour samples typically contain multiple molecular subtypes, and typically provide good expression variability for building regulons. In contrast, sample sets that are more homogeneous may be more challenging to explore with regulons, and this may be the case with sets of normal, non-cancerous samples. We do not recommend computing regulons for cohorts of low variability, or for subsets of a cohort.

2.4 Inferring *dual regulons*

The `mbrAssociation` function will call dual regulons following exactly the same steps as described in **section 1.4**, but now for regulons computed for the TCGA-BRCA cohort.

```
#-- Run 'tni2mbrPreprocess' and check datasets
mbr_tcgaBRCA <- tni2mbrPreprocess(tni = rtni_tcgaBRCA, regulatoryElements = risk.tfs)
```

```
#-- Run 'mbrAssociation' pipeline
mbr_tcgaBRCA <- mbrAssociation(mbr_tcgaBRCA, pValueCutoff = 0.05)
```

```
#-- Get a list of dual regulons
#-- (Supplementary Table 3)
mbr_tcgaBRCA_results <- mbrGet(mbr_tcgaBRCA, "dualsCorrelation")
mbr_tcgaBRCA_results[1:10,]
```

Supplementary Table 3. Top 10 *dual regulons* in the TCGA-BRCA cohort.

Dual Regulon	MI Regulators	R Regulons	Pvalue	Adjusted Pvalue*
ESR1~FOXA1**	0.42	0.85	6.57e−90	3.47e−87
FOXA1~GATA3**	0.43	0.77	2.45e−82	1.29e−79
ESR1~GATA3**	0.37	0.84	1.72e−76	9.07e−74
CEBPB~FOXA1**	0.24	−0.78	2.53e−75	1.33e−72
FOXA1~YBX1**	0.27	−0.70	8.99e−73	4.75e−70
AFF3~ESR1**	0.30	0.74	1.17e−58	6.19e−56
ESR1~ZNF552**	0.30	0.84	1.39e−56	7.34e−54
AR~FOXA1**	0.25	0.62	2.20e−53	1.16e−50
ESR1~YBX1**	0.30	−0.71	5.46e−52	2.88e−49
NFIB~TBX19	0.15	0.68	6.69e−47	3.53e−44

*Benjamini-Hochberg (BH) adjusted p-values
**Dual regulons also listed in Supplementary Figure 2

Supplementary Table 3 shows the top 10 dual regulons in the *mbr_tcgaBRCA_results* object; nine of these are listed in **Supplementary Figure 2**, which shows dual regulons inferred in the microarray-based METABRIC cohort 1 data.

2.5 Retrieving target genes from *dual regulons*

Possible follow-up analyses include enrichment-based methods (*e.g.* pathway enrichment analysis) to explore genes co-regulated by dual regulons. For such analyses, users may want to extract information about dual regulons from the `MBR-class` object. Below, we show how to retrieve the target genes in each regulon, and the number and names of target genes that are shared between two regulons.

```
#-- Get the overlap statistics
mbr_tcgaBRCA_overlap <- mbrGet(mbr_tcgaBRCA, "dualsOverlap")
mbr_tcgaBRCA_overlap[1:3,-c(1,2)]
```

```
##             Universe.Size Regulon1.Size Regulon2.Size Expected.Overlap
## NFIB~TRIM29         21389           156           375         2.735051
## FOXA1~GATA3         21389           668           407        12.711020
## NFIB~TBX19          21389           156           450         3.282061
##             Observed.Overlap       Pvalue Adjusted.Pvalue
## NFIB~TRIM29               39 6.506605e-34    3.435487e-31
## FOXA1~GATA3               69 4.118702e-31    2.174674e-28
## NFIB~TBX19                39 7.530390e-31    3.976046e-28
```

```
#-- Extract network summary and regulons
tni <- mbrGet(mbr_tcgaBRCA, what="TNI")
tniSummary <- tni.get(tni, what="summary")
regulons <- tni.get(tni, what="regulons", idkey = "SYMBOL")
Genes.In.Overlap <- intersect(regulons$FOXA1,regulons$GATA3)
head(Genes.In.Overlap, n = 5)
```

```
## [1] "LIN28B"     "CAMKV"      "AP000477.2" "SLC26A9"    "CASC8"
```

```
#-- Reproduce the overlap statistics (e.g. FOXA1~GATA3 dual regulon)

#-- (1) Get counts
Universe.Size <- tniSummary$results$tnet[1,"Targets"]
Regulon1.Size <- length(regulons$FOXA1)
Regulon2.Size <- length(regulons$GATA3)
Observed.Overlap <- length(Genes.In.Overlap)
Expected.Overlap <- (Regulon2.Size/Universe.Size)*Regulon1.Size

#-- (2) Run 'phyper' function
Pvalue <- phyper(q = Observed.Overlap - 1,
                 m = Regulon1.Size,
                 n = Universe.Size - Regulon1.Size,
                 k = Regulon2.Size, lower.tail = FALSE)

#-- (3) Check results
data.frame(Universe.Size, Regulon1.Size, Regulon2.Size, Expected.Overlap,
           Observed.Overlap, Pvalue)
```

```
##   Universe.Size Regulon1.Size Regulon2.Size Expected.Overlap
## 1         21389           668           407         12.71102
##   Observed.Overlap       Pvalue
## 1               69 4.118702e-31
```

Session information

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] TxDb.Hsapiens.UCSC.hg38.knownGene_3.4.0
##  [2] GenomicFeatures_1.34.1
##  [3] AnnotationDbi_1.44.0
##  [4] TCGAbiolinks_2.10.0
##  [5] SummarizedExperiment_1.12.0
##  [6] DelayedArray_0.8.0
##  [7] BiocParallel_1.16.4
##  [8] matrixStats_0.54.0
##  [9] Biobase_2.42.0
## [10] GenomicRanges_1.34.0
## [11] GenomeInfoDb_1.18.1
## [12] IRanges_2.16.0
## [13] S4Vectors_0.20.1
## [14] BiocGenerics_0.28.0
## [15] pheatmap_1.0.10
## [16] RTNduals_1.7.2
## [17] Fletcher2013b_1.18.0
## [18] igraph_1.2.2
## [19] RedeR_1.30.0
## [20] RTN_2.7.3
## [21] Fletcher2013a_1.18.0
## [22] limma_3.38.3
##
## loaded via a namespace (and not attached):
##   [1] backports_1.1.3             snow_0.4-3
##   [3] circlize_0.4.5              aroma.light_3.12.0
##   [5] plyr_1.8.4                  selectr_0.4-1
##   [7] ConsensusClusterPlus_1.46.0 lazyeval_0.2.1
##   [9] splines_3.5.3               ggplot2_3.1.0
##  [11] sva_3.30.0                  digest_0.6.18
##  [13] foreach_1.4.4               htmltools_0.3.6
##  [15] gdata_2.18.0                magrittr_1.5
##  [17] memoise_1.1.0               cluster_2.0.7-1
##  [19] doParallel_1.0.14           mixtools_1.1.0
##  [21] ComplexHeatmap_1.20.0       Biostrings_2.50.1
##  [23] readr_1.3.1                 annotate_1.60.0
##  [25] R.utils_2.7.0               prettyunits_1.0.2
##  [27] colorspace_1.3-2            blob_1.1.1
##  [29] rvest_0.3.2                 ggrepel_0.8.0
```

```
##  [31] xfun_0.4                   dplyr_0.7.8
##  [33] crayon_1.3.4               RCurl_1.95-4.11
##  [35] jsonlite_1.6               genefilter_1.64.0
##  [37] bindr_0.1.1                zoo_1.8-4
##  [39] survival_2.43-3            iterators_1.0.10
##  [41] glue_1.3.0                 survminer_0.4.3
##  [43] gtable_0.2.0               zlibbioc_1.28.0
##  [45] XVector_0.22.0             GetoptLong_0.1.7
##  [47] shape_1.4.4                scales_1.0.0
##  [49] DESeq_1.34.0               futile.options_1.0.1
##  [51] DBI_1.0.0                  edgeR_3.24.3
##  [53] ggthemes_4.0.1             Rcpp_1.0.0
##  [55] cmprsk_2.2-7               xtable_1.8-3
##  [57] progress_1.2.0             bit_1.1-14
##  [59] matlab_1.0.2               km.ci_0.5-2
##  [61] httr_1.4.0                 gplots_3.0.1
##  [63] RColorBrewer_1.1-2         pkgconfig_2.0.2
##  [65] XML_3.98-1.16              R.methodsS3_1.7.1
##  [67] locfit_1.5-9.1             tidyselect_0.2.5
##  [69] rlang_0.3.0.1              munsell_0.5.0
##  [71] tools_3.5.3                downloader_0.4
##  [73] generics_0.0.2             RSQLite_2.1.1
##  [75] broom_0.5.1                evaluate_0.12
##  [77] stringr_1.3.1              yaml_2.2.0
##  [79] knitr_1.21                 bit64_0.9-7
##  [81] survMisc_0.5.5             caTools_1.17.1.1
##  [83] purrr_0.2.5                bindrcpp_0.2.2
##  [85] nlme_3.1-137               EDASeq_2.16.0
##  [87] formatR_1.5                R.oo_1.22.0
##  [89] xml2_1.2.0                 biomaRt_2.38.0
##  [91] compiler_3.5.3             e1071_1.7-0
##  [93] minet_3.40.0               viper_1.16.0
##  [95] tibble_1.4.2               geneplotter_1.60.0
##  [97] stringi_1.2.4              futile.logger_1.4.3
##  [99] lattice_0.20-38            Matrix_1.2-17
## [101] KMsurv_0.1-5               pillar_1.3.1
## [103] GlobalOptions_0.1.0        data.table_1.11.8
## [105] bitops_1.0-6               rtracklayer_1.42.1
## [107] R6_2.3.0                   latticeExtra_0.6-28
## [109] hwriter_1.3.2              ShortRead_1.40.0
## [111] KernSmooth_2.23-15         gridExtra_2.3
## [113] codetools_0.2-16           lambda.r_1.2.3
## [115] MASS_7.3-51.1              gtools_3.8.1
## [117] assertthat_0.2.0           rjson_0.2.20
## [119] GenomicAlignments_1.18.0   Rsamtools_1.34.0
## [121] GenomeInfoDbData_1.2.0     mgcv_1.8-28
## [123] hms_0.4.2                  VennDiagram_1.6.20
## [125] grid_3.5.3                 tidyr_0.8.2
## [127] class_7.3-15               rmarkdown_1.11
## [129] segmented_0.5-3.0          ggpubr_0.2
```

Supplementary References

Beisser,D. *et al.* (2010) BioNet: an R-Package for the functional analysis of biological networks. *Bioinformatics*, **26**, 1129–1130.

Campbell,T.N. *et al.* (2018) ERα Binding by Transcription Factors NFIB and YBX1 Enables FGFR2 Signaling to Modulate Estrogen Responsiveness in Breast Cancer. *Cancer Research*, **78**, 410–421.

Castro,M.A.A. *et al.* (2016) Regulators of genetic risk of breast cancer identified by integrative network analysis. *Nature Genetics*, **48**, 12–21.

Colaprico,A. *et al.* (2016) TCGAbiolinks: An r/bioconductor package for integrative analysis of tcga data. *Nucleic Acids Research*, **44**, e71.

Curtis,C. *et al.* (2012) The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, **486**, 346–352.

Fletcher,M.N. *et al.* (2013) Master regulators of FGFR2 signalling and breast cancer risk. *Nature Communications*, **4**, 2464.

Margolin,A.A. *et al.* (2006) ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, **7**, S7.

Meyer,P.E. *et al.* (2008) Minet: A r/bioconductor package for inferring large transcriptional networks using mutual information. *BMC Bioinformatics*, **9**, 461.

Nicolle,R. *et al.* (2015) CoRegNet: reconstruction and integrated analysis of co-regulatory networks. *Bioinformatics*, **31**, 3066–3068.

Panaccione,A. *et al.* (2017) Expression Profiling of Clinical Specimens Supports the Existence of Neural Progenitor-Like Stem Cells in Basal Breast Cancers. *Clinical Breast Cancer*, **17**, 298–306.e7.

Rouveirol,C. *et al.* (2007) LICORN: learning cooperative regulation networks from gene expression data. *Bioinformatics*, **23**, 2407–2414.

Theodorou,V. *et al.* (2013) GATA3 acts upstream of FOXA1 in mediating ESR1 binding by shaping enhancer accessibility. *Genome Research*, **23**, 12–22.