R package icreport Vignette

Jinhyun Ju

1 Introduction

The `icreport` package's main functionality inlcudes running independent component analysis (ICA) or principal component analysis (PCA) on a given data set and subsequently generating a comprehensive and compact report of the results.

2 Important Note

The report generating feature of this package requires a program called `pandoc` version 1.12.3 or higher to be installed. This is not a problem when you are using the most recent version of Rstudio (0.98.1102 at the moment - 2 March 2015), since it comes with the required `pandoc` functionality. However, if you are running `icreport` from good-old-fashioned-R you might run into an error message that pandoc is not installed. In such a case, please review the following link to install the correct version of pandoc on your machine. `https://github.com/rstudio/rmarkdown/blob/master/PANDOC.md`

3 Quickstart

If you already know what you want to use the package for, follow this simple example to get started!

3.1 Installation

The package is not yet published on bioconductor or CRAN, so the best way to install the package is directly from github. Installing the package through the function `install_github` from the package `devtools`.

```
install.packages("devtools") # in case you don't have devtools already

devtools::install_github("jinhyunju/icreport") #installing icreport
```

3.2 Loading an example dataset

Here we are going to use a public dataset that is available on the Gene Expression Omnibus (GEO). You can use your own dataset, it just needs to be a matrix which has the dimension of (gene x samples). A dataframe with covariate information is optional with dimensions (samples x covariates).

To generate the example dataset, all you have to do is source the script included in the package. If you are interested in the details of the script you can check the path to the script by printing out the value that is saved in the `example.data.script` object and open it up in any text editor.

Please be aware that the script will install two packages `GEOquery` and `biomaRt` if you don't already have it on your machine.

```
example.data.script <- system.file("templates/create_example_data.R",
                                   package="icreport")

source(example.data.script)
```

This will take a few minutes depending on your internet connection, since it is downloading data from GEO and biomaRt. In case everything ran correctly, it will generate 3 objects, `expr.data`, `sample.info`, and `probe.info`.

- `expr.data` gene expression measurements for 26383 probes and 47 samples.
- `sample.info` 5 covariates for each sample
- `probe.info` positional information for 26828 probes

Onething that you have to watch out for is that the rownames of `sample.info` have to match the column names of the `expr.data`. They don't necessarily have to be in the same order but they should have names that overlap.

3.3 Running ICA on the expression dataset.

Now let's get down to business. The basic inputs for the function `gene_expr_ica` are as follows:

- `phenotype.mx` should be the expression matrix with genes in rows and samples in columns (dimension = genes x samples)
- `info.df` should be the sample information data frame with the covariates in columns (dimension = samples x covariates)
- `check.covars` should be a vector that contains the names of the covariates (column names of `sample.info`) that should be tested for associations with independent components. In this case we are testing all 5 covariates for associations.

For information regarding other advanced options please use `?gene_expr_ica`. The functioncal for a case when you have covariates that you want to check would look like this:

```
library(icreport)
ica.result <- gene_expr_ica(phenotype.mx = expr.data,
                            info.df = sample.info,
                            check.covars = colnames(sample.info))

# This may also take a few minutes depending on the size of your dataset.
```

In case you don't have any covariates the only input you need is the expression data matrix:

```
ica.result <- gene_expr_ica(phenotype.mx = expr.data)
```

3.4 Generating ICA output reports

After running ICA on a given dataset you can easily generate an HTML report to review the results using the function `report2me()`. The essential inputs for `report2me` are:

- `prefix` specifies the name of the html report file.
- `geneinfo.df` should be a dataframe that contains information about gene (or probe) positions. The column names should be "phenotype" for the genes or probes (rownames of `expr.data`), "pheno_chr" for chromosomes, "pheno_start" for starting coordinates and "pheno_end" for end coordinates as shown in the example below.

```
head(probe.info)
```

```
    phenotype pheno_chr pheno_start pheno_end
1 216705_s_at        20    44619522  44652233
2   204639_at        20    44619522  44652233
3   203440_at        18    27950966  28177446
```

```
4   242876_at          1   243488233 243851079
5   222880_at          1   243488233 243851079
6   212607_at          1   243488233 243851079
```

You can specify the output path using the option `output.path = "/path/to/directory"`. Please note that sometimes directories starting with the short cut for home `"~"` are not recognized, so I recommend setting the working directory to the desired output directory first or specifying the full path.

```
report2me(input = ica.result,
          prefix = "Test_ICA_Report",
          geneinfo.df = probe.info)
```

3.5 Running PCA on the expression dataset.

You can use the function `gene_expr_pca()` to run PCA on your dataset. Running PCA on a dataset is almost identical to the process of running ICA, and the essential inputs for the function are the same. In case you have covariates:

```
pca.result <- gene_expr_pca(phenotype.mx = expr.data,
                            info.df = sample.info,
                            check.covars = colnames(sample.info))

## Error in eval(expr, envir, enclos):  could not find function "gene_expr_pca"
```

In case you don't have any covariates:

```
pca.result <- gene_expr_pca(phenotype.mx = expr.data)
```

3.6 Generating PCA output reports

The process of generating a PCA report is identical to the process of generating an ICA report. `report2me()` automatically detects whether the inputs are ICA or PCA results through the `method` attribute and generates a report with the corresponding format.

```
report2me(input = pca.result,
          prefix = "Test_PCA_Report",
          geneinfo.df = probe.info)
```

4 Interpreting Outputs

Before we describe what each plot in the output report means, let us briefly review the model of ICA. The key assumption of ICA is that each observed sample $\mathbf{y}_i$ for $i = 1, ..., n$, is a linear combination of k independent components $\mathbf{s}_1, ..., \mathbf{s}_k$ such that:

$$\mathbf{y}_i = a_{i1}\mathbf{s}_1 + a_{i2}\mathbf{s}_2 + ... + a_{ik}\mathbf{s}_k \tag{1}$$

where each $\mathbf{s}_j$ for $j = 1, ..., k$ are the independent components and the corresponding a_{ij}s are scalar coefficients that represent the relative weights of each component in sample $\mathbf{y}_i$.

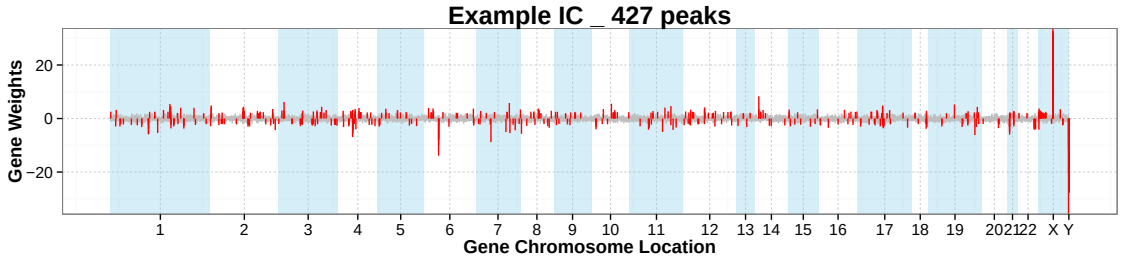
4.1 Individual IC plots

Let us begin with the individual IC plots that constitute most of the report. Each individual plot will first have some printed information about the component.

```
    Variance_Percent Correlated_Covariate     p.value Number_of_Peaks
IC7         4.094776               gender 5.51875e-41             433
---- Top 10 Gene Weights ----
    201909_at 221728_x_at 214218_s_at 224588_at 224590_at 227671_at
IC7    -35.75       33.72       32.69     31.71     31.65     31.54
    231592_at 205000_at 204409_s_at 206700_s_at
IC7     30.28    -27.75       -27.7      -27.46
```

The first two lines summarize the given component by the percent variance, associated covariate, the corresponding p-value for the association, and the number of "peaks". The percent variance is an equivalent to the concept in PCA. Associated covariates are found using ANOVA and called significant if they pass the bonferroni corrected threshold. Peaks are showing the number of genes that have gene weights greater than 2 standard deviations of the given component (more on this to follow). The following lines show the top 10 genes (or probes) contributing the most to the given component and their corresponding gene weight.

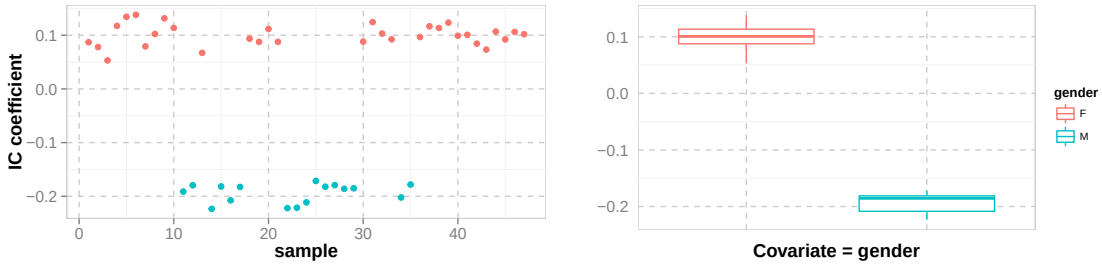
The summarized information is followed by 3 plots that describe a given component. The first plot shows the gene weights for the component sorted by chromosome position.



Explain peaks, gene weights.

Each component $\mathbf{s}_j$ can be thought of as a vector with length g, where g is equal to the number of measured genes. You can think of each component as a direction in the g dimensional gene space, thus we can assess the importance of a specific gene in each direction by looking at their gene weights (or loadings). For example, if the 3rd entry g_3 is the largest for component $\mathbf{s_1}$, we can assume that g_3 is the most important gene for component $\mathbf{s_1}$.

The next plots are showing the IC coefficients for the given component in a scatter and boxplot.



Explain IC coefficients, scenarios where associated covariate exists and doesn't.

Explain PC plots as well.