Using Messina

Mark Pinese

April 26, 2022

Contents

1 Introduction

Messina is a package of algorithms for generating optimal single-gene threshold classifiers. These classifiers are directly useful when using continuous data (eg. gene expression measurements) to develop diagnostic or prognostic tests that will be based on binary assays such as immunohistochemistry.

Messina is also useful for the flexible detection of differential expression in the presence of outliers, either due to experimental variations or hidden sample heterogeneity. When used in this fashion, Messina can detect genes or proteins with biologically-interesting heterogeneous patterns of expression, that are missed by conventional statistics. Such heterogeneity in expression is a hallmark of sample subtypes, and Messina can be used to detect features that could be reflective of underlying, and undiscovered, sample subtypes.

The original Messina algorithm for classification and differential expression analysis is described in the Messina paper [1].

2 Using Messina to construct optimal diagnostic classifiers

This section will illustrate how Messina can be used to identify single features (eg genes) that would be best suited to form the basis of a diagnostic test.

2.1 The problem

Translating gene expression data into a robust clinical test is challenging:

1. Although gene expression datasets typically measure the levels of many thousands of genes, practical clinical diagnostic tests only measure a very small number of markers (typically just one).
2. High-throughput gene expression data and clinical tests use very different measurement methods, so that a finding in one is not necessarily repeatable in the other.
3. Clinical samples are subject to much more varied handling than experimental samples, requiring clinical tests to be very robust to handling variations.

Point (1) above means that although sophisticated classification algorithms can produce high-performance multi-feature classifiers on gene expression data, these classifiers are not translatable to a clinical test. Special classifiers developed to work with very small numbers of genes are required. Points (2) and (3) reflect that, in the translation of a laboratory result to a clinical diagnostic context, maximum robustness is desirable to give the test the best chance of performing satisfactorily in the less controlled clinical environment, using completely different measurement methods.

Conventional classifiers (eg. SVM, kNN) are poorly suited to the problem of generating diagnostic tests, as they do not strongly control for the number of features used in the classifier. Statistical tests (eg. t-test) work on a per-feature basis, but are not specifically designed for classification problems, and so do not produce optimally robust tests. Messina was specifically designed to solve this problem of generating optimal single-gene classifiers.

2.2 Example: Designing a colon cancer screening test

We use as an example the problem of distinguishing between colon biopsies of non-neoplastic tissue, and those containing colon cancer. In this example, we are ultimately interested in developing a screening test. Therefore, the per-test cost should be low, and ideally the test should be so robust that it may also work in samples more conveniently accessed than tissue biopsies, such as peripheral blood. Additionally, as the test will be a screen, we require that the sensitivity of the test be high, although we are willing to accept a lower specificity if it will gain us additional robustness.

We use the colon cancer expression dataset available in Bioconductor package *antiProfiles*. This dataset has already been used to produce a high-performing diagnostic classifier, but it is based on the measurements of 542 Affymetrix probesets, and so would not be economical to use in the screening application we have in mind, as well as quite challenging to transfer between sites and technologies.

We start by loading Messina and the data.

```
library(messina)
library(antiProfilesData)
data(apColonData)
apColonData
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5339 features, 68 samples
##   element names: exprs
## protocolData: none
```

```
## phenoData
##   sampleNames: GSM95473 GSM95474 ... GSM215114 (68 total)
##   varLabels: filename DB_ID ... Status (7 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: hgu133plus2
```

The data contain a number of different sample classes. For our purposes we are only interested in comparing tumour and normal samples. We extract the expression data and class labels, and subset to just these types of interest. Messina does not require gene filtering, so we do not perform it at this stage, although in very large datasets this may be useful to reduce runtimes.

```
x = exprs(apColonData)
y = pData(apColonData)$SubType

sel = y %in% c("normal", "tumor")
x = x[,sel]
y = y[sel]
```

We are now ready to run the Messina algorithm. As we are interested in a screening test, we desire a high test sensitivity, but are willing to accept a lower test specificity. We choose a minimum sensitivity requirement of 0.95, and a minimum specificity requirement of 0.85. We now calculate the Messina fits, specifying that the "positive" class is tumour.

```
fit.apColon = messina(x, y == "tumor", min_sens = 0.95, min_spec = 0.85, seed = 1234, silent = TRUE)
```

The result is a messinaClassResult S4 object with overloaded show and plot functions.

```
fit.apColon
```

```
## An object of class MessinaClassResult
##
## Problem type:classification
## Parameters:
##   An object of class MessinaParameters
##   5339 features, 38 samples.
##   Objective type: sensitivity/specificity.  Minimum sensitivity: 0.95  Minimum specificity: 0.85
##   Minimum group fraction: 0
##   Training fraction: 0.9
##   Number of bootstraps: 50
##   Random seed: 1234
##
## Summary of results:
##   An object of class MessinaFits
##   161 / 5339 features passed performance requirements (3.02%)
##   Top features:
##             Passed Requirements Classifier Type Threshold Value Direction
## 204719_at                 TRUE       Threshold        6.326002        -1
## 207502_at                 TRUE       Threshold        8.048459        -1
## 206784_at                 TRUE       Threshold       10.339348        -1
```

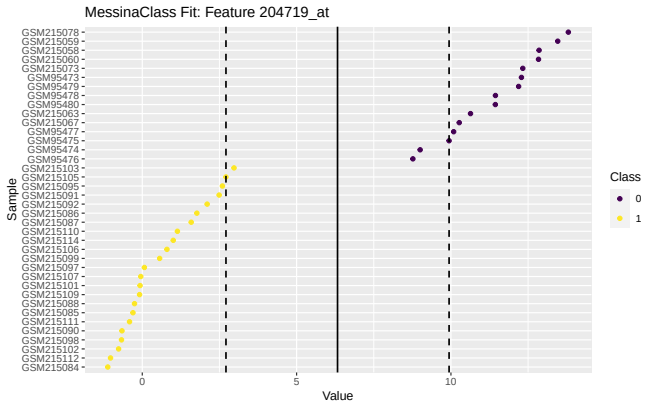
```
## 206134_at                TRUE       Threshold       15.667098        -1
## 207003_at                TRUE       Threshold       10.530530        -1
## 213921_at                TRUE       Threshold        4.882769        -1
## 204259_at                TRUE       Threshold        2.781876         1
## 209735_at                TRUE       Threshold        6.470396        -1
## 205950_s_at              TRUE       Threshold       14.161574        -1
## 206422_at                TRUE       Threshold       11.371876        -1
##               Margin
## 204719_at   7.228815
## 207502_at   6.366498
## 206784_at   6.205261
## 206134_at   6.043961
## 207003_at   6.013640
## 213921_at   5.706219
## 204259_at   4.928655
## 209735_at   4.927077
## 205950_s_at 4.368962
## 206422_at   3.950427
```

Here we see that Messina found 161 probesets that were suitable as the basis of threshold diagnostic classifiers for colon cancer. A summary of the best probesets (by margin) is given at the bottom of the show results. This can also be accessed directly, with more display options, using the messinaTopResults function.

By default, Messina ranks features in decreasing order of their *margin*, a concept borrowed from support vector machine theory. The margin is a measure of how robust the classifier is to noise, and is equal to the distance between the performance limits of the threshold value. These limits are the gene expression values beyond which the threshold cannot pass without the classifier no longer satisfying its user-supplied minimum performance requirements. The distance between the threshold and either limit is an indication of how much error must be present in a gene expression measurement for the performance guarantee of a Messina classifier to be violated. The greater the margin, the more robust the classifier.

To illustrate the performance limits, threshold, and margin concepts, we plot the Messina fit of the best probeset found. We override the Messina default and set the plot type to "point" (vs the default of "bar"), as the data are approximately log transformed.

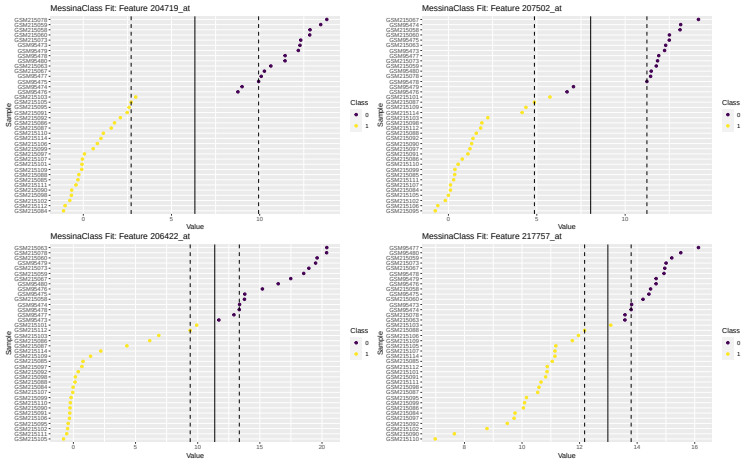
```
plot(fit.apColon, i = 1, plot_type = "point")
```



This plot displays the expression of each sample as a coloured point, with the colour reflecting the sample's group: 0 for non-neoplastic tissue, and 1 for cancer. The threshold that Messina selected as being optimal for separating cancer from normal is denoted by a solid black line, and on either side the dotted lines represent the performance limits of the classifier. If the level of error in expression measurement remains less than the distance between either dotted line and the threshold, the classifier's minimum sensitivity and specificity requirements will be satisfied. The distance between the two dotted lines is the margin.

In this example, the top probe found by Messina could perfectly separate cancer and normal samples, with a very wide margin of approximately 7.2 units, or about 150-fold. As we move down the list of Messina results, the separation between cancer and normal samples progressively worsens, and the probesets become less suitable for devleopment into a diagnostic. In this way, Messina provides a principled ranking of probesets to use as leads for the development of a diagnostic test, reporting the most robust leads first.

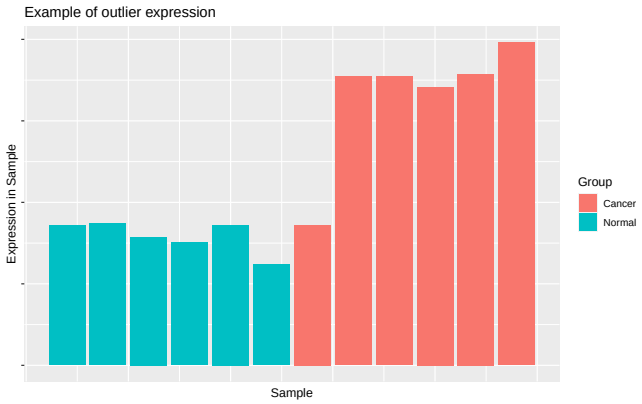
```
plot(fit.apColon, i = c(1,2,10,50), plot_type = "point")
```



3 Detecting differential expression in the presence of outliers

3.1 Expression outliers

Messina provides a powerful framework for detecting differences in expression between two sample groups, even in the presence of 'expression outliers'. Here we define expression outliers for a given feature (eg. gene) as samples that have a level of signal very different from that of most of their class. An example plot follows.



This figure shows the expression of a fictional gene in two experimental groups, cancer and normal, each containing six samples. In general, the cancer samples show increased expression from the normals, and most samples within each group are quite consistent in their signal. However, the leftmost cancer sample is displaying a signal that is consistent with the normal samples. This sample is an expression outlier.

Expression outliers are common features in clinical data, in which unmeasured (and possibly even unknown) covariates can affect expression levels. These covariates can be indicative of unknown substructure within a sample group: they can define sample subgroups. Clinical data are also inevitably not as well controlled for various experimental factors as are laboratory-derived measurements, and so outliers may simply be a consequence of experimental error. Regardless of whether the outliers are due to experimental error, or reflect new and unknown subgroups in the samples, it is valuable to have tools that can identify gene expression in a way that is robust to the presence of outliers.

Unfortunately, the presence of outliers violates a core assumption of classical statistical tests – that all samples within a group are sampled identically and are exchangable. For this reason, even modest levels of expression outliers drastically reduce the power of classical tests to identify differential expression[1], with resulting poor robustness to experimental artifacts, and the potential to ignore genes that have expression that is linked to an unknown subtype. As these genes are of great interest in many differential expression analyses, there is a real need for methods that can reliably identify differential expression in the presence of outliers. The following section will demonstrate the use of Messina in this task.

3.2 Example: Simulation and comparison to the t-test

Section under construction.

4 Identifying the best features to predict survival outcome

A new mode of Messina analysis the identification of optimal single features to use for prognosis, as opposed to the diagnosis case detailed previously. The core concept is the same: to find single-feature threshold classifiers able to separate patients into two groups. In the classification case above, the two groups were classes of patients, such as diseased and healthy. In the prognosis case presented here, the two groups are defined by their survival

time, as short survivors and long survivors. In its survival mode, Messina aims to identify single features that can robustly separate patients who will likely experience an event quickly, from those who will likely go for some time without an event.

In the classification case, the user inputs the minimum sensitivity and specificity that a Messina threshold classifier must satisfy, in effect creating a kind of cost-sensitive classification scheme. The survival case is similar. The user chooses an objective function that quantifies how different the survival times of two groups of patients is, and a minimum value of this function that all classifiers must be able to give on a test cohort.

MessinaSurv currently has built-in three objective functions: 'tau', 'reltau', and 'coxcoef'. These are detailed in the documentation for the MessinaSurv function, and have differing properties. In general, 'tau' is a very stable objective, but penalizes cutoffs that will separate the cohort into very unbalanced subgroups. 'reltau' and 'coxcoef' do not have this penalty, but are more prone to unstable fits. 'coxcoef' has an interpretation familiar to many working in the biomedical field, but is complex and by far the slowest objective function to calculate. 'tau' is a good first choice unless there is a strong reason to believe the cohort will contain a relatively small ($< 25\%$) subgroup of long- or short- survivors, but the user is encouraged to experiment with the different functions. A future version of Messina may allow for user-supplied objectives.

To illustrate the use of Messina on a survival problem, we load a (very small) subset of the TCGA KIRC expression data.

```
library(messina)
library(survival)
data(tcga_kirc_example)

## The data are present as a matrix of expression values, and a Surv object of
## survival times
dim(kirc.exprs)
```

```
## [1]   3 422
```

```
kirc.surv
```

```
##   [1] 1120+ 1436+   16+ 1190   735  1493+ 1491+ 1130+ 1508+ 1477+ 1105+ 1186+
##  [13]  751+ 1137+  785+  706+ 1485+  664+  944+  137  1097+ 1385+  910+  561
##  [25] 1425+ 1307+ 1286+ 1559+  319+  872+ 1054+  776+  735+  334+ 1313+ 1070+
##  [37]  630+  567+  574+  861+  726+  616+ 3343+  884  2566+ 2223+ 2017+ 1853+
##  [49] 2192+ 2086+ 2043+ 1744+ 1423+ 1471+ 1279+ 1217+ 1499+ 1481+ 1396+  873+
##  [61]  683    46+ 1168+  951+  853+  369+   40+  109    26     5+  214+   18
##  [73]  183   151+  162+  560+   22+   90+  103+    0+   48+   55+  186+  190+
##  [85]  677+   10+ 2180+ 2781+ 1487+ 1852+ 1042+  944+ 1135+ 2181+ 2425+  375+
##  [97] 1276+ 1158+  262+ 3335+ 2414+ 2592+ 2357+  957+ 2553+ 2438+ 1864+ 2324+
## [109]  178+  194+   50   110+   90+   15+  214+  315+  258+  248+  168+  343+
## [121]  291+  175+    7+   38+   16+  141+  166+   11+    7+  379+  279+  255+
## [133]   19+  226+  215+  181+   90+  133+   22+    4+   23+    8+  248+   14+
## [145]  194+  197+   15+   14+ 3377+ 2600  2881+ 2746+ 3074+ 2839+  992  2946+
## [157]   13+ 2718+  701  2343  1893+ 1879+ 1955+ 1670+ 2964+ 1624+  108   845+
## [169] 1888+ 1133+  645+  460     2  2746+  562  1589  2256  1666+ 1516+ 1493
## [181] 1367+  372+  969+  344   375  1034   953   374+ 2208+ 2324+ 2249+  182+
## [193] 1343  1270  2184+ 1461+  400+ 1876+  329   161  1884+ 1842+  410+ 1731+
## [205] 2080+  354+ 1854+  479+ 1489+ 1111+  620+ 1107+  334+ 1133+ 1124+  203+
## [217] 1459+  211+ 2660+ 2172+ 1935+ 1785+ 1834+ 1862+ 1871+  204+ 1745+ 1794+
```

```
## [229]  432+ 1487+ 1501+ 1384+  220+ 1433+ 1632+  454+ 1097  1013+ 1412+  951
## [241]  785+ 1124+  827   117+ 1044+  500+  176+ 1307+ 1371+  931+ 1266+  563+
## [253] 1177+ 1126+  840+ 1140+ 1071+ 1092   877  1463   192+ 2411+   54+ 2257+
## [265]  932+ 1590   293+ 1912  1892+ 1495+ 1165+ 1102+ 1133+ 1132+  693+  406+
## [277]  822  1010+  966+  714+  408+  748+ 1018+  603+ 1354+  881+  951+   29+
## [289] 1820+ 1416+ 1924+ 2227   431  2308+ 1998+ 1661  1792+  336   645  2554+
## [301] 1497+ 2422+ 1435+ 2258+ 2283+ 2353+ 1955+ 2186+ 2013+ 1883+  993+ 2125+
## [313] 1951+  649  1566  1945+ 2085+  818+ 1520+  749+  841  1200  1808+ 1527+
## [325] 1274  1450+ 1519+ 1559+ 1792+ 1495+ 1498+ 1530+ 1656+ 1372+   92   139
## [337]  572  1942+ 2190  2429+ 2575+  782   573   678   768   551  1883+ 1889+
## [349] 2230+ 2038+    6  1855+ 1905+ 2548+ 1964  2799+ 2489+  164  2609+ 2226+
## [361] 2016+ 2377+ 1075+ 2270+   41  2311+ 2308+ 2430+  570   774+ 1404    18+
## [373] 1432+ 1943+ 1787+  205+  445+ 1843+ 1690+ 2830   165  1768+ 1668+ 1556+
## [385]   25+  722   561  1558+ 1547+ 1558+ 1515+ 1430+  330   311   662+ 1492+
## [397] 1377+  685+   73    59   945     0+ 2042+ 1491+ 1629+  373+  435   693+
## [409] 1599+ 1329+ 1398+  909+  370+  355+ 1130+  723+ 1005+  727   550+  118+
## [421]  206+  126+
```

We then run messinaSurv on these data. Note that messinaSurv is currently rather slow. It is strongly recommended to use it in conjunction with doMC and as many cores as are available. Note that there are only 3 genes in this dataset – this very small number was chosen only to ensure that the vignette executed quickly. In practice, many thousands of genes may be assayed, and multiprocessing will be absolutely necessary.

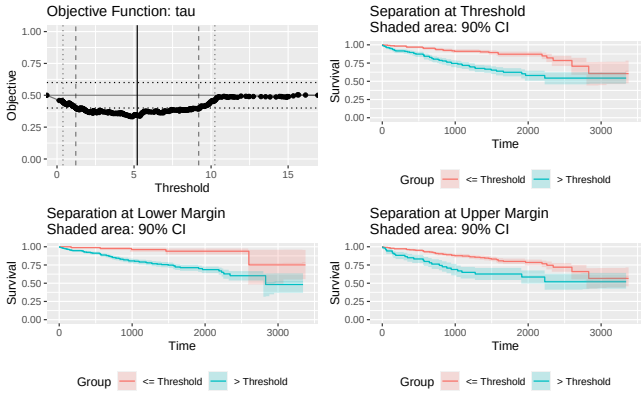
```
fit.surv = messinaSurv(kirc.exprs, kirc.surv, obj_func = "tau", obj_min = 0.6, parallel = FALSE, silent = TRU

fit.surv
```

```
## An object of class MessinaSurvResult
##
## Problem type:survival
## Parameters:
##   An object of class MessinaParameters
##   3 features, 422 samples.
##   Objective type: survival (tau).  Minimum objective value: 0.6
##   Minimum group fraction: 0.1
##   Training fraction: 0.8
##   Number of bootstraps: 50
##   Random seed:
##
## Summary of results:
##   An object of class MessinaFits
##   1 / 3 features passed performance requirements (33.33%)
##   Top features:
##                 Passed Requirements Classifier Type Threshold Value
## SAA1|6288                      TRUE       Threshold        5.207119
## CCDC74A|90557                 FALSE            <NA>        7.617463
## C1orf168|199920               FALSE            <NA>              NA
##                 Direction     Margin
## SAA1|6288              -1 7.97642745
## CCDC74A|90557          -1 0.04505006
## C1orf168|199920        NA 0.00000000
```

Plotting the fit.surv object produces a plot of the best feature found.

```
plot(fit.surv, bootstrap_type = "ci")
```



5 Session info

```
sessionInfo()
```

```
## R version 4.2.0 RC (2022-04-19 r82224)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.15-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.15-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB              LC_COLLATE=C
##  [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
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.5           antiProfilesData_1.31.0 Biobase_2.56.0
## [4] BiocGenerics_0.42.0     messina_1.32.0          survival_3.3-1
## [7] knitr_1.38
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.2    xfun_0.30           purrr_0.3.4
##  [4] splines_4.2.0       lattice_0.20-45     colorspace_2.0-3
##  [7] vctrs_0.4.1         generics_0.1.2      viridisLite_0.4.0
```

```
## [10] htmltools_0.5.2     yaml_2.3.5          utf8_1.2.2
## [13] rlang_1.0.2         pillar_1.7.0        withr_2.5.0
## [16] glue_1.6.2          DBI_1.1.2           foreach_1.5.2
## [19] lifecycle_1.0.1     plyr_1.8.7          stringr_1.4.0
## [22] munsell_0.5.0       gtable_0.3.0        codetools_0.2-18
## [25] evaluate_0.15       labeling_0.4.2      fastmap_1.1.0
## [28] fansi_1.0.3         highr_0.9           Rcpp_1.0.8.3
## [31] scales_1.2.0        BiocManager_1.30.17 farver_2.1.0
## [34] BiocStyle_2.24.0    digest_0.6.29       stringi_1.7.6
## [37] dplyr_1.0.8         grid_4.2.0          cli_3.3.0
## [40] tools_4.2.0         magrittr_2.0.3      tibble_3.1.6
## [43] crayon_1.5.1        pkgconfig_2.0.3     ellipsis_0.3.2
## [46] Matrix_1.4-1        assertthat_0.2.1    rmarkdown_2.14
## [49] iterators_1.0.14    R6_2.5.1            compiler_4.2.0
```

References

[1] Mark Pinese, Christopher J. Scarlett, James G. Kench, Emily K. Colvin, Davendra Segara, Susan M. Henshall, Robert L. Sutherland, and Andrew V. Biankin. Messina: A novel analysis tool to identify biologically relevant molecules in disease. *PLoS ONE*, 4(4):e5337, Apr 2009. URL: http://dx.doi.org/10.1371/journal.pone.0005337, doi:10.1371/journal.pone.0005337.