An Introduction to lr2cluster

Kurnia Susvitasari

2022-10-29

This document provides an introductory tutorial on the `lr2cluster` package. This package implements tools to help identify cluster assignments using multiple data sources when direct methods such as contact tracing and genomic sequencing are only available for some data. There are also complimentary functions, that can be used to estimate a cluster's true size and to choose which new cases to be sequenced next. This tutorial provides an overview of `lr2cluster`'s basic functionalities.

Installing the Package

To install the package, you need to install the package remotes first:

```
install.packages("remotes")
```

Once remotes is successfully installed, run the following to install and load `lr2cluster`:

```
remotes::install_github("ksusvita92/lr2cluster")
```

```
library(lr2cluster)
```

Valencia Data

Data for TB cases in Valencia is available at https://doi.org/10.7554/eLife.76605. The following is a short script to prepare the data. In this tutorial, we do not provide the cases' true geographical information. The location data used in this tutorial is a toy example which can be downloaded from GitHub repository here.

```
library(readxl)
library(dplyr)

url <- paste("https://elifesciences.org/download/aHR0cHM6Ly9jZG4uZWxpZmVzY2llbmNlcy5vcmc",
"vYXJ0aWNsZXMvNzY2MDUvZWxpZmUtNzY2MDUtc3VwcDEtdjEueGxzeA--/elife-76605-supp1-v1.xlsx?_ha",
"sh=pzQwKD1DzDLre7kKrWI%2Fhd%2BjY2FGgpekrPI4vXrlWNo%3D", sep = "")
destfile <- "rawdt.xlsx"
curl::curl_download(url, destfile)
rawdt <- read_excel(destfile, range = "A3:AB778", na = "NA")

# subset the necessary column and cases
tb_valencia <- rawdt %>%
  transmute(ID = ...1, Cluster = `Genomic
Cluster ID`,
  Gender = Gender,
  Foreign = ifelse(`Country of birth`=="SPAIN", "No", "YES"),
  Diabetes = Diabetes,
```

```
  HIV = `HIV infected`) %>%
  filter(!is.na(Gender), !is.na(Foreign), !is.na(Diabetes), !is.na(HIV))

# subset clusters having >2 members
nm <- as.data.frame(table(tb_valencia$Cluster)) %>% filter(Freq > 2) %>% pull(Var1)
tb_valencia <- tb_valencia %>% filter(Cluster %in% nm)

# download location data from a repo
url <- paste("https://raw.githubusercontent.com/ksusvita92/Genomic-Clustering/master/",
  "analysis%20scripts/location_data.csv", sep = "")
location_data <- read.csv(url)
tb_valencia <- tb_valencia %>%
  inner_join(location_data, by = "ID")
str(tb_valencia)
```

```
## tibble [533 x 8] (S3: tbl_df/tbl/data.frame)
##  $ ID       : chr [1:533] "G368m" "G401m" "G553m" "G566m" ...
##  $ Cluster  : chr [1:533] "CL001" "CL001" "CL001" "CL001" ...
##  $ Gender   : chr [1:533] "male" "female" "female" "male" ...
##  $ Foreign  : chr [1:533] "YES" "No" "No" "YES" ...
##  $ Diabetes : chr [1:533] "yes" "no" "no" "no" ...
##  $ HIV      : chr [1:533] "no" "no" "no" "no" ...
##  $ Latitude : num [1:533] 48.4 47.6 48.7 48.7 49 ...
##  $ Longitude: num [1:533] -8.93 -9.02 -8.86 -8.87 -8.74 ...
```

Transformation to pairwise data

Our pairwise logistic regression model uses pairwise data, as discussed in the paper. For each variable in the raw individual data, each pair of cases has a pairwise variable that represents the dissimilarity between the two cases' individual variables. For example, each case has latitude and longitude in the individual data, but each pair of cases has distance as a variable in the pairwise data.

To transform Valencia data into pairwise data, run:

```
dt <- tb_valencia[tb_valencia$Cluster != "unique",] #exclude unclustered cases
pairdt <- zCovariate(cluster = dt$Cluster,
                     X = dt[,3:6],
                     location = dt[,7:8],
                     id = dt$ID)
head(pairdt)
```

```
##    case to.case y   Spatial Gender Foreign Diabetes HIV
## 1 G401m   G368m 1 90.349667   DIFF    DIFF     DIFF  no
## 2 G553m   G368m 1 26.103191   DIFF    DIFF     DIFF  no
## 3 G566m   G368m 1 27.218045   male     YES     DIFF  no
## 4 G1163   G368m 1 66.364288   male     YES     DIFF  no
## 5 G1411   G368m 1  7.991447   DIFF    DIFF     DIFF  no
## 6  G201   G368m 0 34.309328   male     YES     DIFF  no
```

Notice that there are 4 new columns in the data frame above: `case`, `to.case`, `y`, and `Spatial`.

- `case`, `to.case`: vector of case id; together represent a pair.
- `y`: a binary variable; `y = 1` means if a pair is in the same cluster, `y = 0` means otherwise.
- `Spatial`: numeric vector that represents spatial distance between a pair.

Each categorical variable will have one additional level, called `DIFF` which indicates if a pair has different values. To see more details about the function above, run `?zCovariate`.

Pairwise Logistic Regression

There are two logistic regression models introduced in this package: the multinomial logistic regression (MLR) and the pairwise logistic regression (PLR). The difference between the two models lies on the data type used to train the model, and the response variable. MLR uses individual-level data and the response variable is the index of the cases' clusters, whereas PLR uses pairwise-level data and binary response variable indicating whether two cases are in the same cluster.

Fit PLR model

PLR perform regression analysis which maps pairwise predictors to a binary response variable. Suppose we want to fit predictors `sex`, `foreign`, `dx_data`, and `latitude` and `longitude`.

```
fit_plr <- plr(formula = Cluster ~ Gender+Foreign+Diabetes+HIV+Latitude+Longitude,
               data = dt)
summary(fit_plr)
```

```
##
## Call:
## glm(formula = mod, family = binomial(), data = traindata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.4494  -0.2584  -0.1947  -0.1153   3.9147
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.397422   0.253876 -13.382  < 2e-16 ***
## Genderfemale  0.045867   0.235045   0.195 0.845283
## Gendermale    0.023765   0.133781   0.178 0.859006
## ForeignNo     0.561727   0.148603   3.780 0.000157 ***
## ForeignYES    1.057671   0.235614   4.489 7.16e-06 ***
## Diabetesno    0.080251   0.153546   0.523 0.601219
## Diabetesyes  -0.309873   0.600558  -0.516 0.605871
## HIVno        -0.012660   0.184637  -0.069 0.945334
## HIVyes       -0.240448   1.031831  -0.233 0.815738
## Spatial      -0.018604   0.001893  -9.827  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2499.0  on 11324  degrees of freedom
## Residual deviance: 2333.8  on 11315  degrees of freedom
## AIC: 2353.8
##
## Number of Fisher Scoring iterations: 7
```

`summary()` function provides the estimated regression coefficients, together with the Wald test. An important note for the users is if ones want to fit spatial distance between two cases, ones must name the geographical

location as `latitude` (or `lat`, `Lat`, `Latitude`), and `longitude` (or `long`, `Long`, `Longitude`).

Predict if two cases are in the same cluster

Suppose there are a collection of new cases for which we have no information on their true clusters, and we want to predict the probability of a pair belonging to the same cluster. To illustrate this, let us split the `tb_valencia` data into two: one is to train the PLR model and the other is to test the model.

To make sure we split the data such that there is at least one case per cluster in each data split, use `createDataPartition()` from `caret` package. In this example, we set 60% proportion of the data to be in the training set.

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(12345)
id <- createDataPartition(dt$Cluster, p = .6, list = F)
traindt <- dt[id,]
testdt <- dt[-id,]
```

Use `plr()` function to fit PLR model on `traindt`, and run the following to get the estimated probability of the response variable:

```
fit_plr <- plr(formula = Cluster ~ Gender+Foreign+Diabetes+HIV+Latitude+Longitude,
               data = traindt)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred_plr <- predict(obj = fit_plr,
                    newdata = testdt,
                    case.id = testdt$ID) #case.id can be NULL
```

The function `predict.plr()` (or `predict()`) returns a data frame which contains a vector of the predicted probability of a pair of cases are in the same cluster, and its standard error.

Finding the optimum threshold

As in any binary classification problem where a threshold must be selected, `lr2cluster` provides a function to find an optimum threshold which depends on how the user values the cost of false positive errors (saying two cases are in the same cluster when they are not) versus false negative errors (saying two cases are not in the same cluster when they are).

The function `optThreshold()` requires a vector of the true response variable and its prediction. The later is obtained from `pred_plr$y`, and to obtain the former, use `zCovariate()` function on `testdt`.

```
# get the true response variable of testdt data
tr_resp <- zCovariate(cluster = testdt$Cluster)$y
opt_threshold <- optThreshold(response = tr_resp,
                              prediction = pred_plr$y,
                              cost.ratio = 30) #default is 1
```

```
## Setting levels: control = 0, case = 1
```

```
opt_threshold
```

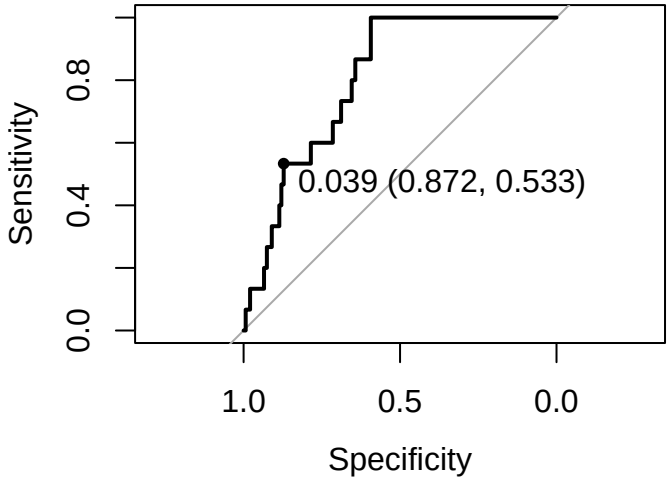
```
##
##  -- The optimum threshold using generalized Youden's index --
## ======================================================================
##
##   threshold specificity sensitivity    accuracy         auc
##  0.03939362  0.87190083  0.53333333  0.86775510  0.80352617
```

`optThreshold()` returns a list of values, such as:

- `threshold`: the optimum threshold obtained, given `cost.ratio`.
- `specificity`, `sensitivity`: the true negative rate and the true positive rate, evaluated at the optimum threshold.
- `accuracy`: accuracy of the prediction.
- 'roc": an object of class "roc".

To plot the ROC, you can run:

```
plot(opt_threshold)
```



roc-1.pdf

which also shows the position of the optimum threshold given `cost.ratio`.

Cluster Assignment

The previous section shows how to predict if two new cases belong to the same cluster. However, one may wonder if those new cases belong to one of the known clusters in the training set. This section shows how to find the most probable clusters a new case can be assigned to based on some scores called *cluster scores*. These scores, which are on a scale between 0 to 1, represent how likely a new case belong to a given cluster.

Finding probable clusters a case most likely to belong to

For a given new case, cluster assignment can be done by setting a threshold, and selecting all clusters for which the cluster score exceeds the threshold. Alternatively, one can choose the K clusters with the highest cluster score for the case. The following code does the first if a threshold if provided, and otherwise assigns a case to the best K clusters.

To do this task, we will use `traindt` and `testdt` again, so that we can test the accuracy of our assignments. We will use the same variables as the previous section as well.

```
assgn_plr <- clusterPLR(formula = Cluster ~ Gender+Foreign+Diabetes+HIV+Latitude+Longitude,
                        data = traindt,
                        newdata = testdt,
                        threshold = NULL,
                        nbest = 3)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
assgn_plr
```

```
##
##  -- Cluster assignment using pairwise logistic regression --
## ================================================================
##
## Choosing the best 3 clusters.
## Showing only the first six cases.
##
##   rank_1 rank_2 rank_3
## 1  CL021  CL009  CL072
## 2  CL020  CL072  CL045
## 3  CL009  CL021  CL016
## 4  CL007  CL063  CL002
## 5  CL011  CL001  CL034
## 6  CL011  CL001  CL034
```

The code above is to find the best $K = 3$ clusters a case can be assigned to using PLR model. If ones want to compare the method using random assignment or MLR model, run `clusterRandom()` or `clusterMLR()`.

To obtain the most probable clusters for all new cases, run:

```
getbest(assgn_plr) #or
assgn_plr$best_cluster
```

To obtain the cluster scores, run:

```
getclusterScore(assgn_plr) #or
assgn_plr$cluster_score
```

See `?clusterPLR`, `?clusterRandom`, and `?clusterMLR` for more details.

Compute the accuracy of assigning cases to their correct clusters

The accuracy in this context is the fraction of new cases whose true clusters is in the best K clusters predicted. For example, suppose that a case's true cluster is "Cluster A". If "Cluster A" is predicted as one of the K clusters, then the method will consider the assignment correct.

To compute the accuracy for predicting the new cases assignment, run

```
acc(obj = assgn_plr, true.cluster = testdt$Cluster)
```

Other Applications

Cases to be sequenced next

Instead of predicting clusters a case can be assigned to, we can turn this relationship around, and ask, for a given cluster, which unassigned cases are most likely to belong to it. This could be used, for example, to decide which of several unsequenced cases should be sequenced next, if we are interested in identifying all the cases that belong to a particular cluster.

To do this task, run

```
next_case <- case2sequence(obj = assgn_plr,
                           case.id = testdt$ID,
                           nbest = 3)
next_case
```

```
##
##  -- Predict cases to be sequenced next given clusters --
## ============================================================
##
## Choosing the best 3 cases to be sequenced next.
## Showing only few cases.
##
##   cluster priority_1 priority_2 priority_3
## 1   CL001      G1939   G788FE29       G201
## 2   CL002       G249       G932       G108
## 3   CL003      G1089      G1842      G307m
## 4   CL004       G108      G1653      G1786
## 5   CL007       G932       G249      G882m
## 6   CL008      G693m      G1440       G108
```

The above code returns the suggested 3 cases to be chosen given a cluster based on their cluster score's rank. If one wants to find the K best cases, change $nbest = K$ or, one can also provide a threshold which serves as a cut-off to any cases with lower score.

To obtain the best cases for all clusters, run

```
getbest(next_case) #or
next_case$best_cases
```

To obtain the accuracy on this task, run

```
acc(obj = next_case, true.cluster = testdt$Cluster)
```

Estimate a cluster's true size

Suppose that we have a cluster of interest C with some cases in it, and a collection of unassigned new cases. We can estimate the total number of new cases that would get assigned to C, and therefore estimate cluster C's true size.

```
clusterSize(obj = assgn_plr, rho = 0)
```

The argument `rho` in `clusterSize()` represents the probability that a case does not belong to any of any given clusters.

Closing Remarks

`lr2cluster` is a tool to assign newly identified cases of an infectious disease to existing transmission clusters using several data streams. The application is extended to also, for example, predict which new cases to be sequenced next, given a cluster, and estimate a cluster's true size.

For general questions and bug reports, please send a message to ksusvita@gmail.com.