QAD-Package

Konrad Medicus, Fabian Oberreiter, Tobias Hilgart

July 5, 2019

1 Introduction

The QAD-*'Quantification of Asymmetric Dependence'* package introduces a copula-based dependency measure capable of detecting and depicting asymmetry.
Dependency measures such as the various correlations are not applicable to measure asymmetric dependence. Using copulae, one can 'measure' the distance of a given copula to independence (dependency measure ζ_1; Trutschnig, 2011) and fulfill all requirements for an asymmetric dependency measure, such as invariant under scale changes, 1 iff Y is a function of X, 0 iff independent etc.
However, when one works with samples – observations $(x_1, y_1), \ldots, (x_n, y_n)$ – the true distribution they are drawn from is generally not known, and neither is the underlying copula.
Thus there is a need to find a 'good estimate' for said copula in the sense, that it approximates the dependency measure well.
However, in said measure, the 'known' approximation called *empirical copula* does not provide a good estimate. This is why the package introduces a 'smooth aggregation' of the empirical copula, called *empirical checkerboard copula*, that does provide a good estimate and enables the user to approximate the asymmetric dependence from a sample by computing it.

2 Background

The following is supposed to give a very general idea of the mathematics used behind the QAD-package and is not detailing the construction of the underlying objects and functions.

A (two dimensional) copula is a distribution function from $[0,1]^2 \to [0,1]$ which has uniform marginal distributions. Sklar's theorem states that for any pair of random variables, there exists a copula that can map the values of the individual (marginal) distributions back to the value of the joint distribution. Thus, copulae connect the marginal distributions to the joint distribution. Similarly to the empirical distribution function, we can approximate the underlying copula of a bivariate sample with the empirical copula. As a mere aggregation does not result in some desired convergence properties, the empirical checkerboard copula (introduced in Griessenberger, 2018) is computed, which does satisfy these properties. This empirical checkerboard copula is then used to calculate a 'distance' (the D_1-metric; introduced in Trutschnig, 2011) to the product copula, which is the copula corresponding to two independent random variables, and thereby the dependence is measured. Due to the computation process not being symmetrical, the two dependence values do not have to be the equal – there can be an asymmetry in the data – the difference of these values is the asymmetry measure.

3 qad

The qad-function does exactly what has been described above: From a given data frame containing a bivariate sample, one column per variable, it calculates the empirical checkerboard copula and the dependency measure, thus quantifying the asymmetric dependence of two random variables X and Y.
In the example below, we calculate the dependence between a sample drawn from uniformly distributed random variable X and the sine of that sample, which we call Y. We expect Y to be strongly dependent on X and less so the other way around. This is because Y is directly computed from X, while computing X from Y is not possible in this case.

3.1 Example I

```
n = 300
x = runif(n,0,30)
y = sin(x)
df = data.frame(x,y)
model = qad(df, print = FALSE, permutation = TRUE)
model$results
```

```
##                       coef p.values
## 1         q(x1,x2) 0.7263205     0.00
## 2         q(x2,x1) 0.2269512     0.23
## 3 mean.dependence 0.4766358     0.00
## 4 asymmetry       0.4993693     0.00
```

Indeed, we see that $q(x1, x2)$, measuring how dependent Y is on X, is much higher than $q(x2, x1)$. This is a simple example of two asymmetrically dependet random variables.

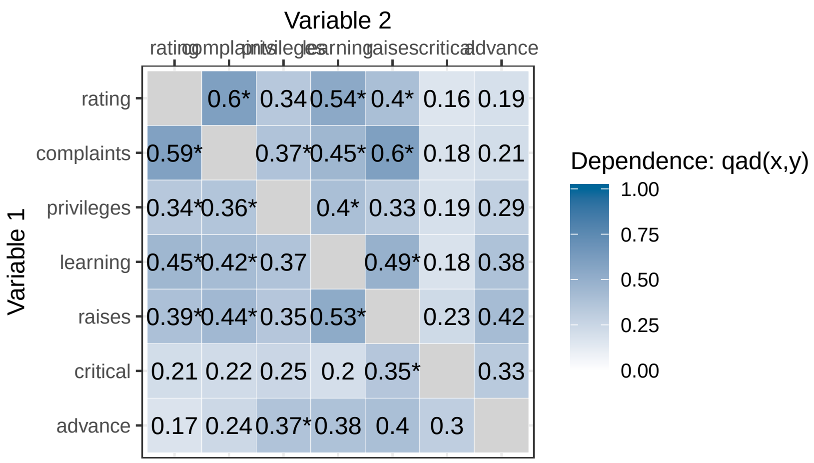
A permutated p-test can be performed to test for independence, as has been done in the example above. In this case, with a significance level of 5%, we have to reject the null hypothesis of independence for $q(x1, x2)$, but cannot do so for $q(x2, x1)$.
Such a permutated p-test is computed in the following way: From the original two

samples $x_1, \dots, x_n$ and $y_1, \dots, y_n$, we take the doubled sample $(x_1, y_1), \dots, (x_n, y_n), (y_1, x_1), \dots (y_n, x_n)$. From it we draw n observations, calculate the qad value and check wether the calculated value is at least as big as the one of the original sample. We repeat this a number R amount of times (R can be set with the argument nperm) and the fraction of runs where this is the case is our p-value.
To view the results of the qad-function, one can use the summary function on a qad-object. With pairwise.qad, one can calculate the qad-value for each pair of columns in a given data frame. The function heatmap.qad visualizes these results nicely.

3.2 Example II

This small example illustrates how pairwise.qad can be used to look for pairwise dependencies in a multivariate sample. We use the attitude data set, which shows the proportion of favourable responses to each of 7 questions for employees in 30 departments. We calculate the qad values and corresponding p-values. Significant result are marked with * in the heatmap.

```
#load attitude dataset
att = attitude
model = pairwise.qad(att, permutation = TRUE)
heatmap.qad(model, significance = TRUE)
```



4 cci

cci (conditional confidence interval) provides a confidence interval, conditional on the sample size, for the qad-measure of two independent random variables. Thus it can be used to test for independence. The default significance level is 95% and one-sided as well as two-sided tests can be performed.

4.1 Example III

The following code illustrates how a one sided hyptothesis test can be done, using cci. Here we calculate our qad-object and check wether the value of the dependency meassure is within the bounds calculated by cci. If it is not, we have to reject the hypothesis of independence.

```
c = cci(n, alternative = "one.sided")

x = runif(n,0,20)
y = sin(x)
df = data.frame(x,y)
model = qad(df,print=FALSE)

if(coef(model, select = 'q(x1,x2)') < c[2]){
 print('Accept H0')
}else{
 print('Reject H0')
}

## [1] "Reject H0"
```

5 Additional Functionalty

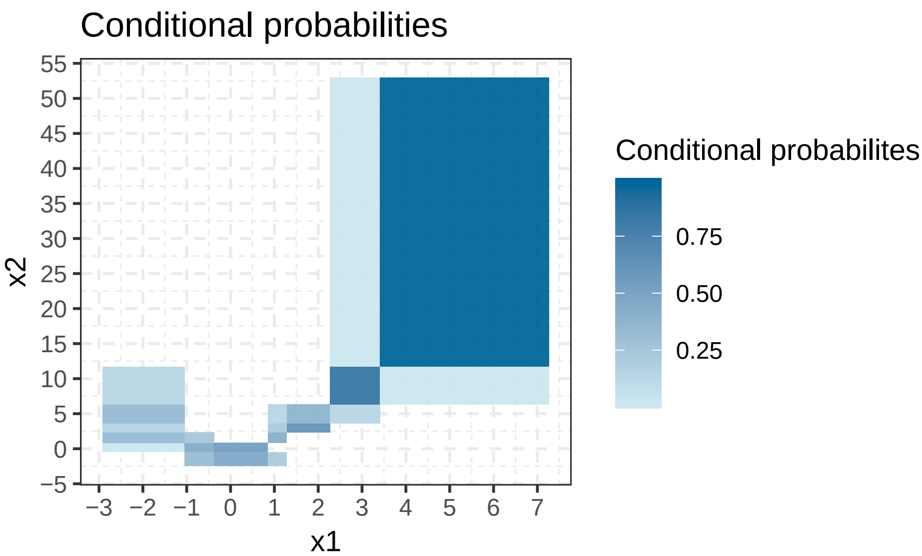
If we want to calculate just the empirical copula (smoothing=FALSE) or the checkerboard aggregation (smoothing=TRUE), we can do so by plugging a data frame with a bivariate sample into the emp_ c_ copula()-function. The resulting mass/density matrix can be plotted using plot_ density.
Alternatively, calling the plot-function on a qad-object directly plots its mass matrix (coming from the checkerboard aggregation). We can also add the sample points and retransform them from the copula into the original data setting.
Using the conditional probabilities, with the predict command on a qad-object, one gets the probabilities to land in certain intervals, given either certain x or y values.

5.1 Example IV

In this example we call plot() on a qad model. This visualizes the empirical checkerboard copula used to calculate the dependency meassure. The values of the blue rectangles in the plot give us an estimation of the conditional probability resulting in a value in the y-range of the rectangle, given that the x-value lies in the x-range.

```
x = rnorm(50,1,2)
y = x^2 + rnorm(50,0,1)
```

```
df = data.frame(x,y)
model = qad(df,print=FALSE)
plot(model, copula = FALSE)
```



5.2 Example V

The next example shows the call and output of the predict function. It shows the conditional probabilities to end up with y-value in the listed intervals for all the given values (-1 to 2).

```
predict(model, values = c(-1,0,1,2), conditioned = "x1",
        nr_intervals = 5)

##    (-2.46,-0.1] (-0.1,1.85] (1.85,3.98] (3.98,10.22] (10.22,52.96]
## -1        0.496       0.456       0.048        0.000             0
## 0         0.676       0.324       0.000        0.000             0
## 1         0.220       0.336       0.332        0.112             0
## 2         0.000       0.000       0.696        0.304             0
```

References

https://arxiv.org/ftp/arxiv/papers/1902/1902.00203.pdf
https://cran.r-project.org/web/packages/qad/qad.pdf
http://www.trutschnig.net/slides_qad_handout.pdf
http://www.trutschnig.net/qad_maths.pdf