C-JAMP: Copula-based Joint Analysis of Multiple Phenotypes

Stefan Konigorski

April 24, 2018

Overview

The general goal of C-JAMP (Copula-based Joint Analysis of Multiple Phenotypes) is to jointly model two (or more) traits Y_1, Y_2 of a phenotype conditional on covariates using copula functions, in order to estimate and test the association of the covariates with either trait.

Copulas are functions used to construct a joint distribution by combining the marginal distributions with a dependence structure (Joe 1997, Nelsen 2006). In contrast to multivariate linear regression models, which model the linear dependence of two random variables coming from a bivariate normal distribution with a certain Pearson's correlation, more general dependencies can be flexibly investigated using copula functions. This can be helpful when the conditional mean of Y_i given Y_j is not linear in Y_j. Additional properties of copula models are that they allow an investigation of the dependence structure between the phenotypes separately from the marginal distributions. Also, the marginal distributions can come from different families and not necessarily from the normal distribution.

Copula models can be used to make inference about the marginal parameters and to identify markers associated with one or multiple phenotypes. Through the joint modeling of multiple traits, the power of the association test with a given trait can be increased, also if the markers are only associated with one trait. This is of special interest for genetic association studies, and in particular for the analysis of rare genetic variants, where the low power of association tests is one of the main challenges in empirical studies. The potential of C-JAMP to increase the power of association tests is supported by the results of empirical applications of C-JAMP in real-data analyses (Konigorski, Yilmaz & Bull, 2014; Konigorski, Yilmaz, Pischon, 2016).

The functions in this package implement C-JAMP for fitting joint models based on the Clayton and 2-parameter copula, of two normally-distributed traits Y_1, Y_2 conditional on multiple predictors, and allow obtaining coefficient and standard error estimates of the copula parameters (modeling the dependence between Y_1, Y_2) and marginal parameters (modeling the effect of the predictors on Y_1 and on Y_2), as well as performing Wald-type hypothesis tests of the marginal parameters. Summary functions provide a user-friendly output. In addition, likelihood-ratio tests for testing nested copula models are available. Furthermore, functions are available to generate genetic data, and phenotypic data from bivariate normal distrutions and bivariate distributions based on copula functions. Finally, for genetic association analyses, functions are available to calculate the amount of phenotypic variance that can be explained by the genetic markers.

Background on copula functions

In more detail, a d-dimensional copula can be defined as a function $C : [0,1]^d \rightarrow [0,1]$, which is a joint cumulative distribution function with uniform marginal cumulative distribution functions. Hence, C is the distribution of a multivariate random vector, and can be considered independent of the margins. For $Y_1, ..., Y_d$ and a covariate vector x, the joint distribution F of $Y_1, ..., Y_d$, conditional on x, can be constructed by combining the marginal distributions of $Y_1, ..., Y_d$, $F_1, ..., F_d$, conditional on x, using a copula function C_ψ with dependence parameter(s) ψ:

$$F(Y_1, ..., Y_d|x) = C_\psi(F_1(Y_1|x), ..., F_d(Y_d|x)).$$

For continuous marginal distributions, the copula C_ψ is unique and the multivariate distribution can be constructed from the margins in a uniquely defined way (Sklar, 1959).

Popular copula functions include the Clayton family

$$C_\phi(u_1, ..., u_d, \phi) = \left(\sum_{i=1}^{d} u_i^{-\phi} - (d-1) \right)^{-1/\phi}$$

with $\phi > 0$, and the Gumbel-Hougaard family

$$C_\theta(u_1, ..., u_d, \theta) = exp\left(- \left[\sum_{i=1}^{d} (-log(u_i))^\theta \right]^{1/\theta} \right)$$

with $\theta > 1$. A third family which includes both Clayton and Gumbel-Hougaard copula (for $\theta = 1$ and $\phi \to 0$) is the 2-paramater copula family

$$C_\psi(u_1, ..., u_d, \phi, \theta) = \left(\left[\sum_{i=1}^{d} \left(u_i^{-\phi} - 1 \right)^\theta \right]^{1/\theta} + 1 \right)^{-1/\phi}$$

with $0 \leq u_1, u_2 \leq 1$, and the copula parameters $\psi = (\phi, \theta), \phi > 0, \theta \geq 1$, which allows a flexible modeling of both the lower- and upper-tail dependence.

For the 2-parameter copula family, Kendall's tau can be derived as (Joe, 1997)

$$\tau = 1 - \frac{2}{\theta(\phi + 2)}.$$

Additional dependence parameters of interest are the lower and upper tail dependence measures λ_L, λ_U, which explain the amount of dependence between extreme values. For the 2-parameter copula family, these dependence measures become (Joe, 1997)

$$\lambda_L = 2^{-1/\theta\phi}, \lambda_U = 2 - 2^{1/\theta}.$$

Data Y_1, Y_2 can be sampled from the Clayton copula (with standard normal margins) using the `generate_clayton_copula()` function. This uses the following steps to generate standard normal Y_1, Y_2 from the Clayton copula $C_\phi(\Phi(Y_1), \Phi(Y_2)) = C_\phi(U_1, U_2)$, where Φ is the standard normal cumulative distribution function:

1. Generate uniform random variables $U_1, \tilde{U}_2 \sim Unif(0, 1)$.
2. Generate Y_1 from the inverse standard normal distribution of U_1, $Y_1 := \Phi^{-1}(U_1)$.
3. In order to obtain the second variable Y_2, use the conditional distribution of U_2 given U_1, set $\tilde{U}_2 = \frac{\partial C(U_1, U_2)}{\partial U_1}$ and solve for U_2. This yields $U_2 = \left(1 - U_1^{-\phi} \cdot \left(1 - \tilde{U}_2^{-\phi/(1+\phi)}\right)\right)^{-1/\phi}$.
4. Generate Y_2 from the inverse standard normal distribution of U_2, $Y_2 := \Phi^{-1}(U_2)$.

```
# Generate phenotype data from the Clayton copula:
dat1a <- generate_clayton_copula(n = 1000, phi = 0.5)
```

```
## [1] "Kendall's tau between Y1, Y2 = 0.2"
```

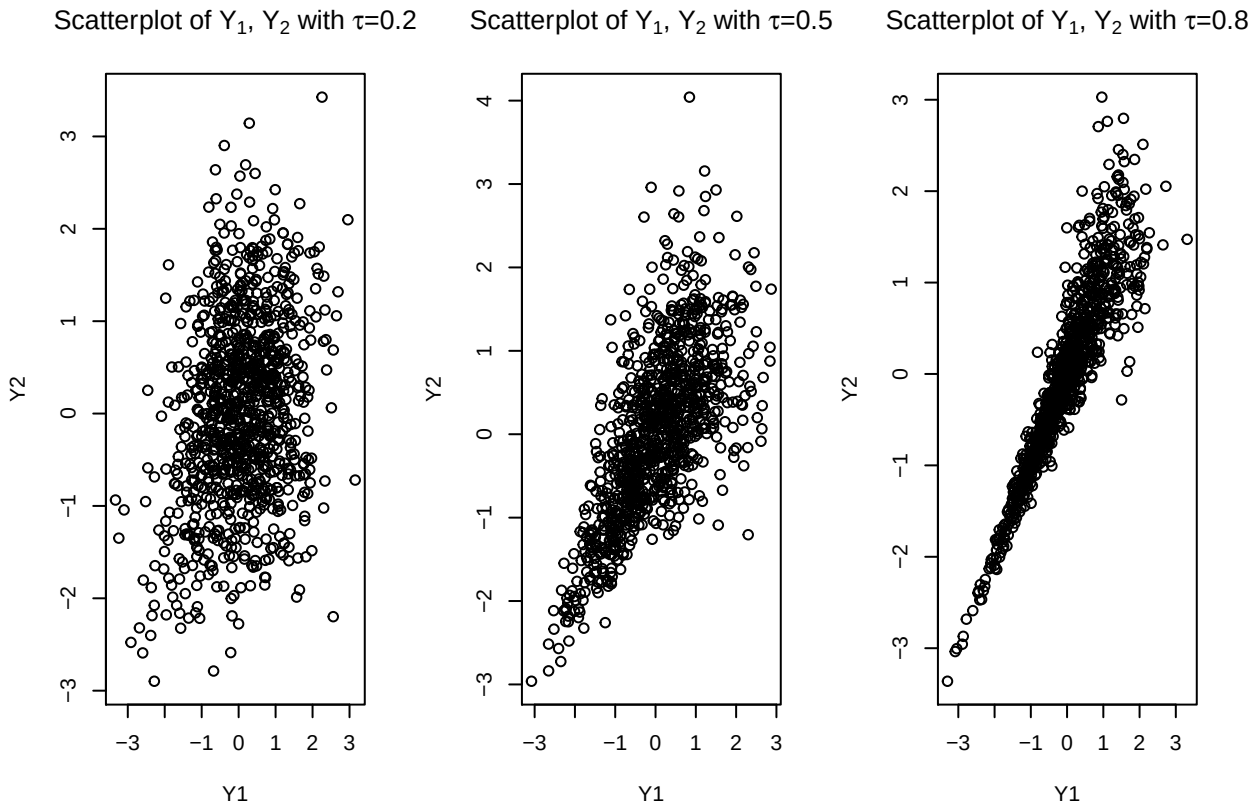
```
dat1b <- generate_clayton_copula(n = 1000, phi = 2)
```

```
## [1] "Kendall's tau between Y1, Y2 = 0.5"
```

```
dat1c <- generate_clayton_copula(n = 1000, phi = 8)
```

```
## [1] "Kendall's tau between Y1, Y2 = 0.8"
```

```
par(mfrow = c(1, 3))
plot(dat1a$Y1, dat1a$Y2, xlab = "Y1", ylab = "Y2",
     main = expression(paste("Scatterplot of ", Y[1], ", ", Y[2], " with ", tau, "=0.2")))
plot(dat1b$Y1, dat1b$Y2, xlab = "Y1", ylab = "Y2",
     main = expression(paste("Scatterplot of ", Y[1], ", ", Y[2], " with ", tau, "=0.5")))
plot(dat1c$Y1, dat1c$Y2, xlab = "Y1", ylab = "Y2",
     main = expression(paste("Scatterplot of ", Y[1], ", ", Y[2], " with ", tau, "=0.8")))
```



Data generation functions

In order to generate sample data for simulation studies and to illustrate C-JAMP, different functions are provided. Firstly, the functions `generate_singleton_data()`, `generate_doubleton_data()` and `generate_genodata()` can be used to generate genetic data in the form of single nucleotide variants (SNVs). `generate_singleton_data()` generates singletons (i.e., SNVs with one observed minor allele); `generate_doubleton_data()` generates doubletons (i.e., SNVs with two observed minor alleles), and the function `generate_genodata()` generates n observations of k SNVs with random minor allele frequencies. The minor allele frequencies can be computed using the function `compute_MAF()`.

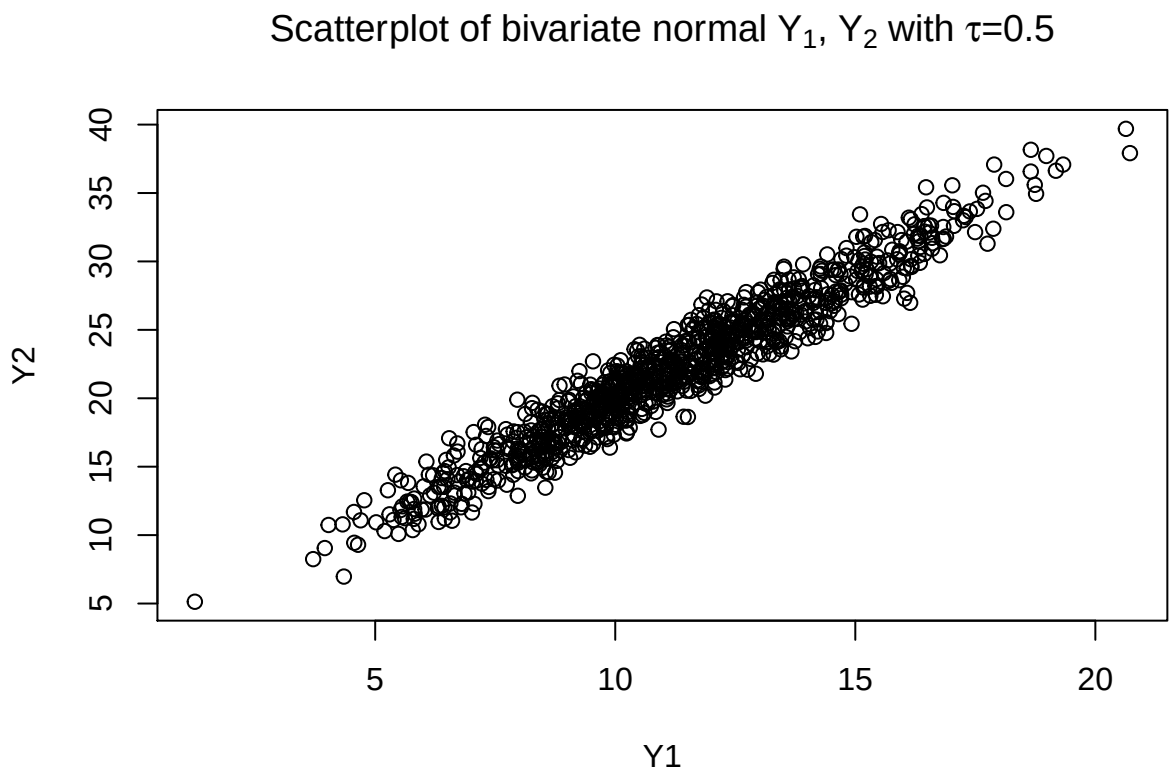
Next, `generate_phenodata_1_simple()`, `generate_phenodata_1()`, `generate_phenodata_2_bvn()` and `generate_phenodata_2_copula()` can be used to generate phenotype data based on SNV input data, covariates, and a specification of the covariate effect sizes. Here, the functions `generate_phenodata_1_simple()` and `generate_phenodata_1()` generate one normally-distributed or binary phenotype Y conditional on

provided SNVs and two generated covariates X_1, X_2. The functions `generate_phenodata_2_bvn()` and `generate_phenodata_2_copula()` generate two phenotypes Y_1, Y_2 with dependence Kendall's τ conditional on the provided SNVs and covariates X_1, X_2 from the bivariate normal distribution or the Clayton copula function with standard normal marginal distributions. `generate_phenodata_2_copula()` is using the function `generate_clayton_copula()` described above.

```
# Generate genetic data:
genodata <- generate_genodata(n_SNV = 20, n_ind = 1000)
compute_MAF(genodata)
```

```
##  SNV1  SNV2  SNV3  SNV4  SNV5  SNV6  SNV7  SNV8  SNV9 SNV10 SNV11 SNV12
## 0.451 0.427 0.434 0.031 0.091 0.240 0.072 0.223 0.402 0.309 0.338 0.325
## SNV13 SNV14 SNV15 SNV16 SNV17 SNV18 SNV19 SNV20
## 0.230 0.246 0.364 0.431 0.094 0.168 0.458 0.145
```

```
# Generate phenotype data from the bivariate normal distribution given covariates:
phenodata_bvn <- generate_phenodata_2_bvn(genodata = genodata,
                                          tau = 0.5, b1 = 1, b2 = 2)
plot(phenodata_bvn$Y1, phenodata_bvn$Y2, xlab = "Y1", ylab = "Y2",
     main = expression(paste("Scatterplot of bivariate normal ", Y[1], ", ",
                             Y[2], " with ", tau, "=0.5")))
```

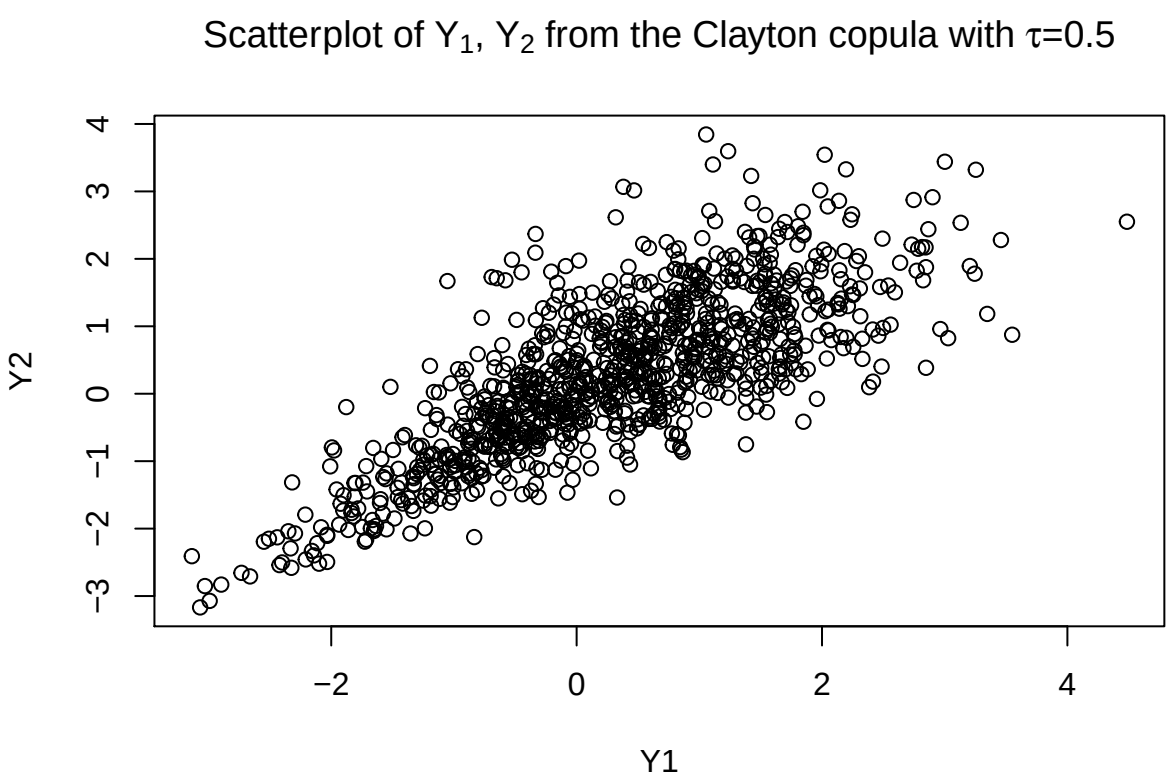



```
# Generate phenotype data from the Clayton copula given covariates:
phenodata <- generate_phenodata_2_copula(genodata = genodata$SNV1,
                                         MAF_cutoff = 1, prop_causal = 1,
                                         tau = 0.5, b1 = 0.3, b2 = 0.3)
```

```
## [1] "Kendall's tau between Y1, Y2 = 0.5"
```

```
## [1] "Kendall's tau between Y1, Y2 = 0.5"
```

```
plot(phenodata$Y1, phenodata$Y2, xlab = "Y1", ylab = "Y2",
     main = expression(paste("Scatterplot of ", Y[1], ", ", Y[2],
                             " from the Clayton copula with ", tau, "=0.5")))
```



For the analysis of genetic associations with the phenotypes, the amount of variability in the phenotypes that can be explained by the SNVs might be of interest. This can be computed based on four different approaches using the function `compute_expl_var()` (Laird & Lange, 2011), which is described in more detail in the help pages of the function.

```
compute_expl_var(genodata = genodata, phenodata = phenodata$Y1,
                 type = c("Rsquared_unadj", "Rsquared_adj",
                          "MAF_based", "MAF_based_Y_adjusted"),
                 causal_idx = rep(TRUE,20), effect_causal =
                   c(0.3 * abs(log10(compute_MAF(genodata$SNV1))), rep(0,19)))
```

```
## $Rsquared_unadj
## [1] 0.0290968
##
## $Rsquared_adj
## [1] 0.009262208
##
## $MAF_based
## [1] 0.005330038
##
## $MAF_based_Y_adjusted
## [1] 0.003847294
```

C-JAMP: Copula-based joint analysis of multiple phenotypes

C-JAMP is implemented for the joint analysis of two phenotypes Y_1, Y_2 conditional on one or multiple predictors x with linear marginal models. Functions are available to fit the joint model

$$F(Y_1, Y_2|x) = C_\psi(F_1(Y_1|x), F_2(Y_2|x))$$

for the Clayton and 2-parameter copula, with marginal models

$$Y_1 = \alpha^T X + \epsilon_1, \epsilon_1 \sim N(0, \sigma_1^2)$$
$$Y_2 = \beta^T X + \epsilon_2, \epsilon_2 \sim N(0, \sigma_2^2).$$

Maximum likelihood estimates for all parameters $(\psi, \alpha, \beta, \sigma_1, \sigma_2)$ can be obtained using the function `get_estimates_naive()`, which is based on the ´lm()´ function.

```
predictors <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2, SNV = genodata$SNV1)
get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2, predictors_Y1 = predictors,
                    predictors_Y2 = predictors, copula_param = "both")
```

```
##          log_phi log_theta_minus1     Y1_log_sigma     Y2_log_sigma
##      1.003949479      0.310802299      0.024392578      0.023707242
##   Y1_(Intercept)            Y1_X1            Y1_X2           Y1_SNV
##      0.006544615      0.502382799      0.566976993      0.015620415
##   Y2_(Intercept)            Y2_X1            Y2_X2           Y2_SNV
##      0.051589400      0.494065047      0.520149266     -0.024657377
```

For these or any other parameter estimates, the log-likelihood (or rather the minus log-likelihood) of the copula model can be computed with the function `minusloglik()` for the Clayton and 2-parameter copula:

```
predictors <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2, genodata[, 1:5])
estimates <- get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                 predictors_Y1 = predictors,
                                 predictors_Y2 = predictors, copula_param = "both")
minusloglik(Y1 = phenodata$Y1, Y2 = phenodata$Y2, predictors_Y1 = predictors,
            predictors_Y2 = predictors, parameters = estimates, copula = "2param")
```

```
## [1] 3483.913
```

It should be noted that the `minusloglik()` function (and the `cjamp()`, `cjamp_loop()` functions below) assume quantitative predictors and use an additive model. That means that for the analysis of categorical predictors with more than 2 levels, dummy variables have to be created beforehand. Accordingly, if single nucleotide variants (SNVs) are included as predictors, the computation is based on an additive genetic model if SNVs are provided as 0-1-2 genotypes and on a dominant model if SNVs are provided as 0-1 genotypes.

The `lrt_copula()` can be used to test the model fit of different nested copula models with the same marginal models. For this, likelihood ratio tests are performed. In more detail, to test the fit of the 1-parameter Clayton copula model fit versus the 2-parameter copula, i.e. $H_0 : \phi \to 0$, the likelihood ratio test statistic

$$logLR = 2 \cdot \left(L\left(\hat{\alpha}, \hat{\beta}, \hat{\psi}\right) - L\left(\hat{\alpha}(\psi_0), \hat{\beta}(\psi_0), \hat{\psi}\right)\right)$$

can be used, where $\hat{\psi} = (\hat{\phi}, \hat{\theta})$, $\hat{\psi}_0 = \left(\hat{\phi}(\theta = 1), 1\right)$, see Yilmaz & Lawless (2011). Here, $\hat{\alpha}(\psi_0)$ and $\hat{\beta}(\psi_0)$ are the maximum likelihood estimates of α and β under the null model $\psi = \psi_0$. P-values are obtained by using that $logLR$ is asymptotically distributed as $0.5 + 0.5\chi_1^2$ under the null hypothesis $\psi = \psi_0$ and under the assumption that other free parameters in ψ_0 don't lie on the boundary of the parameter space (Self & Liang, 1987).

```
# Example: Test whether 2-parameter copula model has a better
#          model fit compared to Clayton copula (no).
predictors <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2,
                         SNV = genodata$SNV1)
estimates_c <- get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                   predictors_Y1 = predictors,
                                   predictors_Y2 = predictors,
                                   copula_param = "phi")
minusloglik_Clayton <- minusloglik(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                   predictors_Y1 = predictors,
                                   predictors_Y2 = predictors,
                                   parameters = estimates_c, copula = "Clayton")
estimates_2p <- get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                    predictors_Y1 = predictors,
                                    predictors_Y2 = predictors,
                                    copula_param = "both")
minusloglik_2param <- minusloglik(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                  predictors_Y1 = predictors,
                                  predictors_Y2 = predictors,
                                  parameters = estimates_2p, copula = "2param")
lrt_copula(minusloglik_Clayton, minusloglik_2param)
```

```
## $chisq
## [1] -2102.096
##
## $pval
## [1] 0.5
```

Log-likelihood ratio tests of marginal parameters within the same copula model can be performed using the `lrt_param()` function:

```
# Example: Test marginal parameters (alternative model has better fit).
predictors_1 <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2)
estimates_1 <- get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                   predictors_Y1 = predictors_1,
                                   predictors_Y2 = predictors_1,
                                   copula = "phi")
minusloglik_1 <- minusloglik(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                             predictors_Y1 = predictors_1,
                             predictors_Y2 = predictors_1,
                             parameters = estimates_1, copula = "Clayton")
predictors_2 <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2,
                           SNV = genodata$SNV1)
estimates_2 <- get_estimates_naive(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                                   predictors_Y1 = predictors_2,
                                   predictors_Y2 = predictors_2,
                                   copula = "phi")
minusloglik_2 <- minusloglik(Y1 = phenodata$Y1, Y2 = phenodata$Y2,
                             predictors_Y1 = predictors_2,
                             predictors_Y2 = predictors_2,
                             parameters = estimates_2, copula = "Clayton")
lrt_param(minusloglik_1, minusloglik_2, df=2)
```

```
## $chisq
## [1] 2.011721
```

```
##
## $pval
## [1] 0.3657298
```

Finally, to obtain maximum likelihood estimates of the parameters $(\psi, \alpha, \beta, \sigma_1, \sigma_2)$ as well as standard error estimates under the Clayton or 2-parameter copula models, the function `cjamp()` can be used. `cjamp()` uses the `optimx()` function in the optimx package to maximize the log-likelihood function (i.e., minimize `minusloglik()`). For this, the BFGS optimization method is recommended. In order to deal with convergence problems, several checks are built-in, and different starting values for the optimization algorithm are automatically tried. Standard error estimates of the parameter estimates are obtained from the observed inverse information matrix. Finally, p-values are computed for the marginal parameter estimates from hypothesis tests of the absence of effects of each predictor on each phenotype in the marginal models.

```
# Restrict example to sample size 100 to decrease running time:
predictors <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2, genodata[, 1:3])[1:100,]
cjamp_res <- cjamp(copula = "2param", Y1 = phenodata$Y1[1:100], Y2 = phenodata$Y2[1:100],
                   predictors_Y1 = predictors, predictors_Y2 = predictors,
                   scale_var = FALSE, optim_method = "BFGS", trace = 0,
                   kkt2tol = 1E-16, SE_est = TRUE, pval_est = TRUE,
                   n_iter_max = 10)
cjamp_res
```

```
## $`Parameter point estimates`
##       Y1_sigma       Y2_sigma Y1_(Intercept)          Y1_X1          Y1_X2
##   1.016224e+00   9.929620e-01   2.423349e-01   4.350554e-01   5.247546e-01
##        Y1_SNV1        Y1_SNV2        Y1_SNV3 Y2_(Intercept)          Y2_X1
##  -8.915398e-02   6.964400e-02  -1.010249e-01   1.964235e-01   5.723007e-01
##          Y2_X2        Y2_SNV1        Y2_SNV2        Y2_SNV3            phi
##   4.568820e-01  -1.032245e-01   1.923883e-01  -1.846370e-01   2.068568e+00
##          theta            tau       lambda_l       lambda_u
##   1.000000e+00   5.084267e-01   7.152770e-01   2.556038e-07
##
## $`Parameter standard error estimates`
##       Y1_sigma       Y2_sigma Y1_(Intercept)          Y1_X1          Y1_X2
##   0.0643193062   0.0636963923   0.2411894098   0.0918430447   0.1774842600
##        Y1_SNV1        Y1_SNV2        Y1_SNV3 Y2_(Intercept)          Y2_X1
##   0.1212571750   0.1271905140   0.1272314500   0.2385265288   0.0893081485
##          Y2_X2        Y2_SNV1        Y2_SNV2        Y2_SNV3            phi
##   0.1704847569   0.1175217947   0.1220994460   0.1229165093   0.3775926157
##          theta            tau       lambda_l       lambda_u
##   0.0001078504   0.0456215604   0.0437505299   0.0001495124
##
## $`Parameter p-values`
## Y1_(Intercept)          Y1_X1          Y1_X2        Y1_SNV1        Y1_SNV2
##   3.150175e-01   2.169644e-06   3.110248e-03   4.621890e-01   5.839964e-01
##        Y1_SNV3 Y2_(Intercept)          Y2_X1          Y2_X2        Y2_SNV1
##   4.271812e-01   4.102311e-01   1.472880e-10   7.364431e-03   3.797571e-01
##        Y2_SNV2        Y2_SNV3
##   1.151021e-01   1.330626e-01
##
## $`Convergence code of optimx function`
## convcode
##        0
##
## $`Karush-Kuhn-Tucker conditions 1 and 2`
```

```
## KKT1 KKT2
## TRUE TRUE
##
## $`Maximum log-likelihood`
## maxloglik
##  244.2731
##
## attr(,"class")
## [1] "cjamp"
```

If the goal is to test a large number of predictors in the same marginal models (such as in genetic association studies), the wrapper function `cjamp_loop()` provides an easy use of the `cjamp()` function and only extracts the coefficient and standard error estimates as well as the p-values for the predictor(s) of interest, and not for all other covariates.

```
# Restrict example to sample size 100 to decrease running time:
covariates <- data.frame(X1 = phenodata$X1, X2 = phenodata$X2)[1:100,]
predictors <- genodata[1:100,1:5]
cjamp_loop_res <- cjamp_loop(copula = "Clayton", Y1 = phenodata$Y1[1:100],
                             Y2 = phenodata$Y2[1:100], predictors = predictors,
                             covariates_Y1 = covariates, covariates_Y2 = covariates,
                             scale_var = FALSE, optim_method = "BFGS", trace = 0,
                             kkt2tol = 1E-16, SE_est = TRUE, pval_est = TRUE,
                             n_iter_max = 10)
cjamp_loop_res
```

```
## $`Parameter point estimates for effects of predictors on Y1`
##        SNV1        SNV2        SNV3        SNV4        SNV5
## -0.08029167  0.07997334 -0.10558475 -0.69381126 -0.25168978
##
## $`Parameter point estimates for effects of predictors on Y2`
##       SNV1       SNV2       SNV3       SNV4       SNV5
## -0.1095230  0.1995290 -0.1810671 -0.6230700 -0.1732627
##
## $`Parameter standard error estimates for effects on Y1`
##      SNV1      SNV2      SNV3      SNV4      SNV5
## 0.1214494 0.1269342 0.1278624 0.2921368 0.2273061
##
## $`Parameter standard error estimates for effects on Y2`
##      SNV1      SNV2      SNV3      SNV4      SNV5
## 0.1214581 0.1236540 0.1247037 0.2893925 0.2250838
##
## $`Parameter p-values for effects of predictors on Y1`
##       SNV1       SNV2       SNV3       SNV4       SNV5
## 0.50854038 0.52866980 0.40893522 0.01755117 0.26817611
##
## $`Parameter p-values for effects of predictors on Y2`
##       SNV1       SNV2       SNV3       SNV4       SNV5
## 0.36719760 0.10661255 0.14650737 0.03131651 0.44143664
##
## $`Convergence code of optimx function`
## [1] 0 0 0 0 0
##
## $`Karush-Kuhn-Tucker conditions 1 and 2`
## [1] TRUE TRUE TRUE TRUE TRUE
```

```
##
## $`Maximum log-likelihood`
## [1] 246.9648 245.8566 246.2286 244.5604 246.7378
##
## attr(,"class")
## [1] "cjamp"
```

Both `cjamp()` and `cjamp_loop()` return `cjamp` objects as output, so that the `summary.cjamp()` function can be used through the generic `summary()` to provide a reader-friendly formatted output of the results.

```
# Summary of regular cjamp function
summary(cjamp_res)
```

```
## [1] "C-JAMP estimates of marginal parameters."
##                point_estimates SE_estimates      pvalues
## Y1_(Intercept)      0.24233494   0.24118941 3.150175e-01
## Y1_X1               0.43505542   0.09184304 2.169644e-06
## Y1_X2               0.52475463   0.17748426 3.110248e-03
## Y1_SNV1            -0.08915398   0.12125718 4.621890e-01
## Y1_SNV2             0.06964400   0.12719051 5.839964e-01
## Y1_SNV3            -0.10102489   0.12723145 4.271812e-01
## Y2_(Intercept)      0.19642348   0.23852653 4.102311e-01
## Y2_X1               0.57230075   0.08930815 1.472880e-10
## Y2_X2               0.45688198   0.17048476 7.364431e-03
## Y2_SNV1            -0.10322454   0.11752179 3.797571e-01
## Y2_SNV2             0.19238833   0.12209945 1.151021e-01
## Y2_SNV3            -0.18463704   0.12291651 1.330626e-01
```

```
# Summary of looped cjamp function
summary(cjamp_loop_res)
```

```
## [1] "C-JAMP estimates of marginal parameters on Y1."
##         point_estimates SE_estimates    pvalues
## Y1_SNV1     -0.08029167    0.1214494 0.50854038
## Y1_SNV2      0.07997334    0.1269342 0.52866980
## Y1_SNV3     -0.10558475    0.1278624 0.40893522
## Y1_SNV4     -0.69381126    0.2921368 0.01755117
## Y1_SNV5     -0.25168978    0.2273061 0.26817611
## [1] "C-JAMP estimates of marginal parameters on Y2."
##         point_estimates SE_estimates    pvalues
## Y2_SNV1      -0.1095230    0.1214581 0.36719760
## Y2_SNV2       0.1995290    0.1236540 0.10661255
## Y2_SNV3      -0.1810671    0.1247037 0.14650737
## Y2_SNV4      -0.6230700    0.2893925 0.03131651
## Y2_SNV5      -0.1732627    0.2250838 0.44143664
```

References

Joe H (1997). Multivariate models and multivariate dependence concepts. London: Chapman & Hall.

Konigorski S, Yilmaz YE, Bull SB (2014). Bivariate genetic association analysis of systolic and diastolic blood pressure by copula models. BMC Proc, 8(Suppl 1): S72.

Konigorski S, Yilmaz YE, Pischon T (2016). Genetic association analysis based on a joint model of gene expression and blood pressure. BMC Proc, 10(Suppl 7): 289-294.

Laird NM, Lange C (2011). The fundamentals of modern statistical genetics. New York: Springer.

Nelsen RB (2006). An introduction to copulas. Springer, New York.

Sklar A (1959). Fonctions de répartition à n dimensions et leurs marges. Publications de l'Institut de statistique de l'Université de Paris 8: 229–231.

Self GS, Liang KY (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. J Am Stat Assoc, 82: 605–610.

Yilmaz YE, Lawless JF (2011). Likelihood ratio procedures and tests of fit in parametric and semiparametric copula models with censored data. Lifetime Data Anal, 17(3): 386-408.