Pleiotropy for quantitive traits: pleio

Dan Schaid, Jason Sinnwell

23 June, 2017

Example Data

The pleio package contains a pre-made simulated dataset with multiple quantitative traits simulated from a multivariate normal distribution with common correlation structure, correlation of 0.5, and genotypes simulated based on minor allele frequency of 0.2, and assumes that traits 2 and 3 have non-zero coefficients, while all other traits are not asssociated with dose of minor allele.

Here, we load the simulated dataset and show matrix y for phenotypes and the distribution of the minor dosage in the single genotype, *geno*.

```
## load package and dataset
require(pleio)
data(pleio.qdemo)

## preview simulated data
head(y)
```

```
##              t1          t2         t3         t4         t5         t6
## [1,] -0.6795023  1.07928655 -0.3447868  0.7757427  0.2441987  0.2849724
## [2,] -0.5310806  0.06419289 -0.8963973  0.2715258 -0.4579625 -1.2279082
## [3,]  1.0244553  1.18181268  0.1977245  0.3652163  0.3497826  1.0145627
## [4,] -0.3054566 -0.49250649  1.0348276 -0.1683192  1.8157372  0.4571641
## [5,] -0.4246874 -1.57093941 -0.3505441  0.7817763 -2.4317261 -1.0474812
## [6,]  2.0213843  1.45198074  0.2430903  2.2323818  1.2128110  2.0509974
```

```
table(geno)
```

```
## geno
##   0   1   2
## 312 170  18
```

Sequential Pleiotropy Tests

The *pleio.q.sequential* function is a high-level way to perform sequential tests of the number of traits (and which traits) are associated with a genotype. The algorithm starts with testing the usual multivariate null hypthothesis that all betas are zero. If this test rejects, because the p-value is less than a user-spiecifed threshold, then allow one beta to be non-zero in order to test whether the remaining betas = 0. If the test allowing for one non-zero beta rejects, then allow two non-zero betas (testing all combinations of two non-zero betas). Continue this sequential testing until the p-value for a test is greater than the specified threshold. When the sequential testing stops, one can conclude that the final model contains the non-zero betas, while all other betas are inferred to be zero. For m traits, the sequential testing stops either when the p-value is less than the threshold, or when (m-1) traits are tested. If the p-value remains less than *pval.threshold* when testing (m-1) traits, this implies that all m traits are associated with the genotype.

Below we run two functions, *pleio.q.fit*, which performs pre-calculations on the models to be tested, and *pleio.q.sequential*, which performs the sequential pleiotropy tests on the pre-computed object from *pleio.q.fit*.

The final result lists the indices of the non-zero betas (the indices of the traits associated with a genotype), and the p-value that tests the fit of the final model. A p-value greater than the threshold is expected for the final model, showing that the final model fits the data well. For this example, the sequential approach stopped at 2 traits because the p-value is greater than the *pval.threshold* argument given of 0.05.

```
fit <- pleio.q.fit(y, geno)

test.seq <- pleio.q.sequential(fit, pval.threshold=.05)
test.seq
```

```
## $pval
## [1] 0.2744734
##
## $index.beta
## [1] 2 3
```

Equivalent Steps to Sequential Fit

The sequential steps above can be performed with more user control using *pleio.q.test*, with *count.nonzero.beta* as the number of non-zero betas for the null hypothesis. The result of *pleio.q.test* contains the global test statistic, degrees of freedom (df), p-value for testing the model, the indices of the non-zero betas in the model, and a data.frame called "tests" that contains the tests performed for the null hypothesis models (i.e., the indices of the non-zero betas and the corresponding statistic, tk, for each model). For m traits, and $k =$ *count.nonzero.beta*, there are m-choose-k models in the set that are considered in the null hypothesis, and the minimum tk test statistic over the set provides the global test statistic reported.

```
test0 <- pleio.q.test(fit, count.nonzero.beta = 0)
test0
```

```
## $stat
## [1] 37.08576
##
## $pval
## [1] 1.694389e-06
##
## $df
## [1] 6
##
## $index.nonzero.beta
## [1] 0
##
## $tests
##   index.1       tk
## 1       0 37.08576
```

```
test1 <- pleio.q.test(fit, count.nonzero.beta = 1)
test1
```

```
## $stat
## [1] 23.53879
##
## $pval
## [1] 0.000266202
##
## $df
## [1] 5
```

```
##
## $index.nonzero.beta
## [1] 3
##
## $tests
##   index.1       tk
## 1       1 32.44700
## 2       2 24.26597
## 3       3 23.53879
## 4       4 28.02627
## 5       5 37.01172
## 6       6 36.32446
```

```
test2 <- pleio.q.test(fit, count.nonzero.beta = 2)
test2
```

```
## $stat
## [1] 5.1274
##
## $pval
## [1] 0.2744734
##
## $df
## [1] 4
##
## $index.nonzero.beta
## [1] 2 3
##
## $tests
##    index.1 index.2       tk
## 1        1       2 21.43943
## 2        1       3 21.50900
## 3        1       4 20.53175
## 4        1       5 32.01780
## 5        1       6 31.14303
## 6        2       3  5.12740
## 7        2       4 17.74535
## 8        2       5 24.14999
## 9        2       6 24.26479
## 10       3       4 17.37530
## 11       3       5 23.31140
## 12       3       6 23.24654
## 13       4       5 27.59217
## 14       4       6 25.76981
## 15       5       6 36.13547
```