Permutation tests

Ken Rice
Thomas Lumley

UW Biostatistics

Seattle, June 2008

Overview

- Permutation tests

- A mean

- Smallest p-value across multiple models

- Cautionary notes

Testing

In testing a null hypothesis we need a test statistic that will have different values under the null hypothesis and the alternatives we care about (eg a relative risk of diabetes)

We then need to compute the sampling distribution of the test statistic when the null hypothesis is true. For some test statistics and some null hypotheses this can be done analytically. The p-value for the is the probability that the test statistic would be at least as extreme as we observed, if the null hypothesis is true.

A permutation test gives a simple way to compute the sampling distribution for any test statistic, under the strong null hypothesis that a set of genetic variants has absolutely no effect on the outcome.

Permutations

To estimate the sampling distribution of the test statistic we need many samples generated under the strong null hypothesis.

If the null hypothesis is true, changing the exposure would have no effect on the outcome. By randomly shuffling the exposures we can make up as many data sets as we like.
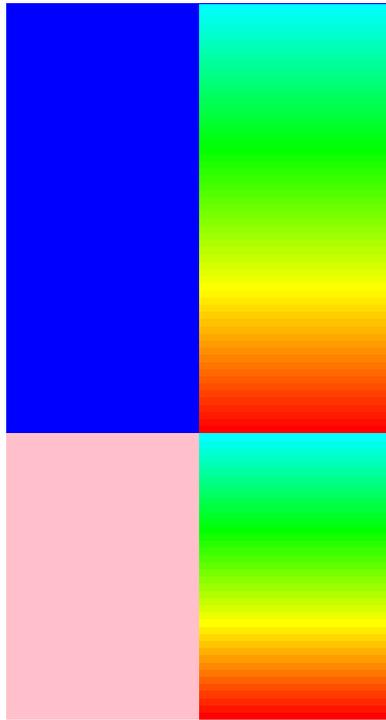
If the null hypothesis is true the shuffled data sets should look like the real data, otherwise they should look different from the real data.

The ranking of the real test statistic among the shuffled test statistics gives a p-value
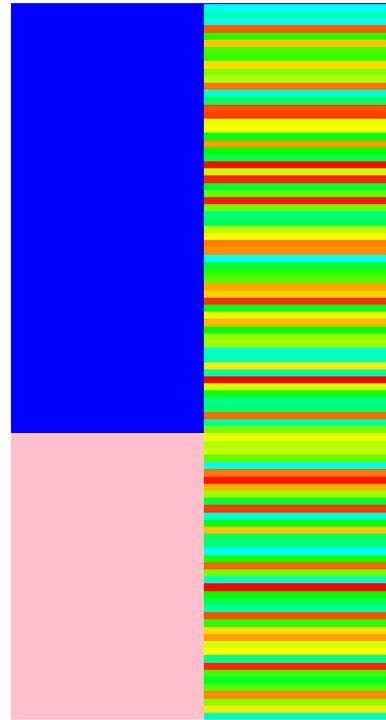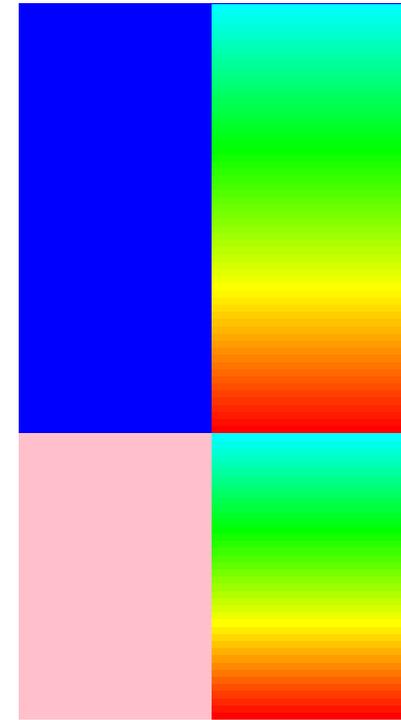
Example: null is true

Data

Shuffling outcomes

Shuffling outcomes (ordered)

gender   outcome

gender   outcome

gender   outcome

Example: null is false

Data   **Shuffling outcomes**   **Shuffling outcomes (ordered)**
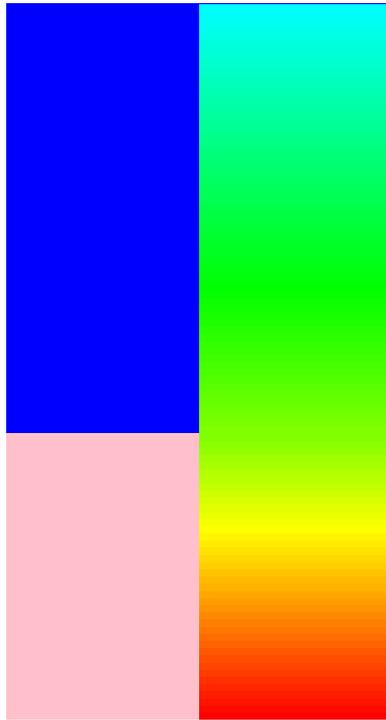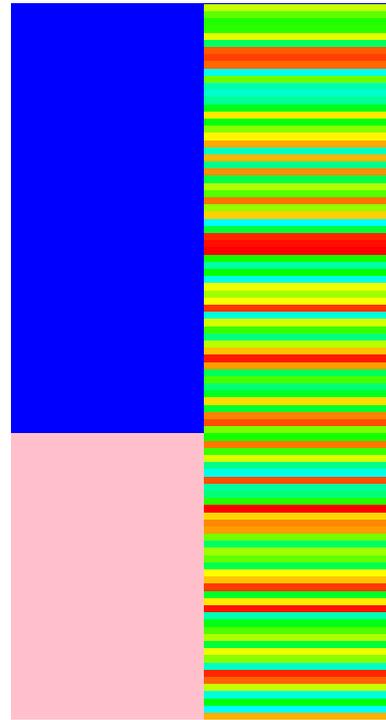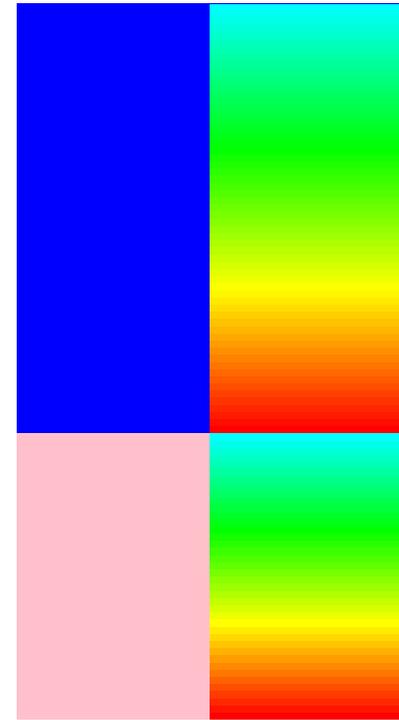
gender   outcome   gender   outcome   gender   outcome

Means

Our first example is a difference in mean outcome in a dominant model for a single SNP

```
## make up some 'true' data
carrier<-rep(c(0,1), c(100,200))
null.y<-rnorm(300)
alt.y<-rnorm(300, mean=carrier/2)
```

In this case we know from theory the distribution of a difference in means and we could just do a t-test.

We can compare the t-test results to the results of a permutation test on the mean difference.

Means: t-test

```
> t.test(null.y~carrier, var.equal=TRUE)
Two Sample t-test

data:  null.y by carrier
t = 0.5348, df = 298, p-value = 0.5932
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1740603  0.3039669
sample estimates:
mean in group 0 mean in group 1
    0.067211652     0.002258386

> t.test(alt.y~carrier, var.equal=TRUE)
Two Sample t-test

data:  alt.y by carrier
t = -5.0415, df = 298, p-value = 8.033e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8381897 -0.3675325
sample estimates:
mean in group 0 mean in group 1
    -0.1405250      0.4623361
```

Means: permutation test

```
null.diff<-mean(null.y[carrier==1])-mean(null.y[carrier==0])
alt.diff<-mean(alt.y[carrier==1])-mean(alt.y[carrier==0])



one.test <- function(x,y) {
        xstar<-sample(x)
        mean(y[xstar==1])-mean(y[xstar==0])
        }



many.truenull <- replicate(1000, one.test(carrier, null.y))
many.falsenull <- replicate(1000, one.test(carrier, alt.y))
```

Means: permutation test

```
hist(many.truenull)
abline(v=null.diff, lwd=2, col="purple")
mean(abs(many.truenull) > abs(null.diff))


hist(many.falsenull)
abline(v=alt.diff, lwd=2, col="purple")
mean(abs(many.falsenull) > abs(alt.diff))
```

Example: null is true

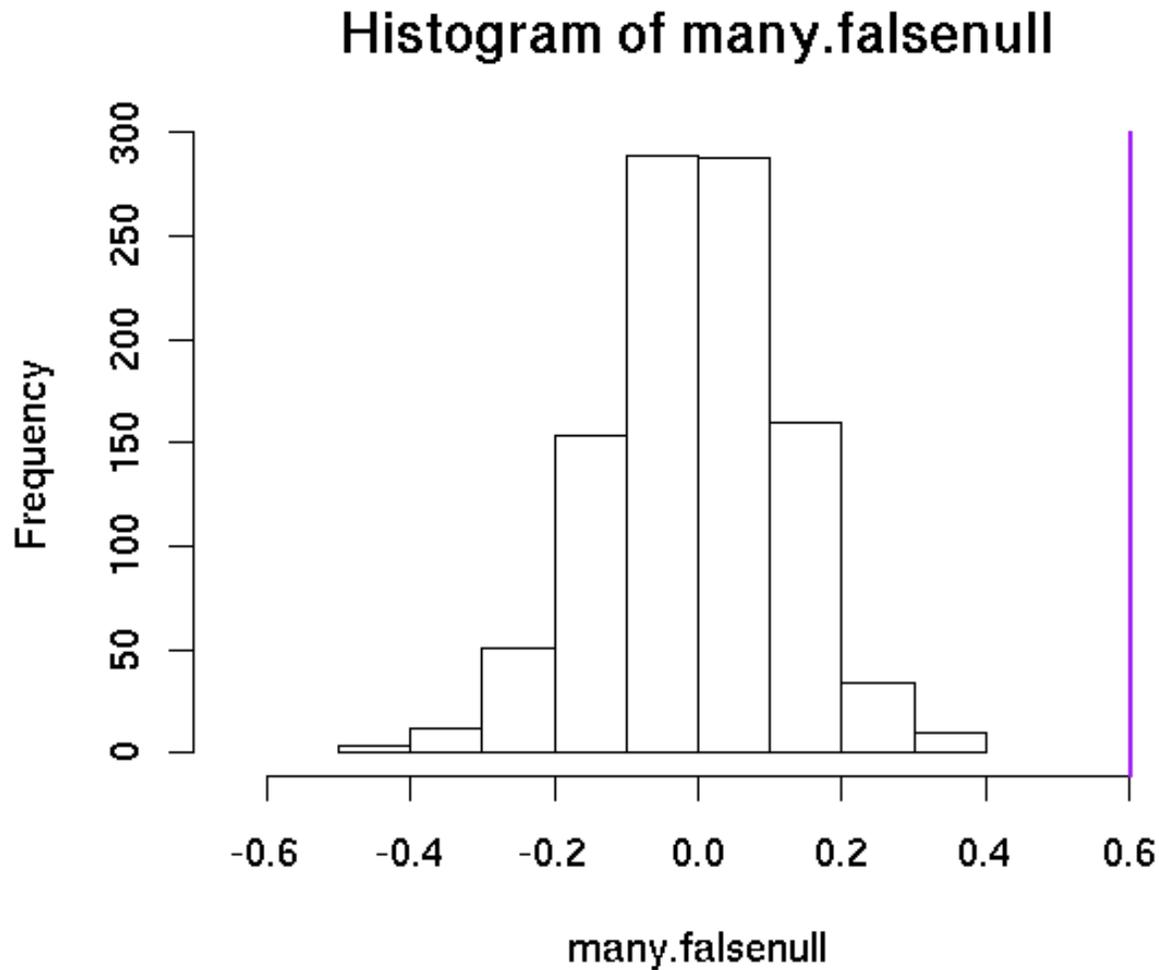Histogram of many.truenull
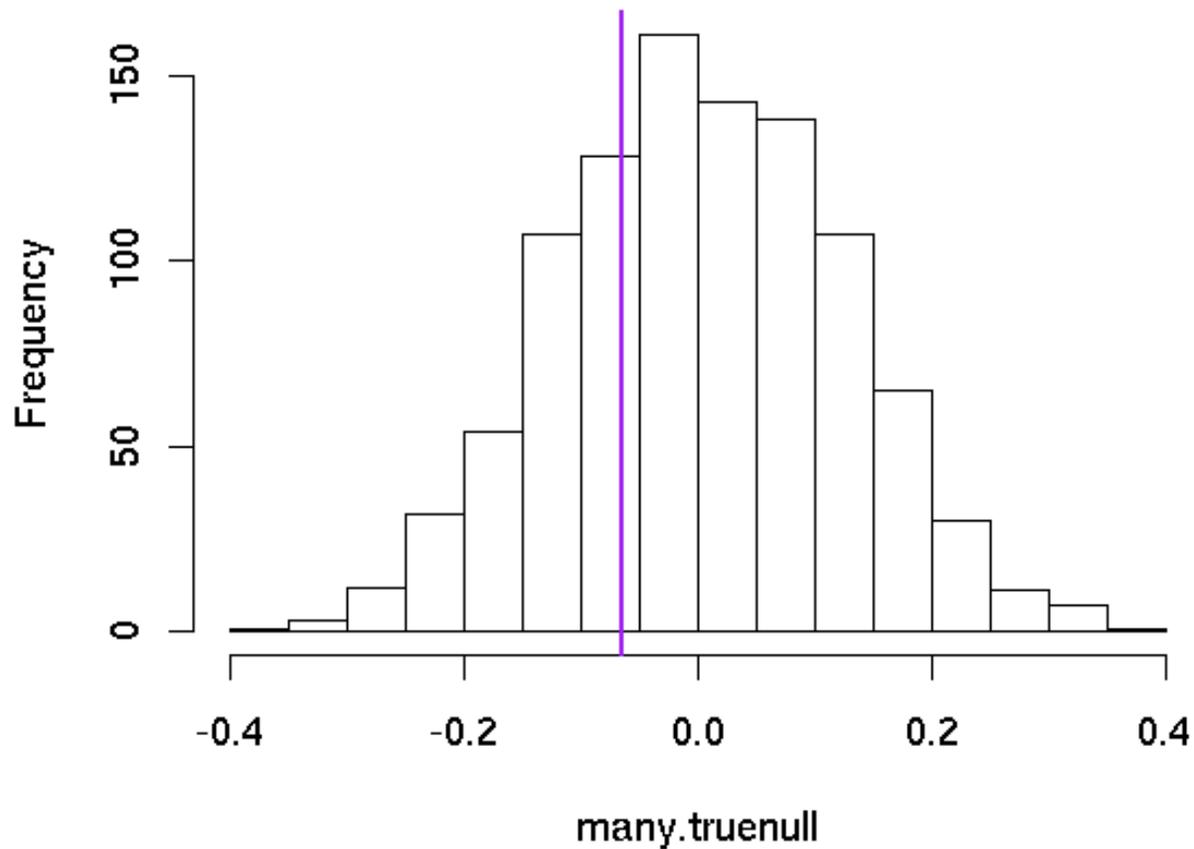
616 shuffled differences exceeed true difference: $p = 617/1001$

Example: null is false

Histogram of many.falsenull

No shuffled difference exceeds true difference: $p = 1/1001$.

How many permutations?

With 1000 permutations the smallest possible p-value is 0.001, and the uncertainty near $p = 0.05$ is about $\pm 1\%$

If we have multiple testing we may need much more precision.

Using 100,000 permutations reduces the uncertainty near $p = 0.05$ to $\pm 0.1\%$ and allows p-values as small as 0.00001.

A useful strategy is to start with 1000 permutations and continue to larger numbers only if p is small enough to be interesting, eg $p < 0.1$.

Parallel computing of permutations is easy: just run R on multiple computers.

Debugging

R error messages are sometimes opaque, because the error occurs in a low-level function.

`traceback()` reports the entire call stack, which is useful for seeing where the error really happened.

Suppose our outcome variable were actually a data frame with one column rather than a vector:

```
>  one.test(carrier,ywrong)
Error in '[.data.frame'(y, xstar == 1) : undefined columns selected
```

We didn't know we were calling '[.data.frame', so we don't understand the message

Debugging

```
> traceback()
5: stop("undefined columns selected")
4: '[.data.frame'(y, xstar == 1)
3: y[xstar == 1]
2: mean(y[xstar == 1])
1: one.test(carrier, ywrong)
```

so the problem happens in our line of code `mean(y[xstar==1])` and is a problem with computing `y[xstar==1]`.

We might want to have a look at `y` and `xstar`

Debugging

The post-mortem debugger lets you look inside the code where the error occurred.

```
> options(error=recover)
>  one.test(carrier,ywrong)
Error in '[.data.frame'(y, xstar == 1) : undefined columns selected

Enter a frame number, or 0 to exit

1: one.test(carrier, ywrong)
2: mean(y[xstar == 1])
3: y[xstar == 1]
4: '[.data.frame'(y, xstar == 1)

Selection: 1
Called from: eval(expr, envir, enclos)
```

Debugging

```
Browse[1]> str(xstar)
 num [1:300] 1 1 1 1 1 1 1 0 1 1 ...
Browse[1]> str(y)
'data.frame': 300 obs. of  1 variable:
 $ null.y: num   1.265  0.590 -0.722  0.676 -0.431 ...
```

Turn the post-mortem debugger off with

```
options(error=NULL)
```

Minimum p-value

Little point in permutation test for the mean: same result as t-test

Permutation test is useful when we do not know how to compute the distribution of a test statistic.

Suppose we test additive effects of 8 SNPs, one at a time, and we want to know if the most significant association is real.

For any one SNP the z-statistic from a logistic regression model has a Normal distribution.

We need to know the distribution of the most extreme of eight z-statistics. This is not a standard distribution, but a permutation test is still straightforward.

Minimum p-value

```
dat <- data.frame(y=rep(0:1,each=100), SNP1=rbinom(200,2,.1),
SNP2=rbinom(200,2,.2),SNP3=rbinom(200,2,.2),
SNP4=rbinom(200,2,.4),SNP5=rbinom(200,2,.1),
SNP6=rbinom(200,2,.2),SNP7=rbinom(200,2,.2),
SNP8=rbinom(200,2,.4))


> head(dat)
  y SNP1 SNP2 SNP3 SNP4 SNP5 SNP6 SNP7 SNP8
1 0    0    0    0    0    0    1    0    0
2 0    0    1    0    1    0    1    0    2
3 0    0    1    0    1    1    0    0    0
4 0    0    0    1    1    0    0    0    0
5 0    0    1    0    1    1    0    0    0
6 0    0    0    0    1    0    1    0    1
```

Minimum p-value

```r
oneZ<-function(outcome, snp){
    model <- glm(outcome~snp, family=binomial())
    coef(summary(model))["snp","z value"]
}
maxZ<-function(outcome, snps){
    allZs <- sapply(snps,
        function(snp) oneZ(outcome, snp))
    max(abs(allZs))
}


true.maxZ<-maxZ(dat$y, dat[,-1])

manypermZ<-replicate(10000, maxZ(sample(dat$y), dat[,-1]))
```
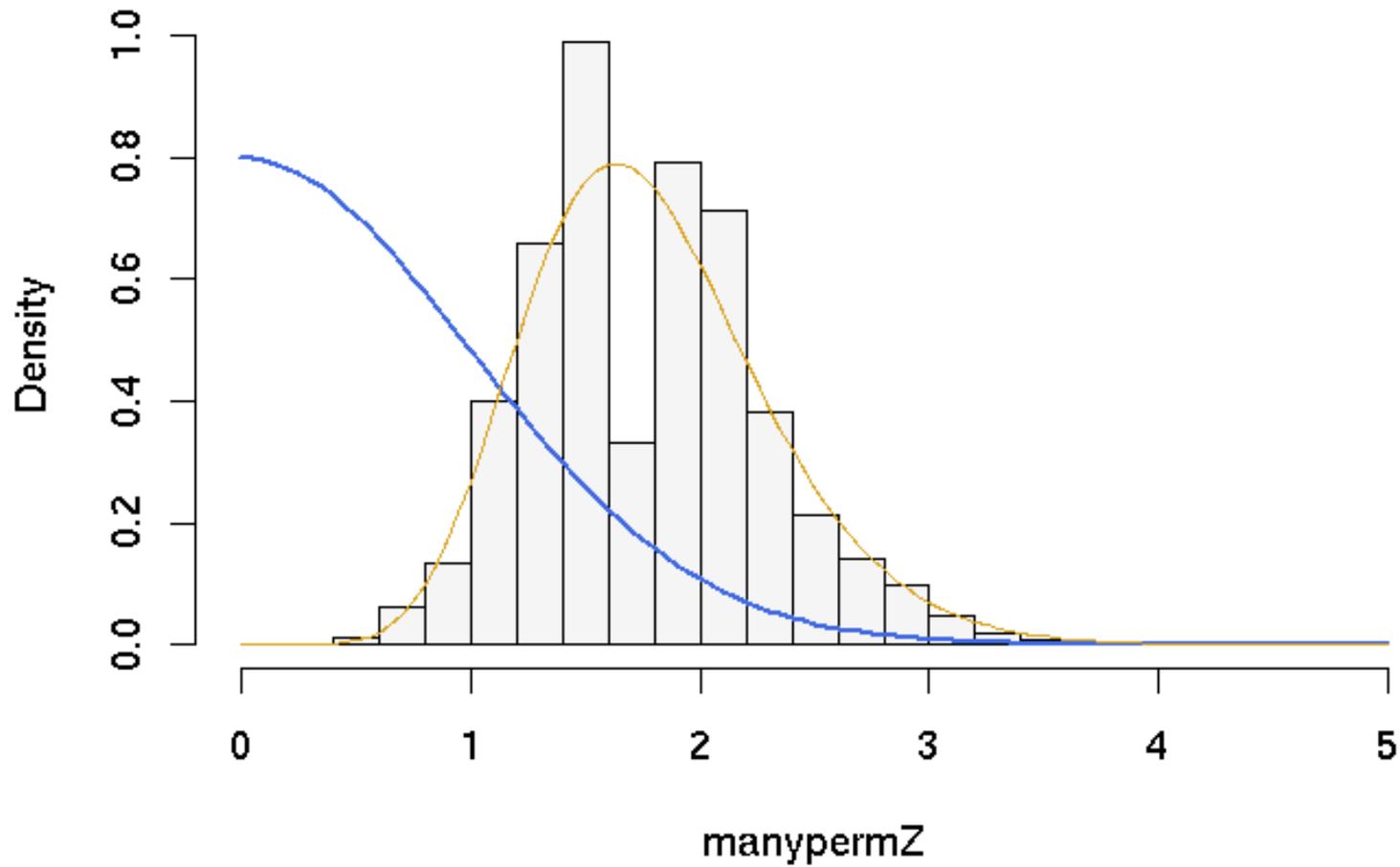
Minimum p-value

Histogram of manypermZ

Minimum p-value

The histogram shows the permutation distribution for the maximum Z-statistic.

The blue curve is the theoretical distribution for one Z-statistic

The yellow curve is the theoretical distribution for the maximum of eight independent Z-statistics.

Clearly the multiple testing is important: a Z of 2.5 gives $p = 0.012$ for a single test but $p = 0.075$ for the permutation test.

The theoretical distribution for the maximum has the right range but the permutation distribution is quite discrete. The discreteness is more serious with small sample size and rare SNPs.

[The theoretical distribution is not easy to compute except when the tests are independent.]

More debugging

Permutation tests on other people's code might reveal a lack of robustness.

For example, a permutation might result in all controls being homozygous for one of the SNPs and this might give an error message

We can work around this with `tryCatch()`

```
oneZ<-function(outcome, snp){
    tryCatch({model <- glm(outcome~snp, family=binomial())
              coef(summary(model))["snp","z value"]},
        error=function(e) rep(NA, 8)
        )
}
```

Now `oneZ` will return a vector of `NA` if there is an error in the model fitting.

Caution: wrong null

Permutation tests cannot solve all problems: they are valid only when the null hypothesis is 'no association'

Suppose we are studying a set of SNPs that each have some effect on outcome and we want to test for interactions (epistasis).

Permuting the genotype data would break the links between genotype and outcome and created shuffled data with no main effects of SNPs.

Even if there are no interactions the shuffled data will look different from the real data.

Caution: weak null hypothesis

A polymorphism could increase the variability of an outcome but not change the mean.

In this case the strong null hypothesis is false, but the hypothesis of equal means is still true.

- If we want to detect this difference the permutation test is unsuitable because it has low power
- If we do not want to detect this difference the permutation test is invalid, because it does not have the correct Type I error rate.

When groups are the same size the Type I error rate is typically close to the nominal level, otherwise it can be too high or too low.

To illustrate this we need many replications of a permutation test. We will do 1000 permutation tests for a mean, each with 1000 permutations.

```r
meandiff<-function(x,trt){
mean(x[trt==1])-mean(x[trt==2])
}
meanpermtest<-function(x,trt,n=1000){
observed<-meandiff(x,trt)
perms<-replicate(n, meandiff(x, sample(trt)))
mean(abs(observed)>abs(perms))
}


trt1<-rep(c(1,2),c(10,90))

perm.p<-replicate(1000, {
        x1<-rnorm(100, 0, s=trt1)
        meanpermtest(x1,trt1)})

table(cut(perm.p,c(0,.05,.1,.5,.9,.95,1)))/1000
```

(0,0.05]	(0.05,0.1]	(0.1,0.5]	(0.5,0.9]	(0.9,0.95]	(0.95,1]
86	99	564	244	6	0

The p-values are too small, relative to a uniform distribution. If we reverse the standard errors we get

(0,0.05]	(0.05,0.1]	(0.1,0.5]	(0.5,0.9]	(0.9,0.95]	(0.95,1]
27	28	275	354	67	249

If the two groups each have 50 observations we get

(0,0.05]	(0.05,0.1]	(0.1,0.5]	(0.5,0.9]	(0.9,0.95]	(0.95,1]
50	45	403	407	52	43

which is much better.