Power for Interactions

Rik Henson, rik.henson@mrc-cbu.cam.ac.uk. Please report any errors or confusions (sure there are some!)

2023-08-19

Is it *really* difficult to detect interactions?

There is a common perception that it is difficult to detect interactions (moderations) in multiple linear regression (MLR). There are several possible reasons for this, but a priori, I think the power for detecting an interaction is often equivalent to that for detecting a main effect, except in a few special cases, as demonstrated in the code below.

In a MLR with two independent variables (regressors) X and Z predicting the dependent variable Y, we can express a (linear) interaction as the product of X and Z:

`Y ~ Bx*X + Bz*Z + Bxz*(X*Z) + E`

where `Y`, `X` and `Z` are vectors, `Bx`, `Bz` and `Bxz` are parameters (scalars), `*` is the element-wise product, and `E` is a vector of residuals. We'll label the interaction term as XZ.

This paper McClelland & Judd 1993 argues that moderation effects are difficult to detect. It makes a useful distinction between "experimental" and "field" studies. Experimental studies are assumed to manipulate (often extreme) values of X and Z, represented here as 2 levels of each factor (i.e categorical variables, indicated using dummy variables with values [0,1] in the regressors). Field studies, on the other hand, measure continuous values of X and Z. Below I simulate both types, and argue that interactions are in principle just as detectable as main effects for experimental studies, and can be less or more detectable than main effects in field studies, depending on the range (spread) of X and Z, the correlation between them, and the presence of any error in measuring X and Z.

Before simulating, here are some other possible reasons for a difficulty in detecting an interaction, which are not so relevant to discussion here:

1. It may be an empirical fact that interactions between the types of variables measured in many studies have a smaller effect size than the main effect of those variables. Or there may be theoretical reasons why the interaction could only be ordinal rather than disordinal (cross-over) (see Lakens Blog, in which case the effect size might be half that for the main effects. Here however we consider the case when there is no such empirical or a priori knowledge, i.e, when effect sizes for X, Z and XZ are equal (ie `Bx=Bz=Bxz=1`).
2. The addition of the interaction term to the MLR model eats an additional degree of freedom (df). This is true, but it is strange to not include it, unless we have a priori reason not to expect an interaction, and in any case, its effect on statistical power is negligible (compared to tests of main effects) if we have enough data, as here.
3. It is important to mean-correct X and Z before multiplying them in equation above, otherwise the interaction term XZ will be more correlated with X and Z (if one or both have non-zero mean), i.e, the interaction and main effects will be confounded. (Note interaction terms are orthogonal to main effects in experimental studies, by design).
4. Sometimes people think they have low power to detect "interactions" because they have fewer values per cell when they break down the data according to the specific levels of two or more factors. However, this really refers to the power to detect the "simple effects" on subsets of the data, for example to interpret what is driving an overall interaction; here we only consider the omnibus interaction involving all the data.
5. Sometimes people say (for experimental studies) that taking differences of random samples reduces the signal-to-noise ratio (SNR). This is true if the mean of two independent samples have the same sign, because signal will be decreased by subtraction but noise will be increased. Hence one might think that interactions, as differences of differences, would have even lower SNR. However, interactions are no different from main effects in this sense, since both involve taking differences and averages across conditions (i.e., in a 2x2 design, the contrasts for the two main effects across four conditions might be [1 1 -1 -1] and [1 -1 1 -1], in which case the interaction would be [1 -1 -1 1], all of which involve the same amount of subtraction/averaging, as code below demonstrates).
6. There has been research looking at the effect of "range restriction" on power to detect interactions in MLR (e.g., Aguinis & Stone-Romero 1997), for example where participants are only selected for a study if they score above some criterion on one variable, eg X. This clearly reduces the range of the interaction term too, which of course reduces its power, but here we assume no such range restriction.

Simulations

Power is easily simulated by generating random data many times (ie simulating many studies), fitting a model, and counting how many studies produce a significant result (here, defined as the p-value being less than alpha=0.05, calculated separately for each main effect and interaction).

Let's start by defining some parameters:

```
nsim = 20         # Number of simulations (determines precision of power estimate)
nexp = 100        # Number of experiments/studies (determines precision of power estimate)
nsub = 40         # Number of subjects (must be >1 and multiple of 4 if arg$expt=1)
Beta = c(1, 1, 1) # Betas (effect sizes) for X, Z, XZ
nsd = 15          # sd of measurement noise
```

Most should be self-explanatory, or expanded below. The values of `nsim`, `nexp` and `nsub` are just chosen to give reasonable precision on power estimates below that can be calculated in reasonable time on typical PC (10-30s), but you can increase them if you want more precise estimates (at the expense of time).

We can gather these parameters into an R list `arg` and pass to the function `pow` below to return the proportion of true effects detected (with p-value < 0.05). In brief, the `pow` function itself loops through `nsim` iterations of `nexp` experiments on `nsub` subjects, on each occasion generating random data with the `gen_dat` function, and then fitting the MLR above using `lm` or `lme` in R. The details of these functions can be explored later, but for the moment, just define them:

R Functions

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
# main power function
pow <- function(arg) {
  if (is.null(arg$Xrep)) {arg$Xrep = 0}
  if (is.null(arg$Zrep)) {arg$Zrep = 0}
  if (is.null(arg$seqTest)) {arg$seqTest = 0}
  if (is.null(arg$msd)) {arg$msd = 0}

  SE = sub_eff(arg$nsub, arg$Xrep, arg$Zrep)

  set.seed(100) # if want same results each time run
  psig = matrix(NA, nrow = arg$nsim, ncol = 3)
  alpha = 0.05
  for (i in 1:arg$nsim) {
    nsig = 0
    for (j in 1:arg$nexp) {
      d  = gen_dat(arg)

      if (arg$msd > 0) { # If regressors measured with noise (msd)
        d$X = d$X + rnorm(n = length(d$X), mean = 0, sd = arg$msd)
        d$Z = d$Z + rnorm(n = length(d$Z), mean = 0, sd = arg$msd)
        d$X = (d$X - mean(d$X)); d$Z = (d$Z - mean(d$Z))
        d$XZ = d$X*d$Z
      }

       if (arg$expt & (arg$Xrep | arg$Zrep)) {
        d$ind = SE$ind
        m = lme(y ~ X + Z + XZ, data = d, random = ~ 1 | ind)
        nsig = nsig + (summary(m)$tTable[2:4,5] < alpha)
      } else { # basic lm quicker (ie when no subject effects)
        m = lm(y ~ X + Z + XZ, data = d)
        if (arg$seqTest) { nsig = nsig + (anova(m)$`Pr(>F)`[1:3] < alpha) }
        else             { nsig = nsig + (summary(m)$coefficients[2:4, 4] < alpha) }
      }
     }
    psig[i, ] = nsig / arg$nexp
  }
  return(psig)
}

# generate data
gen_dat <- function(arg) {
  if (is.null(arg$Xrep)) {arg$Xrep = 0}
  if (is.null(arg$Zrep)) {arg$Zrep = 0}

    if (arg$expt) {
    O = matrix(1, nrow = arg$nsub/4, ncol = 1)
    X = c(0, 0, 1, 1)
    Z = c(0, 1, 0, 1)
    X = X %x% O;    Z = Z %x% O
    X = scale(X);   Z = scale(Z)
  } else if (is.null(arg$R)) {
    C = matrix(c(arg$vX, arg$cXZ, arg$cXZ, arg$vZ), ncol = 2)
    R = mvtnorm::rmvnorm(arg$nsub, c(X = 0, Z = 0), C)
    X = R[, 1];     Z = R[, 2]
  } else {
    X = arg$R[, 1]; Z = arg$R[, 2]
  }

  X = (X - mean(X)) # /sd(X) - may not want to scale if range meaningful and of interest
  Z = (Z - mean(Z)) # /sd(X) - may not want to scale if range meaningful and of interest
  XZ = X * Z

  if (arg$expt & (arg$Xrep>0 | arg$Zrep>0)) {
    Xm = matrix(c(1, arg$Xrep, arg$Xrep, 1), ncol = 2)
    Zm = matrix(c(1, arg$Zrep, arg$Zrep, 1), ncol = 2)
    C = (Xm %x% Zm) %x% (arg$scor * diag(arg$nsub/4))
    C = C + (diag(arg$nsub) * (1-arg$scor))
    C = arg$nsd * C
  }  else {
    C = arg$nsd * diag(arg$nsub)
  }

  E = mvtnorm::rmvnorm(1, matrix(0, nrow = arg$nsub, ncol = 1), C)

  y = arg$Beta[1] * X + arg$Beta[2] * Z + arg$Beta[3] * XZ + t(E)
  d = data.frame(y, X, Z, XZ)
  return(d)
}

# sub function to deal with subject effects
sub_eff <- function(nsub,Xrep,Zrep) {
  SX = diag(2)
  SZ = diag(2)
  if (Xrep > 0) { SX = matrix(c(1, 1), nrow = 2, ncol = 1) }
  if (Zrep > 0) { SZ = matrix(c(1, 1), nrow = 2, ncol = 1) }

  mat = (SX %x% SZ) %x% diag(nsub/4)

  ind = matrix(NA, nrow = nrow(mat), ncol = 1)
  for (j in 1:ncol(mat)) {
       ind[(mat[,j]!=0)] = j
  }
  SE = list(mat=mat, ind=ind)
  return(SE)
}

# plot results
plot_bar <- function(psig) {
  d = data.frame(effect = c('X','Z','XZ'), power = apply(psig,2,mean), sd = apply(psig,2,sd))
  ggplot(d) +
    geom_bar( aes(x=factor(effect, levels = c('X','Z','XZ')), y=power), stat="identity", fill="skyblue", alpha=0.
7) + xlab('Effect') +
    geom_errorbar( aes(x=effect, ymin=power-sd, ymax=power+sd), width=0.4, colour="orange", alpha=0.9, linewidth=
1.3)
}
```

Experimental studies

Ok, so for experimental studies, we can create regressors from dummy coding the two levels of X and Z in a 2x2 design. This is done within the `gen_dat` function with the lines:

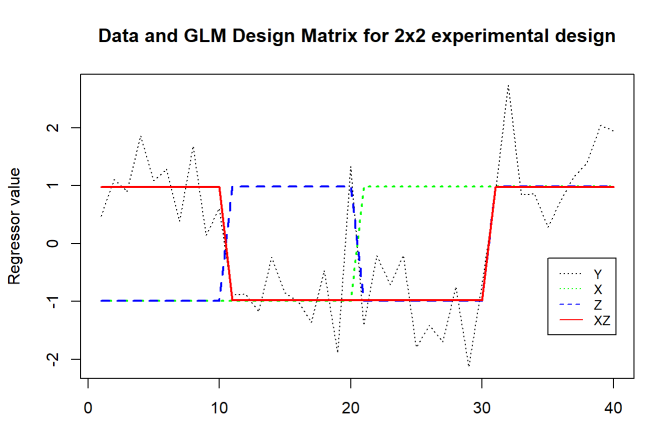
```
arg = list(nsub=40) # not in gen_dat, but to get snippet below working
O = matrix(1, nrow = arg$nsub/4, ncol = 1)
X = c(0, 0, 1, 1); Z = c(0, 1, 0, 1)
X = X %x% O; Z = Z %x% O
X = scale(X); Z = scale(Z)

XZ = X * Z
```

(see Henson, 2015 if you want more information about the general linear model for factorial designs, eg ANOVAs). Note that the scaling of dummy regressors does matter - they need to be Z-scored (`scale` in R) so that their mean=0 and SD=1 - on assumption that we do not know the true scaling of the underlying factors being manipulated (or orthonormalised if more than 2 levels per factor). The interaction term is then calculated by the product of Z-scored X and Z regressors (see comment above about mean-correction).

To visualise these regressors together with some example data (with an interaction):

```
set.seed(100)
Y = 0 + 0*X + 0*Z + 1*XZ + rnorm(40, mean=0, sd=1) # Just generate some example data with an interaction
colors = c("black", "green", "blue", "red")
matplot(cbind(Y, X, Z, XZ), type = "l", lwd = c(1,2,2,2), main="Data and GLM Design Matrix for 2x2 experimental d
esign", col=colors, lty=c(3,3,2,1), ylab="Regressor value")
legend(35, -0.25, legend=c("Y", "X", "Z", "XZ"), col=colors, lty=c(3,3,2,1), cex=0.8)
```



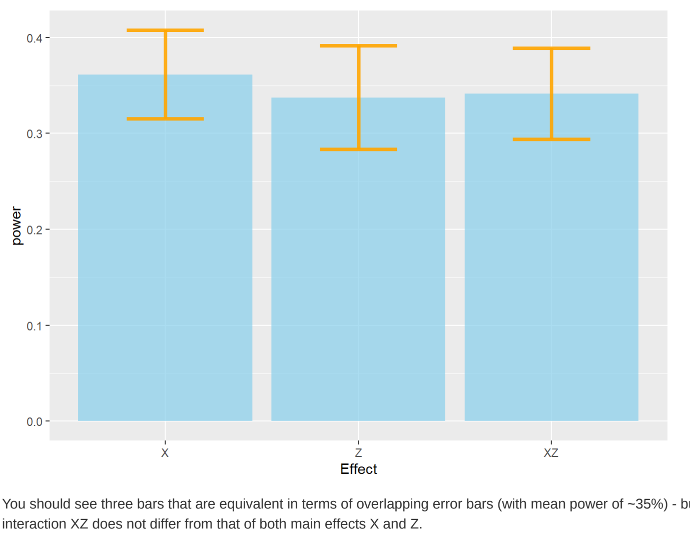
The GLM finds the best linear combination of the three regressors (solid lines) to fit the data (dotted line); in this case, you should be able to see that the parameter estimates should be (on average) 1 for XZ regressor and 0 for the X and Z regressors.

The rest of `gen_dat` deals with adding correlated data across subjects (determined by the parameter `scor`) if one or both factors is a repeated measure, which we will consider for within-subject designs later. First, let's estimate power for a simple case where each level of X and Z is measured on different subjects (i.e, a fully between-subject design), i.e., we have nsub/4 subjects in each of 4 groups.

2x2 experiment, fully between-subjects

To tell the code that X and Z are from an experimental design (rather than field study) in which two levels were tested for each, we set the `expt` variable to 1. We can gather these into a list called `arg` to pass to the `pow` function:

```
arg = list(nsim = nsim, nexp = nexp, nsub = nsub, Beta = Beta, nsd = nsd)
arg = list.append(arg, expt = 1)
#system.time({psig = pow(arg)}) # should be ~10s
plot_bar(pow(arg))
```



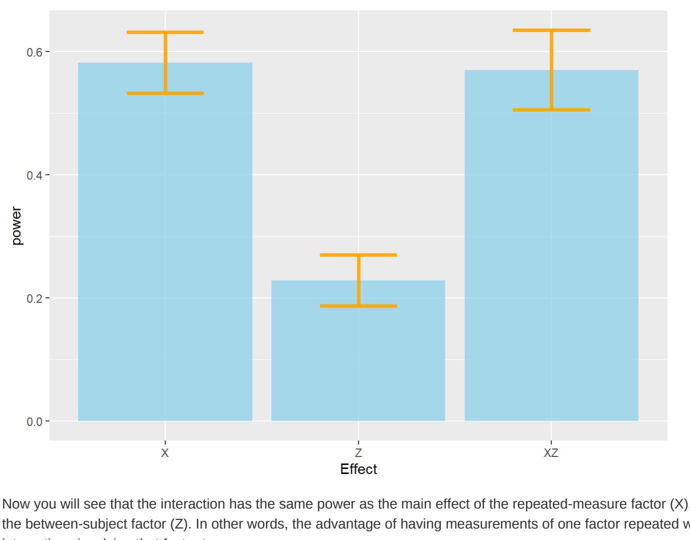
You should see three bars that are equivalent in terms of overlapping error bars (with mean power of ~35%) - but most importantly, the power for interaction XZ does not differ from that of both main effects X and Z.

2x2 experiment, within-subject factor(s)

If you want to convert to a fully within-subject design, you need to set Xrep and Zrep parameters to 1, to indicate that both are repeated-measures. The degree of correlation between measures is then controlled by the parameter `scor`, which we can set to 0.5 for example. This affects the covariance used to create the data in `gen_dat`, and switches to (slightly slower) `lme` in `pow` function, in order to model subjects as random effects.

If you uncomment the first call to plot_bar below, you will see that power increases to ~60% relative to the between-subject design above, but remains equivalent between the interaction and the main effects. Perhaps more interesting is to make one factor (eg X) a repeated-measure, but keep the other factor (eg Z) as two groups of independent subjects:

```
arg$Xrep = 1; arg$Zrep = 1; arg$scor = 0.5;
#plot_bar(pow(arg))
arg$scor = 0.5; arg$Xrep = 1; arg$Zrep = 0;
plot_bar(pow(arg))
```



Now you will see that the interaction has the same power as the main effect of the repeated-measure factor (X), and both have greater power than the between-subject factor (Z). In other words, the advantage of having measurements of one factor repeated within subjects transfers to interactions involving that factor too.

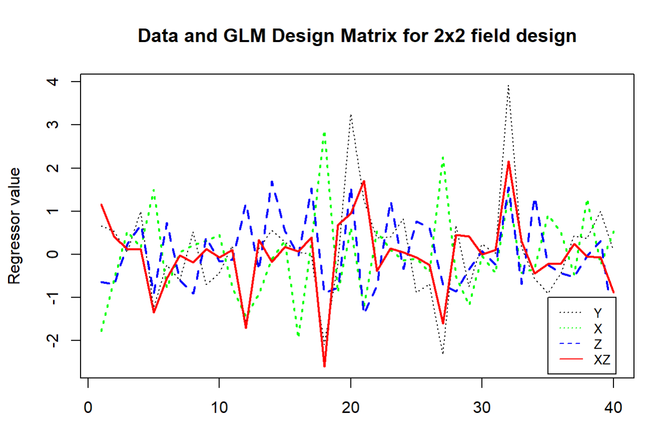
Field studies

Now let's consider a study in which X and Z are continuous variables, e.g, measurements of some variable in the environment (we will assume zero noise in those measurements for moment; though see later). We can do this by saying it is not an experimental study, ie `arg$expt=0`. This then generates the regressors afresh for each simulated study (to minimise random error) from a multivariate normal distribution paramaterised by variances `vX` and `vZ` and covariance `cXZ`, as in these lines of `gen_dat`:

```
arg = list(vX = 1, vZ = 1, cXZ = 0) # not in gen_dat, but to get snippet below working
C = matrix(c(arg$vX, arg$cXZ, arg$cXZ, arg$vZ), ncol = 2)
R = mvtnorm::rmvnorm(nsub, c(X = 0, Z = 0), C)
X = R[, 1]; Z = R[, 2]
X = (X - mean(X))
Z = (Z - mean(Z))
XZ = X * Z
```

The above example assumes X and Z are independent, ie covariance `cXZ=0` (i.e., orthogonal on average), as in first example below. To visualise these regressors and some example data (without an interaction):

```
set.seed(100)
Y = 0 + 0*X + 0*Z + 1*XZ + rnorm(40, mean=0, sd=1) # Just generate some example data with an interaction
colors = c("black", "green", "blue", "red")
matplot(cbind(Y, X, Z, XZ), type = "l", lwd = c(1,2,2,2), main="Data and GLM Design Matrix for 2x2 field design",
col=colors, lty=c(3,3,2,1), ylab="Regressor value")
legend(35, -1, legend=c("Y", "X", "Z", "XZ"), col=colors, lty=c(3,3,2,1), cex=0.8)
```

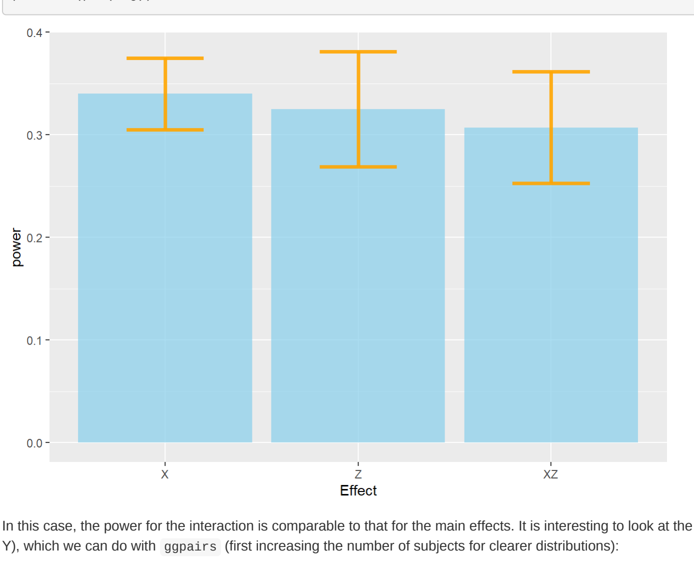


Again, in this example generated with only an interaction, you should be able to see that the data closely follows the interaction regressor XZ.

Independent variables

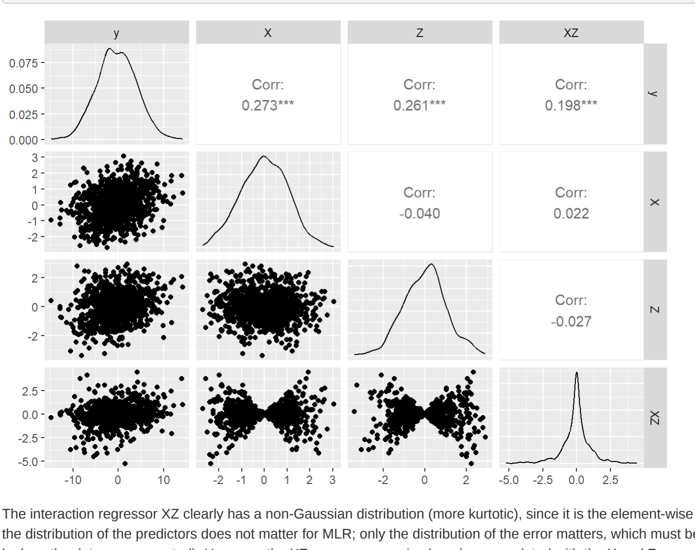
To start with, we will assume X and Z have unit variance (sd=1), by setting the `vX=1` and `vZ=1`, and are independent, by setting `cXZ=0`:

```
arg = list(nsim = nsim, nexp = nexp, nsub = nsub, Beta = Beta, nsd = nsd)
arg = list.append(arg, expt = 0, vX = 1, vZ = 1, cXZ = 0)
plot_bar(pow(arg))
```



In this case, the power for the interaction is comparable to that for the main effects. It is interesting to look at the distributions of X, Z and XZ (and Y), which we can do with `ggpairs` (first increasing the number of subjects for clearer distributions):

```
arg2=arg; arg2$nsub = 1000; ggpairs(gen_dat(arg2))
```



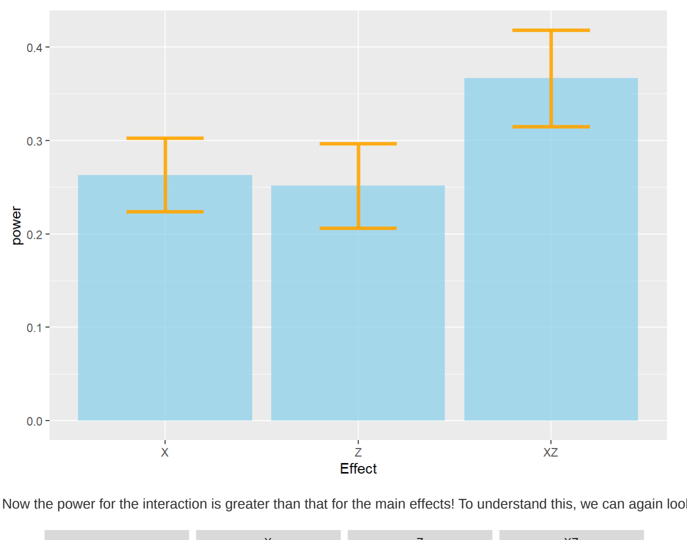
The interaction regressor XZ clearly has a non-Gaussian distribution (more kurtotic), since it is the element-wise product of two Gaussians. (Note the distribution of the predictors does not matter for MLR; only the distribution of the error matters, which must be Gaussian, which is ensured here by how the data are generated). However, the XZ regressor remains largely uncorrelated with the X and Z regressors, so has comparable power (we will see below how high correlation between regressors reduces the power to detect their unique effects).

In fact, if you increase the number of simulations (e.g. `nsim=1000; nexp=1000`), to reduce the error bars on the power estimates, you will see (after several hours!) a small but reliable decrease in power for the interaction XZ relative to the main effects. I suspect this reflects a reduced effective "range" of the XZ regressor owing to the higher-order terms in Equation 2 of McClelland & Judd 1993. Nonetheless, at least for the parameter values here, the loss in power is negligible compared to effects of more correlated regressors and measurement noise considered below.

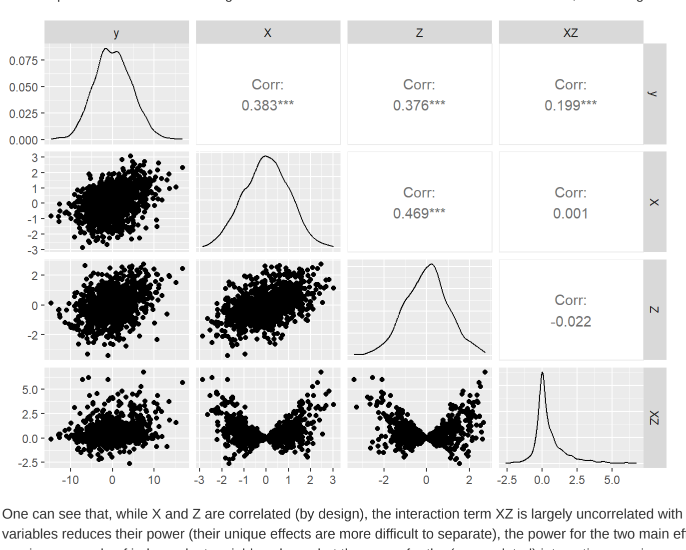
Correlated variables

What happens if X and Z are correlated? To see this, we will set `cXZ=0.5`:

```
arg$cXZ = 0.5
plot_bar(pow(arg))
```



Now the power for the interaction is greater than that for the main effects! To understand this, we can again look at the distributions of X, Z and XZ:

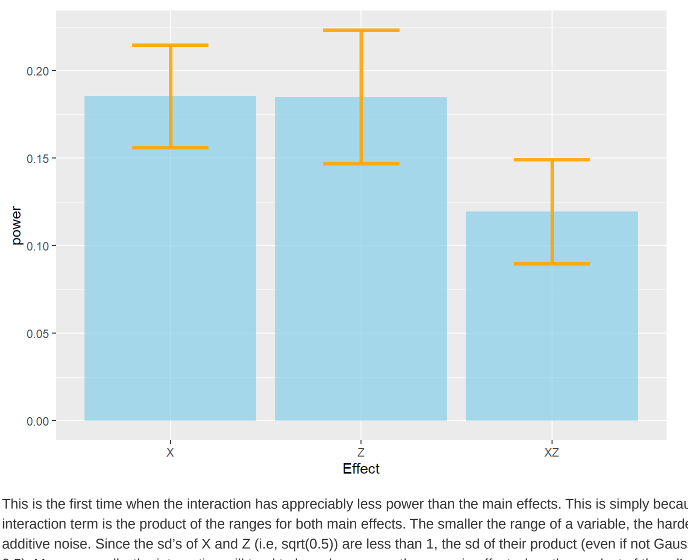


One can see that, while X and Z are correlated (by design), the interaction term XZ is largely uncorrelated with both. Since correlation between two variables reduces their power (their unique effects are more difficult to separate), the power for the two main effects is reduced compared to previous example of independent variables above, but the power for the (uncorrelated) interaction remains comparable to the previous example. This is regardless of the sign of the covariance between X and Z.

Low range of variables

Consider what happens when X and Z have a smaller range, ie `vX<1` and `vZ<1` (while being independent, ie `cXZ=0`):

```
arg$vX = 0.5; arg$vZ = 0.5; arg$cXZ = 0
plot_bar(pow(arg))
```



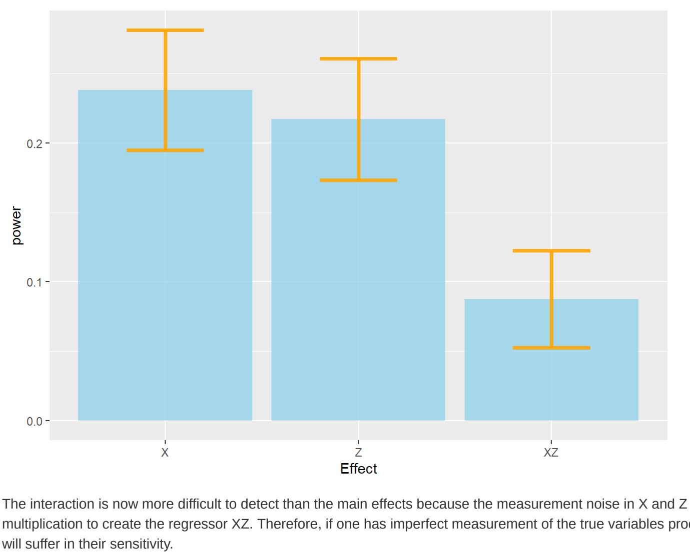
This is the first time when the interaction has appreciably less power than the main effects. This is simply because the range (sd in this case) of the interaction term is the product of the ranges for both main effects. The smaller the range of a variable, the harder it is to detect in the presence of additive noise. Since the sd's of X and Z (i.e, sqrt(0.5)) are less than 1, the sd of their product (even if not Gaussian) must be even smaller (i.e, 0.5). More generally, the interaction will tend to have less power than a main effect when the product of the sd's of the main effects is less than the sd of that main effect (e.g, when `vX*vZ < vX` when comparing to main effect of X). On the other hand, if the sd's of X and Z are greater than 1, then the interaction regressor will have a larger sd than the main effects, and the interaction will therefore have higher power than the main effects!

So when will the range (e.g. sd) of measured variables be smaller or greater than 1? This is difficult to answer in general, because it depends on whether X and Z are measured with meaningful scales (and that those scales are commensurate across X and Z). If their scales are not meaningful / commensurate, then they are usually standardised (i.e, Z-scored) before entering the MLR, which ensures they have sd=1 and therefore that the interaction will have comparable power (at least when X and Z are independent). Nonetheless, it should be remembered that if X and Z have meaningful scales (and are therefore not standardised), along which they vary little, then their interaction can be more difficult to detect.

Measurement noise (in variables)

Another situation where the interaction has less power is when the variables are not measured exactly, i.e, there is noise in their measurement. Strictly speaking, this is an "errors-in-variables" situation, but in practice such error can be absorbed into the `lm` model's single error term. This can be simulated by setting the parameter `msd>0` (note that the data are generated in `gen_dat` with the true values of X and Z, but then fit in `pow` with values of X and Z that have additional Gaussian noise with sd of `msd`).

```
arg = list(nsim = nsim, nexp = nexp, nsub = nsub, Beta = Beta, nsd = nsd)
arg = list.append(arg, expt = 0, vX = 1, vZ = 1, cXZ = 0)
arg$nsd = 2; arg$msd = 2 # reduce main noise (nsd) to get power comparable to previous examples
plot_bar(pow(arg))
```



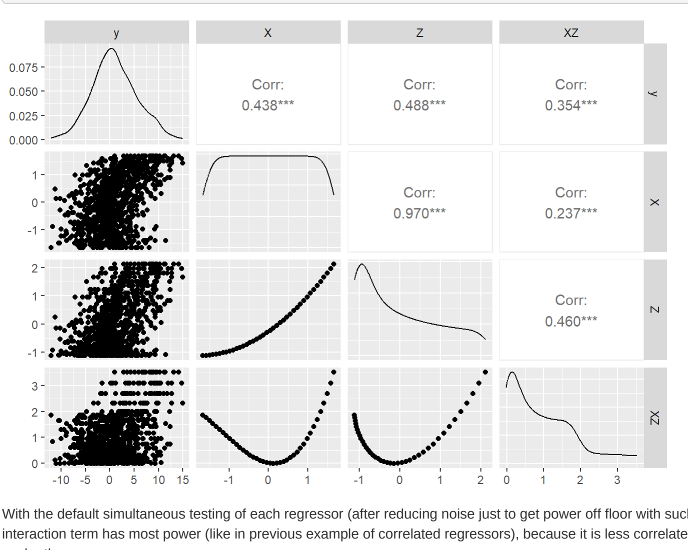
The interaction is now more difficult to detect than the main effects because the measurement noise in X and Z has been amplified simply by their multiplication to create the regressor XZ. Therefore, if one has imperfect measurement of the true variables producing the data, then interactions will suffer in their sensitivity.

Sequential testing

In a final example where the power for detecting an interaction is reduced, we will generate cases where the interaction term is correlated with one or both main effects (even after Z-scoring), and where one decides to partition shared variance "sequentially" to each term in the MLR, giving priority to main effects over interactions. This is a situation where the order of regressors matters in a MLR, and arises when interactions appear to "the right of" main effects in the `lm` equation. In the `pow` function, sequential testing is implemented by using the default `anova` function on the lm fit, which uses type I sum of squares, rather than the usual `summary` function.

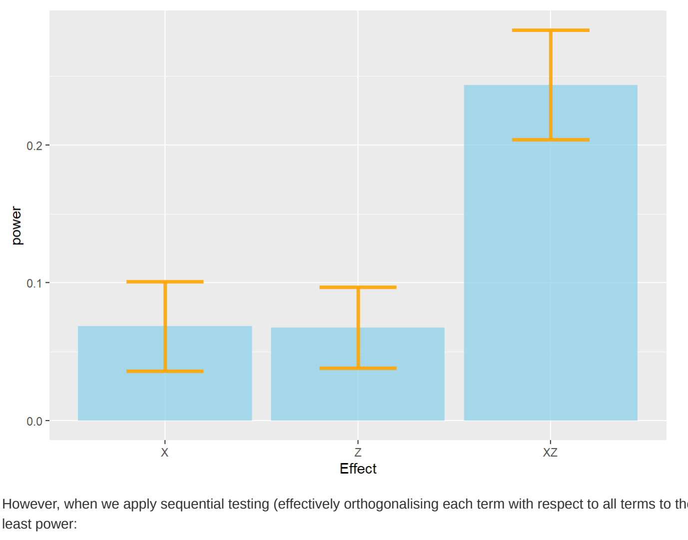
We will also take this opportunity to illustrate a final option of passing user-specified regressors directly to `pow` via the `arg$R` parameter. Here we will generate a uniform distribution for X, and make Z a squared version of X (before mean-correcting X). Both will be standardised before entering the MLR, to ensure their range is equivalent, but the lack of mean-correction of X prior to multiplication with Z ensures that X is highly correlated with Z, and also that the product XZ is correlated with both main effects, as can be seen here:

```
arg = list(nsim = nsim, nexp = nexp, nsub = nsub, Beta = Beta, nsd = nsd, expt = 0)
X = seq(1,arg$nsub)/arg$nsub; Z = X^2; X = scale(X); Z = scale(Z); arg$R = cbind(X, Z)
arg2=arg; arg2$nsub = 1000; ggpairs(gen_dat(arg2))
```



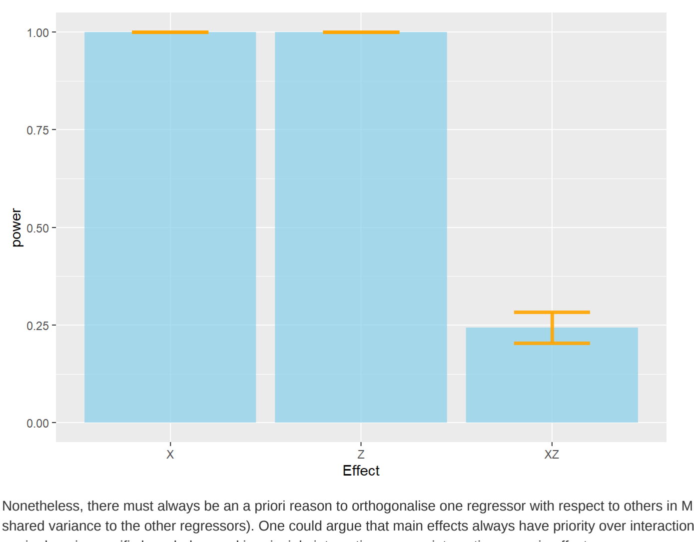
With the default simultaneous testing of each regressor (after reducing noise just to get power off floor with such highly correlated regressors), the interaction term has most power (like in previous example of correlated regressors), because it is less correlated with X and Z than they are with each other:

```
arg$nsd=1
plot_bar(pow(arg))
```



However, when we apply sequential testing (effectively orthogonalising each term with respect to all terms to the left of it), the interaction now has least power:

```
arg$seqTest = 1;
plot_bar(pow(arg))
```



Nonetheless, there must always be an a priori reason to orthogonalise one regressor with respect to others in MLR (i.e, effectively assigning the shared variance to the other regressors). One could argue that main effects always have priority over interactions, but I would argue that this is again domain-specific knowledge, and in principle interactions are as interesting as main effects.

Conclusion

The relative power of interactions versus main effects in multiple linear regression depends on a number of considerations, such as the range of each regressor, the correlation between them, whether there is additional measurement noise in regressor values, and whether significance is assessed with simultaneous or sequential testing. Notwithstanding the caveats mentioned at the start of this piece (e.g, equal effect sizes), I would argue that, in experimental designs, where the scaling of factors is arbitrary (ordinal), and main effects and interactions are orthogonal by design, interactions are equivalent in their power to main effects, assuming all are either measured within- or between-subjects (and when one factor is measured within-subject, the interaction involving that factor gains the same power advantage over between-subject effects). For field designs, with continuous, measured variables, I would argue that interactions still do not *necessarily* have less power, but they can when 1) there is measurement noise in the variables, 2) when the variables have a meaningful scale (such that their standardisation is inappropriate) and their range of values is low, or 3) sequential testing is employed such that interaction terms are effectively de-emphasised by orthogonalising (residualising) them with respect to (correlated) main effects.