Import necessary libraries

```
library(data.table)
library(tidyverse)
library(broom)
library(broom.mixed)
library(lmerTest)
library(reshape2)
library(mvtnorm)
library(MASS)
library(lavaan)
library(sem)
library(semPlot)
library(forecast)
```

Throughout this notebook I use raw (unstandardized) variables to make comparisons easier. When to scale variables (i.e., pre and post separately or together) is a choice of the researcher that will affect results. If pre and post are scaled according to the same linear transformation in both wide and long form, results presented here will be unaffected. If there is a difference in scaling between wide and long forms, presented results will only marginally differ according to differences in scaling. I also only use random intercepts for students and no other variables for simplicity. Adding an additional random intercept (i.e., for class) will only marginally affect results using maximum likelihoods. I additionally refrain from using random slopes. This notebook is meant to address the direct relationship between repeated measures using pre-post data and multiple linear regression.

Load assessment dataset function

```
Load.Clean.Data <- function(File = 'C:/Users/Cole/Documents/GitHub/Omitted-Variable-Bias/OVB_Master.csv
                            assessment, RM = FALSE) {

  df <- fread(File)[Assessment == assessment & (!is.na(PreScores) & !is.na(PostScores))]

  df.assessment <- df %>%
    dplyr::select(Gender, URM_Status, ACT_SAT_Math_Percentile, PreScores, PostScores,
                  Student_ID) %>%
    mutate(Gender = relevel(as.factor(Gender), ref = 'M'),
           URM_Status = relevel(as.factor(URM_Status), ref = 'Majority'),
           ACT_SAT_Math_Percentile = c(scale(ACT_SAT_Math_Percentile, scale = TRUE)),
           Student_ID = as.factor(Student_ID))

  # Unstandardized for simplicity
  if(!RM){
    df.assessment <- df.assessment %>%
      mutate(Gain = PostScores - PreScores)
  } else { # long form for repeated measures dataset
    df.assessment <- df.assessment %>%
      melt(., measure.vars = c('PreScores', 'PostScores'), variable.name = 'Time',
           value.name = 'Score')
```

```
  }

  return(df.assessment)
}
```

Compare repeated measures to OLS using MBT data

```
df.MBT.OLS <- Load.Clean.Data(assessment = 'MBT')

df.MBT.RM <- Load.Clean.Data(assessment = 'MBT', RM = TRUE)

str(df.MBT.OLS)
```

```
## 'data.frame':    600 obs. of  7 variables:
##  $ Gender                 : Factor w/ 2 levels "M","F": 2 1 2 1 1 2 1 2 2 2 ...
##  $ URM_Status             : Factor w/ 2 levels "Majority","URM": 1 1 1 2 2 1 2 1 1 1 ...
##  $ ACT_SAT_Math_Percentile: num  0.45 0.975 0.45 0.45 0.45 ...
##  $ PreScores              : num  16 19 14 10 10 18 15 11 20 13 ...
##  $ PostScores             : num  14 21 16 13 14 16 20 7 21 17 ...
##  $ Student_ID             : Factor w/ 600 levels "3416407","3642780",..: 104 361 129 139 19 462 454 5
##  $ Gain                   : num  -2 2 2 3 4 -2 5 -4 1 4 ...
```

```
str(df.MBT.RM)
```

```
## 'data.frame':    1200 obs. of  6 variables:
##  $ Gender                 : Factor w/ 2 levels "M","F": 2 1 2 1 1 2 1 2 2 2 ...
##  $ URM_Status             : Factor w/ 2 levels "Majority","URM": 1 1 1 2 2 1 2 1 1 1 ...
##  $ ACT_SAT_Math_Percentile: num  0.45 0.975 0.45 0.45 0.45 ...
##  $ Student_ID             : Factor w/ 600 levels "3416407","3642780",..: 104 361 129 139 19 462 454 5
##  $ Time                   : Factor w/ 2 levels "PreScores","PostScores": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Score                  : num  16 19 14 10 10 18 15 11 20 13 ...
```

There are 600 students in the MBT dataset and the same number of observations in the OLS dataset. The RM dataset has twice as many number of observations (pre and post for each student) since the data is in long form. Other than that, the dataframes are identical.

Repeated measures fit with Score as the dependent variable, time (pre/post) and gender as the independent variable.

```
tidy(lmer(Score ~ Time*Gender + (1 | Student_ID), df.MBT.RM))
```

```
## # A tibble: 6 x 8
##   effect  group   term        estimate std.error statistic    df    p.value
##   <chr>   <chr>   <chr>          <dbl>     <dbl>     <dbl> <dbl>      <dbl>
## 1 fixed   <NA>    (Intercept)   14.9       0.262     56.8   932.  4.47e-305
## 2 fixed   <NA>    TimePostSc~    2.30      0.254      9.06  598.  1.78e- 18
## 3 fixed   <NA>    GenderF       -1.88      0.347     -5.42  932.  7.39e-  8
## 4 fixed   <NA>    TimePostSc~    0.689     0.336      2.05  598.  4.05e-  2
## 5 ran_pa~ Studen~ sd__(Inter~    3.07     NA         NA      NA  NA
## 6 ran_pa~ Residu~ sd__Observ~    2.88     NA         NA      NA  NA
```

We can compare this too linear regressions (not mixed models) with prescore/postscore/gain as the dependent variable and gender as the independent variable.

```
tidy(lm(PreScores ~ Gender, df.MBT.OLS))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    14.9      0.258     57.7  1.50e-246
## 2 GenderF        -1.88     0.342     -5.51 5.39e-  8
```

```
tidy(lm(PostScores ~ Gender, df.MBT.OLS))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    17.2      0.266     64.6  5.78e-272
## 2 GenderF        -1.19     0.352     -3.39 7.49e-  4
```

```
tidy(lm(Gain ~ Gender, df.MBT.OLS))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    2.30      0.254      9.06 1.78e-18
## 2 GenderF        0.689     0.336      2.05 4.05e- 2
```

The coefficient, standard error, t value, and p value for gender in the prescore lm model are equal (within rounding errors) to those for gender from the lmer repated measures model. Similarly, gender in the postscore lm model is equal to (gender + time:gender) from the lmer model. And gender from the gain lm model is equal to time:gender from the lmer model. This analysis illustrates that repeated measures with pre/post data is identical to modeling pre/post/gain as a function of other variables, while not controlling for incoming preparation.

With only one independent variable this is equivale to a t-test on prescore/postscore/gain.

```
tidy(t.test(Gain ~ Gender, df.MBT.OLS))
```

```
## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic p.value parameter conf.low
##      <dbl>     <dbl>     <dbl>     <dbl>   <dbl>     <dbl>    <dbl>
## 1   -0.689      2.30      2.99     -2.02  0.0435      521.    -1.36
## # ... with 3 more variables: conf.high <dbl>, method <chr>,
## #   alternative <chr>
```

The estimate, t value and p value are equivalent to that obtained from the gender:time coefficient in the lmer(Score ~ Time*Gender + (1 | Student_ID) repeated measures model (within rounding errors). Similar results are obtained from using prescore or postscore as the dependent variable.

This analysis extends to using multiple independent variables (including continuous ones and interaction terms).

```
tidy(lmer(Score ~ Time*(Gender*URM_Status + ACT_SAT_Math_Percentile) + (1 | Student_ID),
          df.MBT.RM))
```

```
## # A tibble: 12 x 8
##    effect  group  term        estimate std.error statistic    df    p.value
##    <chr>   <chr>  <chr>          <dbl>     <dbl>     <dbl> <dbl>      <dbl>
##  1 fixed   <NA>   (Intercept)   15.4       0.323    47.6    952.  8.33e-254
##  2 fixed   <NA>   TimePostSc~    2.49      0.323     7.70   595.  5.79e- 14
##  3 fixed   <NA>   GenderF       -2.21      0.401    -5.51   952.  4.72e-  8
##  4 fixed   <NA>   URM_Status~   -1.33      0.526    -2.53   952.  1.15e-  2
##  5 fixed   <NA>   ACT_SAT_Ma~    1.08      0.170     6.32   952.  4.06e- 10
##  6 fixed   <NA>   GenderF:UR~    0.551     0.789     0.698  952.  4.85e-  1
##  7 fixed   <NA>   TimePostSc~    0.547     0.401     1.36   595.  1.73e-  1
##  8 fixed   <NA>   TimePostSc~   -0.468     0.527    -0.889  595.  3.75e-  1
##  9 fixed   <NA>   TimePostSc~   -0.428     0.171    -2.51   595.  1.25e-  2
## 10 fixed   <NA>   TimePostSc~    0.114     0.789     0.145  595.  8.85e-  1
## 11 ran_pa~ Stude~ sd__(Inter~    2.86     NA        NA       NA  NA
## 12 ran_pa~ Resid~ sd__Observ~    2.87     NA        NA       NA  NA
```

```
tidy(lm(Gain ~ Gender*URM_Status + ACT_SAT_Math_Percentile, df.MBT.OLS))
```

```
## # A tibble: 5 x 5
##   term                     estimate std.error statistic  p.value
##   <chr>                       <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                 2.49      0.323     7.70  5.79e-14
## 2 GenderF                     0.547     0.401     1.36  1.73e- 1
## 3 URM_StatusURM              -0.468     0.527    -0.889 3.75e- 1
## 4 ACT_SAT_Math_Percentile    -0.428     0.171    -2.51  1.25e- 2
## 5 GenderF:URM_StatusURM       0.114     0.789     0.145 8.85e- 1
```

The interaction terms with time from the lmer model are equal to their counterparts (without the time interaction) from the Gain lm model.

Repeated measures with pre/post data provides no additional information beyond linear regression with prescore/postscore/gain as the dependent variable and other variables (not prescore) as independent variables. This is conceptually different from measuring postscore or gain as the dependent variable and using prescore as a covariate

Its also worth noting that using postscore and gain produce equivalent results when used as dependent variables using OLS

```
tidy(lm(PostScores ~ PreScores + Gender*URM_Status + ACT_SAT_Math_Percentile, df.MBT.OLS))
```

```
## # A tibble: 6 x 5
##   term                     estimate std.error statistic  p.value
##   <chr>                       <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                 9.78      0.642     15.2   1.75e-44
## 2 PreScores                   0.526     0.0374    14.1   4.90e-39
## 3 GenderF                    -0.501     0.366     -1.37  1.72e- 1
## 4 URM_StatusURM              -1.10      0.470     -2.34  1.96e- 2
## 5 ACT_SAT_Math_Percentile     0.0833    0.157      0.532 5.95e- 1
## 6 GenderF:URM_StatusURM       0.376     0.701      0.536 5.92e- 1
```

```r
tidy(lm(Gain ~ PreScores + Gender*URM_Status + ACT_SAT_Math_Percentile, df.MBT.OLS))
```

```
## # A tibble: 6 x 5
##   term                     estimate std.error statistic  p.value
##   <chr>                       <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)               9.78      0.642     15.2   1.75e-44
## 2 PreScores                -0.474     0.0374   -12.7   7.48e-33
## 3 GenderF                  -0.501     0.366     -1.37  1.72e- 1
## 4 URM_StatusURM            -1.10      0.470     -2.34  1.96e- 2
## 5 ACT_SAT_Math_Percentile   0.0833    0.157      0.532 5.95e- 1
## 6 GenderF:URM_StatusURM     0.376     0.701      0.536 5.92e- 1
```

The coefficient on prescore is offset by a factor of 1. This is the only difference between the two models. Scaling variables using different linear transformations or using random intercepts again cause marginal differences between the two models.

Repeated measures on score <=> linear regression on prescore/postscore/gain (without prescore as covariate)

linear regression on gain (with prescore as covariate) <=> linear regression on postscore (with prescore as covariate)

The following references provide further discussion on the comparison between these methods:

Huck, Schuyler W., and Robert A. McLean. "Using a repeated measures ANOVA to analyze the data from a pretest-posttest design: A potentially confusing task." Psychological Bulletin 82.4 (1975): 511.

Werts, Charles E., and Robert L. Linn. "A general linear model for studying growth." Psychological Bulletin 73.1 (1970): 17.

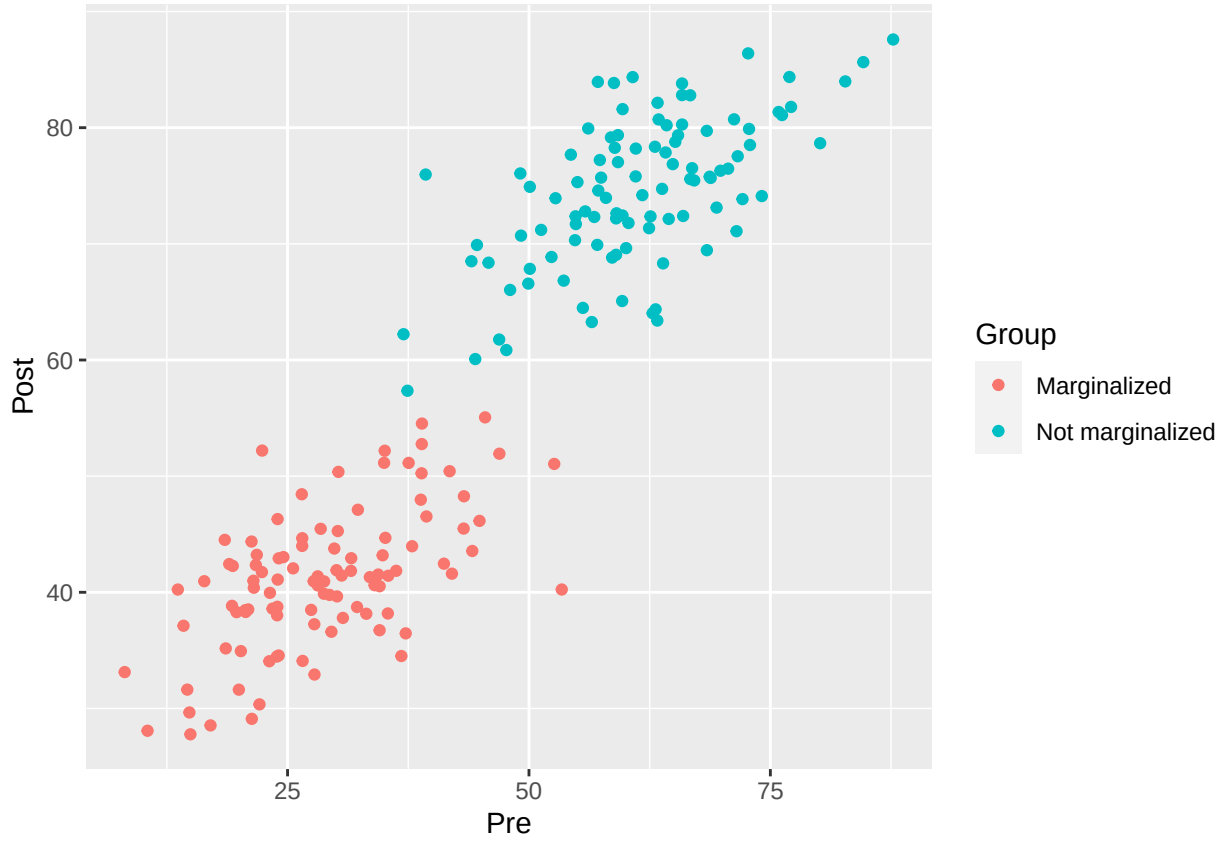
Lord's Paradox

For a brief overview of Lord's paradox, check out the wikipedia page: https://en.wikipedia.org/wiki/Lord%27s_paradox

Consider the following situation: there are 200 students in a class split equally into two groups that are given a pre and post assessment (scored out of 100). Half of the students are underprepared for the course (for whatever reason) and have average prescores of 30. We refer to this as the marginalized group. The other half have stronger backgrounds and have average prescores of 60. We assume that prescores are drawn from normal distributions with standard deviations of 10 for both groups. We also assume that the correlation of prescore with postscore is 0.4.

```r
# Simulate data
set.seed(11)
n = 200
r = 0.4

df <- data.frame(id = 1:n,
                 Pre = c(rnorm(n/2, mean = 30, sd = 10), rnorm(n/2, mean = 60, sd = 10)))
df$Group = ifelse(df$id <= n/2, 'Marginalized', 'Not marginalized')
df$Post <- r * df$Pre + 20 * (df$Group == 'Not marginalized') + rnorm(n, 0, 5) + 30
df$Gain <- df$Post - df$Pre

df.long <- df %>%
```

```
  melt(., measure.vars = c('Pre', 'Post'), variable.name = 'Time', value.name = 'Score')

# Plot simulated data
ggplot(df, aes(x = Pre, y = Post, color = Group)) +
  geom_point()
```




```
tidy(lm(Post ~ Pre + Group, df))
```

```
## # A tibble: 3 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              29.6      1.18       25.2 1.60e-63
## 2 Pre                       0.403    0.0371     10.9 6.67e-22
## 3 GroupNot marginalized   20.1      1.39       14.5 8.20e-33
```

```
tidy(lm(Gain ~ Pre + Group, df))
```

```
## # A tibble: 3 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              29.6      1.18       25.2 1.60e-63
## 2 Pre                      -0.597    0.0371    -16.1 9.14e-38
## 3 GroupNot marginalized   20.1      1.39       14.5 8.20e-33
```

```
tidy(lm(Gain ~ Group, df))
```

```
## # A tibble: 2 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)             12.4       0.751    16.6   2.80e-39
## 2 GroupNot marginalized    0.775     1.06      0.730 4.66e- 1
```

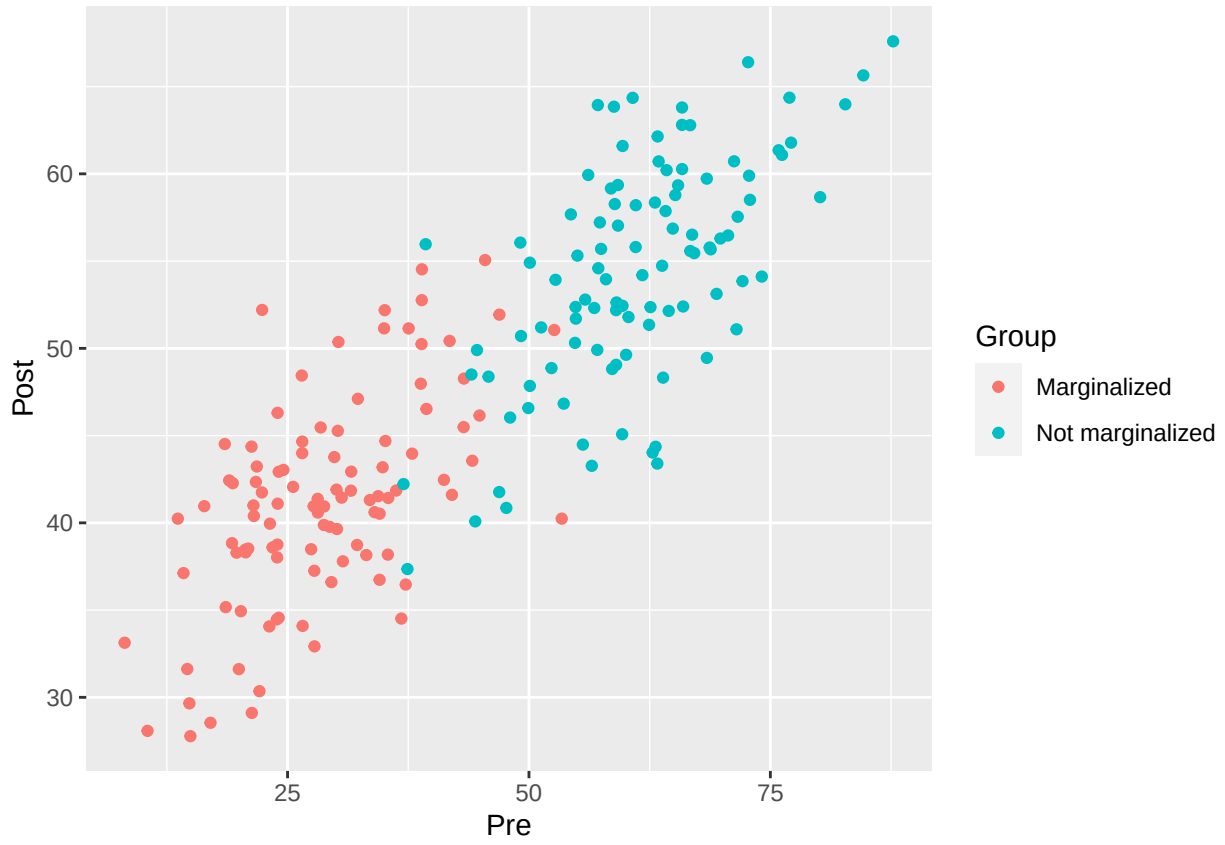
```
tidy(lmer(Score ~ Time*Group + (1 | id), df.long))
```

```
## # A tibble: 6 x 8
##   effect  group  term         estimate std.error statistic    df    p.value
##   <chr>   <chr>  <chr>           <dbl>     <dbl>     <dbl> <dbl>      <dbl>
## 1 fixed   <NA>   (Intercept)    28.8       0.803    35.8    301.  1.62e-110
## 2 fixed   <NA>   TimePost       12.4       0.751    16.6    198.  2.80e- 39
## 3 fixed   <NA>   GroupNot ma~   32.5       1.14     28.6    301.  9.91e- 88
## 4 fixed   <NA>   TimePost:Gr~    0.775     1.06      0.730  198.  4.66e-  1
## 5 ran_pa~ id     sd__(Interc~    6.02     NA        NA       NA  NA
## 6 ran_pa~ Resid~ sd__Observa~    5.31     NA        NA       NA  NA
```

Modeling post (or gain) as a function of prescore produces an estimated average gain for non marginalized students that is 20 points higher than for marginalized students. Modeling gain without pre as a covariate (or using repeated measures) produces (practically) no estimated difference in gains between the two groups. The average gains for students from each group are indeed approximately equal (~12), but which gain a researcher is interested in depends on the research question. As seen from the middle portion of the graph, the difference in gains between students from the two groups is clearly not negligible when looking at students with similar levels of incoming preparation. *I chose the parameters here for visual and statistical effect.*

As another example, consider the following scenario where students from both groups learn approximately equally at all prescores.

```
# Simulate data
set.seed(11)
n = 200
r = 0.4

df <- data.frame(id = 1:n,
                 Pre = c(rnorm(n/2, mean = 30, sd = 10), rnorm(n/2, mean = 60, sd = 10)))
df$Group = ifelse(df$id <= n/2, 'Marginalized', 'Not marginalized')
df$Post <- r * df$Pre + rnorm(n, 0, 5) + 30
df$Gain <- df$Post - df$Pre

df.long <- df %>%
  melt(., measure.vars = c('Pre', 'Post'), variable.name = 'Time', value.name = 'Score')

# Plot simulated data
ggplot(df, aes(x = Pre, y = Post, color = Group)) +
  geom_point()
```




```
tidy(lm(Post ~ Pre + Group, df))
```

```
## # A tibble: 3 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              29.6      1.18      25.2   1.60e-63
## 2 Pre                       0.403    0.0371    10.9   6.67e-22
## 3 GroupNot marginalized     0.141    1.39       0.101 9.19e- 1
```

```
tidy(lm(Gain ~ Pre + Group, df))
```

```
## # A tibble: 3 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              29.6      1.18      25.2   1.60e-63
## 2 Pre                      -0.597    0.0371   -16.1   9.14e-38
## 3 GroupNot marginalized     0.141    1.39       0.101 9.19e- 1
```

```
tidy(lm(Gain ~ Group, df))
```

```
## # A tibble: 2 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              12.4      0.751     16.6  2.80e-39
## 2 GroupNot marginalized   -19.2      1.06     -18.1  7.28e-44
```

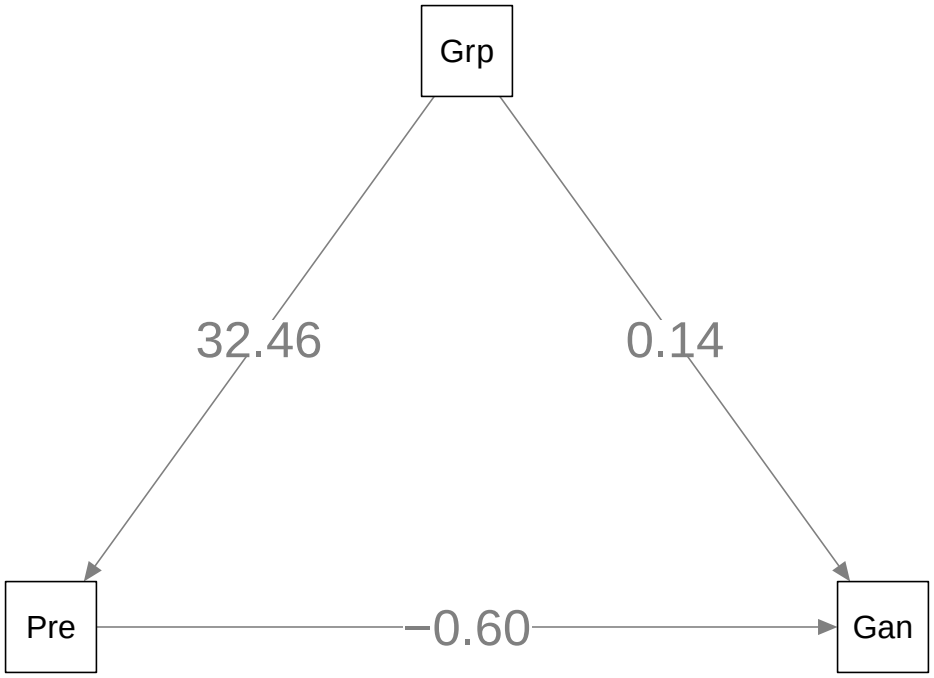
```
tidy(lmer(Score ~ Time*Group + (1 | id), df.long))
```

```
## # A tibble: 6 x 8
##   effect  group  term         estimate std.error statistic    df    p.value
##   <chr>   <chr>  <chr>           <dbl>     <dbl>     <dbl> <dbl>      <dbl>
## 1 fixed   <NA>   (Intercept)     28.8      0.803      35.8  301.  1.62e-110
## 2 fixed   <NA>   TimePost        12.4      0.751      16.6  198.  2.80e- 39
## 3 fixed   <NA>   GroupNot ma~    32.5      1.14       28.6  301.  9.91e- 88
## 4 fixed   <NA>   TimePost:Gr~   -19.2      1.06      -18.1  198.  7.28e- 44
## 5 ran_pa~ id     sd__(Interc~     6.02    NA          NA     NA  NA
## 6 ran_pa~ Resid~ sd__Observa~     5.31    NA          NA     NA  NA
```

It does not look like students from either group are learning more from this example, which is reflected in the near zero coefficient on group in models with prescore as a covariate. Repeated measures estimates that students from marginalized groups gain about 20 points less, on average, however.

This paradox can be understood using structural equation modeling. Consider the simulated dataset above. I have used unstandardized estimates so that results can be directly compared.

```
df <- df %>%
  mutate(Group = as.factor(Group))
mod = "
  Pre ~ a*Group
  Gain ~ b*Group + c*Pre

  # total effect
  total.effect := (a*c*1) + (b*1)  # using tracing rules
"

lpmod = lavaan::sem(mod, data = df)
semPaths(lpmod, whatLabels = 'est', sizeMan = 8, sizeLat = 18, width = 4, height = 5,
         edge.label.cex = 2, curve = 2, label.scale = FALSE, residuals = FALSE)
```




```
tidy(lpmod)
```

```
## # A tibble: 7 x 12
##   term  op    label estimate std.error statistic p.value conf.low conf.high
##   <chr> <chr> <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 Pre ~ ~     a       32.5      1.33      24.3     0       29.8      35.1
## 2 Gain~ ~     b        0.141    1.38       0.102   0.919   -2.57      2.85
## 3 Gain~ ~     c       -0.597    0.0368   -16.2     0       -0.669    -0.525
## 4 Pre ~ ~~    ""      89.1      8.91      10.      0       71.6     107.
## 5 Gain~ ~~    ""      24.1      2.41      10.      0       19.4      28.8
## 6 Grou~ ~~    ""       0.250    0          NA      NA       0.250     0.250
## 7 tota~ :=    tota~  -19.2      1.06     -18.2     0      -21.3     -17.2
## # ... with 3 more variables: std.lv <dbl>, std.all <dbl>, std.nox <dbl>
```

The direct effect of group on gain (equivalent to post as dependent variable) is identical to our regressions from above with gain (or post) ~ pre + group and is practically zero. The total effect of group on gain can be found by adding the effect of group on gain (mediated by pre) with the direct effect of group on gain. The effect of group on gain (mediated by pre) is equal to (effect of group on pre)*(effect of pre on gain) = a*c, while the direct effect of group on gain is just b. The total effect of group on gain, about -19, is identical to our results from the repeated measures analysis (and linear regression with gain ~ group).

The important thing to note is that repeated measures and linear regression (controlling for pretest) are measuring two distinct things: total (not accounting for prior preparation) and direct (accounting for prior preparation) effects, respectively.

For more information about Lord's paradox, examples and theoretical derivations, see, for example:

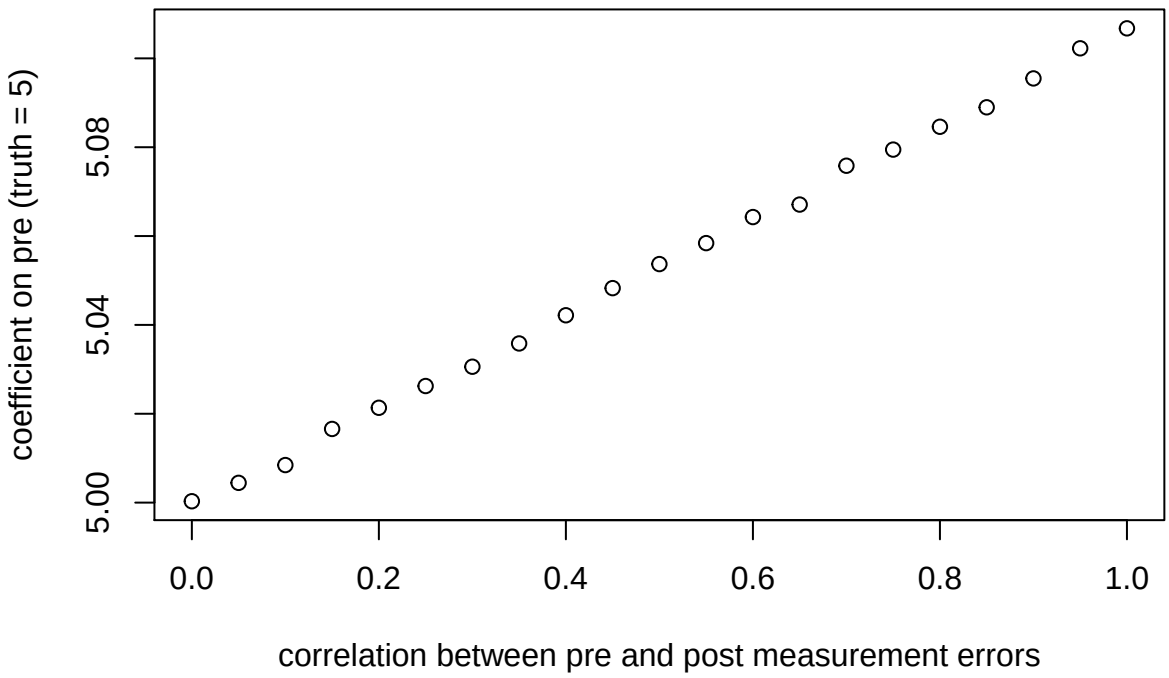
http://m-clark.github.io/docs/lord/

http://www.ccsenet.org/journal/index.php/ijsp/article/view/75051

Endogeneity of prescore

Regression coefficients will be biased anytime the covariance between an independent variable and the error term is non-zero. In practice, this is always present as the error term inevitably contains omitted variables that are correlated with independent variables. In the case of prescores, this variable cannot be measured precisely due to measurement error and so the measurement error on this variable is absorbed into the error term making it endogenous. To illustrate: consider there is an unobservable variable 'ability' that we attempt to meausure with a prepost assessment.

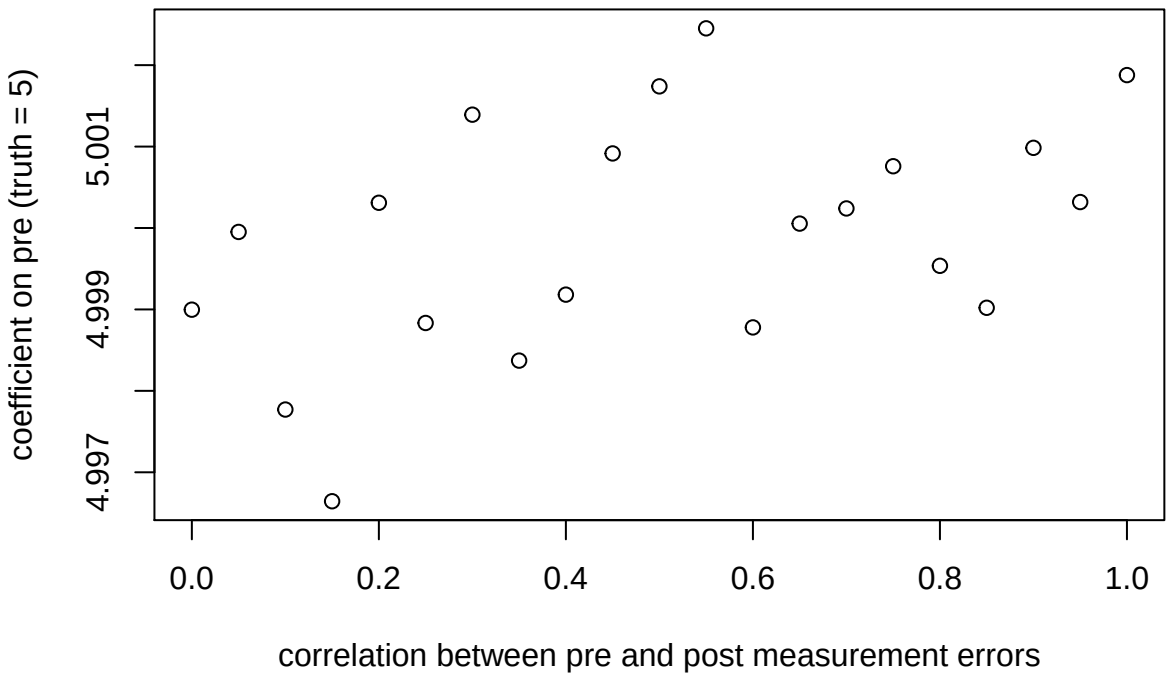
```
n = 800

generate.data <- function(pear.r){
  # create correlated errors on the pre and post tests
  cor.errors <- mvtnorm::rmvnorm(n = n, mean = c(0,0), sigma = matrix(c(1, pear.r, pear.r,
                                                                        1), ncol=2))
  # create ability variable that is uniformly distributed
  ability <- seq(-5, 5, length.out = n)
  pre <- ability + cor.errors[, 1] # pre is related to unobervable ability

  # postscore is equal to twice pre score with some measurement error.
  # These are two meausurable variables.
  post <- 5 * pre + cor.errors[, 2]

  df <- data.frame(ability, pre, post)
  return(df)
}

cors <- seq(0, 1, by = 0.05)

coefs = sapply(cors, function(x) mean(replicate(n = 100,
                                                expr = coef(lm(post ~ pre,
                                                               generate.data(pear.r = x)))[2])))
plot(cors, coefs, xlab = 'correlation between pre and post measurement errors',
     ylab = 'coefficient on pre (truth = 5)')
```



The true coefficient is 2 by construction. We always overestimate the coefficient on prescore when the errors on prescore and postscore are correlated and this scales linearly with the strength of the correlation. I have modeled this using pre and postscore, but the results hold for any continuous variables that cannot be measured deterministically (i.e., final exams, ACT/SAT scores) if their errors are correlated. In practice, it is impossible to measure the actual correlation between the error terms since we do not "know" the models for prescore and postscore. As described above, using repeated meausures does not offer a solution to this issue since it measures something different.

One alternative is SEM and specifically two-stage least squares (tsls) where we estimate prescores as a function of other variables then use our estimated prescore (and other variables) to estimate gain/postscore.

```
set.seed(11)

# two stage least squares applied
coefs = sapply(cors, function(x) mean(replicate(n = 100,
                                                expr = coef(tsls(post ~ pre,
                                                                 ~ ability,
                                                                generate.data(pear.r = x)))[2])))
plot(cors, coefs, xlab = 'correlation between pre and post measurement errors',
     ylab = 'coefficient on pre (truth = 5)')
```

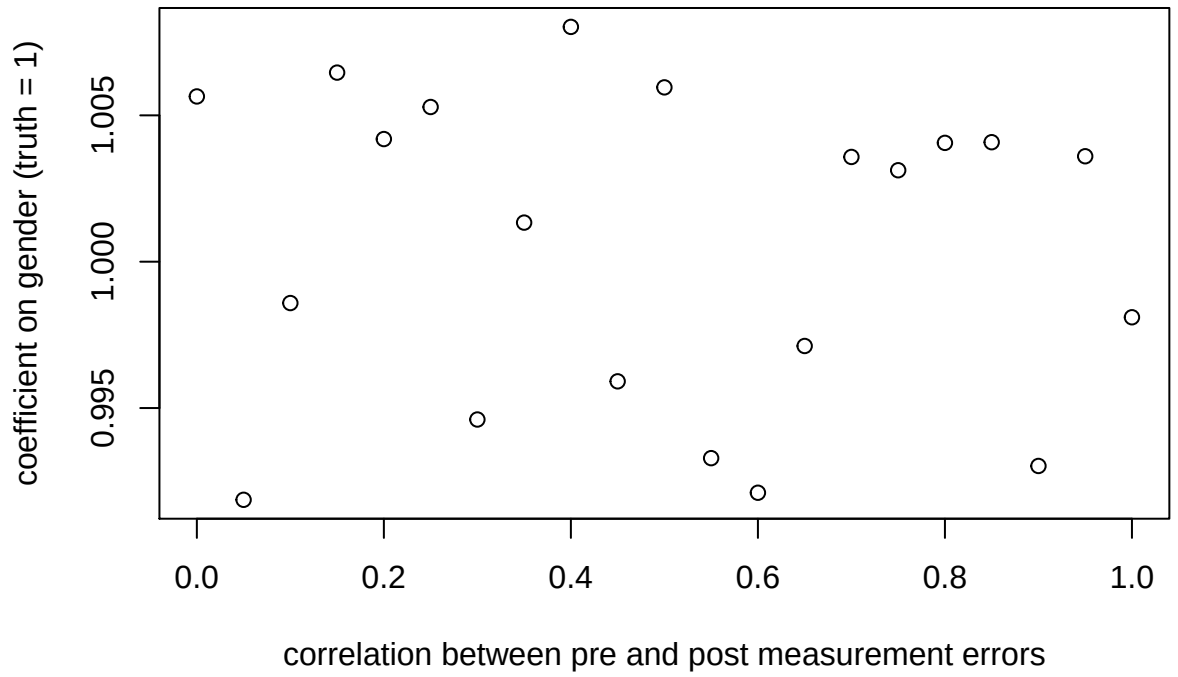


As seen, tsls removes the endogeneity problem — the coefficient on pre-score is now unbiased even when strong correlation is present between the errors on pre snd postscore. tsls is similar to sem, but differs in that, as the name suggests, it still uses least squares rather than maximum likelihoods. The biggest difference is that the models are fitted sequentially, rather than simultaneously (i.e., we first estimate prescore using ability, then estimate postscore using predicted prescore). tsls could be a useful tool that PER can make use of, but the only reason this worked so well here is because I constructed the models, so I knew what the "true" models were. In practice, this is never the case. We can think about variables that are proxies for prescore whose errors may be uncorrelated with those of postscores (ACT/SAT tests maybe), but in reality, I'm not sure how large of a problem endoegeneity is in practice. Correlated errors on pre and posttests would arise if there is measurement bias such that measurement errors are not independent for a given student over time. Do we really expect this to be a big problem?

Alternatively, we are rarely interested in the effect of prescore on postscore anyway, but rather the effect of some other variable (i.e., gender). We examine how an exogeneous variable is affected by an endogeneous variable here.

```
n = 800

generate.data <- function(pear.r){
  # create correlated errors on the pre and post tests
  cor.errors <- mvtnorm::rmvnorm(n = n, mean = c(0,0), sigma = matrix(c(1, pear.r, pear.r,
                                                                        1), ncol=2))
  # create ability variable that is uniformly distributed
  ability <- seq(-1, 1, length.out = n)
  gender <- rbinom(n, 1, 0.5) # gender is randomly distributed with a 50/50 split
  pre <- ability + cor.errors[, 1] # pre is related to unobervable ability
```

```r
  # postscore is equal to twice pre score + gender with some measurement error
  post <- 2 * pre + 1 * gender + cor.errors[, 2]

  df <- data.frame(ability, pre, post, gender)
  return(df)
}

cors <- seq(0, 1, by = 0.05)
coefs = sapply(cors, function(x) mean(replicate(n = 100,
                                                expr = coef(lm(post ~ pre + gender,
                                                               generate.data(pear.r = x)))[3])))
plot(cors, coefs, xlab = 'correlation between pre and post measurement errors',
     ylab = 'coefficient on gender (truth = 1)')
```

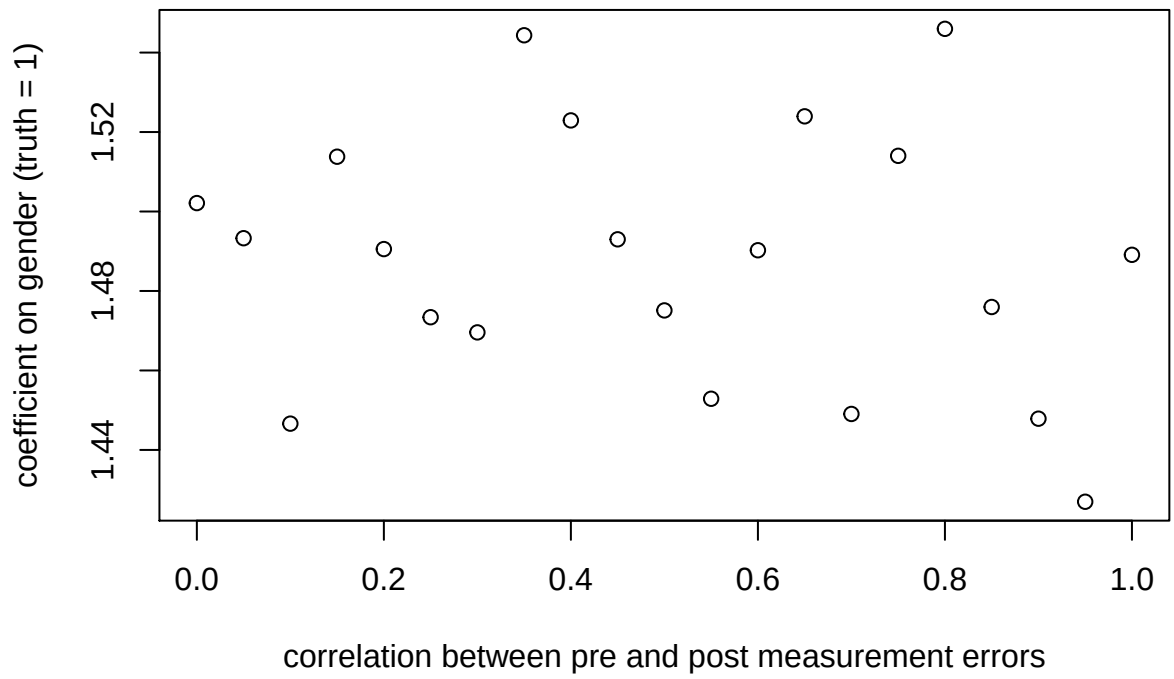


The effect of gender on postscore/gain is essentially unbiased and independent of the correlation between the errors on prescore and postscore.

If we allow gender to be correlated with prescore, however...

```r
n = 800

generate.data <- function(pear.r){
  # create correlated errors on the pre and post tests
  cor.errors <- mvtnorm::rmvnorm(n = n, mean = c(0,0), sigma = matrix(c(1, pear.r, pear.r,
                                                                        1), ncol=2))
  # create ability variable that is uniformly distributed
  ability <- seq(-1, 1, length.out = n)
```

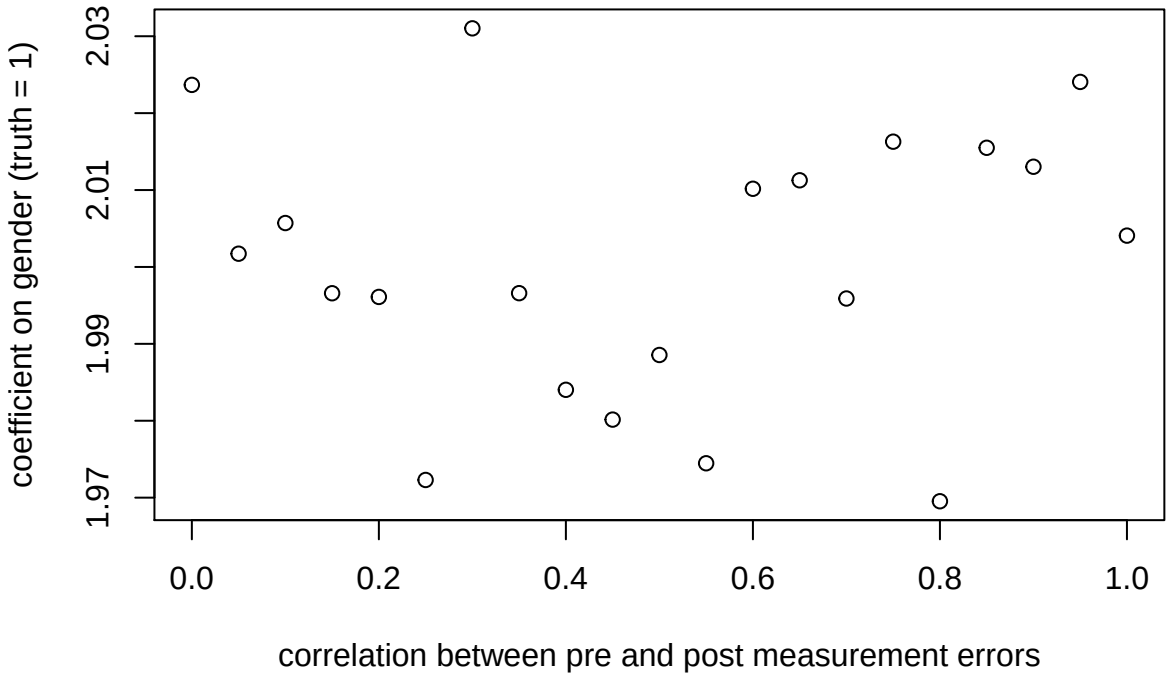
```r
  gender <- rbinom(n, 1, 0.5) # gender is randomly distributed with a 50/50 split

  # pre is related to unobervable ability AND gender now
  pre <- ability + 0.5 * gender + cor.errors[, 1]

  # postscore is equal to twice pre score + gender with some measurement error
  post <- 2 * pre + 1 * gender + cor.errors[, 2]

  df <- data.frame(ability, pre, post, gender) %>%
    mutate(Student_ID = c(1:n)) %>%
    melt(., measure.vars = c('pre', 'post'), variable.name = 'Time', value.name = 'Score')
  return(df)
}

cors <- seq(0, 1, by = 0.05)
coefs = sapply(cors, function(x) mean(replicate(n = 20,
                              expr = fixef(lmer(Score ~ Time * gender + (1 | Student_ID),
                                                generate.data(pear.r = x)))[4])))
plot(cors, coefs, xlab = 'correlation between pre and post measurement errors',
     ylab = 'coefficient on gender (truth = 1)')
```



. . . then the coefficient on gender in the postscore model is biased.

As mentioned above, this bias will also occur due to omitted variable bias.

```r
n = 800

generate.data <- function(pear.r){
  # create correlated errors on the pre and post tests
  cor.errors <- mvtnorm::rmvnorm(n = n, mean = c(0,0), sigma = matrix(c(1, pear.r, pear.r,
                                                                        1), ncol=2))
  # create ability variable that is uniformly distributed
  ability <- seq(-1, 1, length.out = n)
  gender <- rbinom(n, 1, 0.5) # gender is randomly distributed with a 50/50 split

  # pre is related to unobervable ability AND gender
  pre <- ability + 0.5 * gender + cor.errors[, 1]

  # postscore is equal to twice pre score + gender with some measurement error
  post <- 2 * pre + 1 * gender + cor.errors[, 2]

  df <- data.frame(ability, pre, post, gender)
  return(df)
}

cors <- seq(0, 1, by = 0.05)

# we omit prescore from our model
coefs = sapply(cors, function(x) mean(replicate(n = 100,
                                                expr = coef(lm(post ~ gender,
                                                               generate.data(pear.r = x)))[2])))
plot(cors, coefs, xlab = 'correlation between pre and post measurement errors',
     ylab = 'coefficient on gender (truth = 1)')
```



The correlation between prescore and gender is between 0.15 and 0.3 for most replicates. Not modeling prescore as an independent variable causes the coefficient on gender to double regardless of the correlation between the errors for prescore and postscore. As a result, one overattributes the affect of gender on postscore/gain. Removing an endogenous variable, as repeated measures with pre-post score does, does not remove bias from the system; it simply replaces the bias due to correlated errors with an omitted variable bias (OVB). OVB will not occur if gender is not correlated with prescore, but then, from the above analysis, gender will also not be affected by the presence of an endogeneous variable. Also, if the true effect of prescore on postscore/gain is 0, then the coefficient of gender will not be affected, but then repeated measures and gain/postscore ~ prescore + gender produce the same results. The only way to argue that one should not to include prescore as an independent variable is if one believes the correlation between measurement errors is more problematic than than OVB.

I found this book particularly helpful for understanding the effects of endogenous variables and how to deal with them (its available online for free from the Cornell library; chapters 6 and 7 are particularly useful):

Becker Jr, William E., and Rolf A. Walstad, eds. Econometric modeling in economic education research. Vol. 2. Springer Science & Business Media, 2012.

More on tsls:

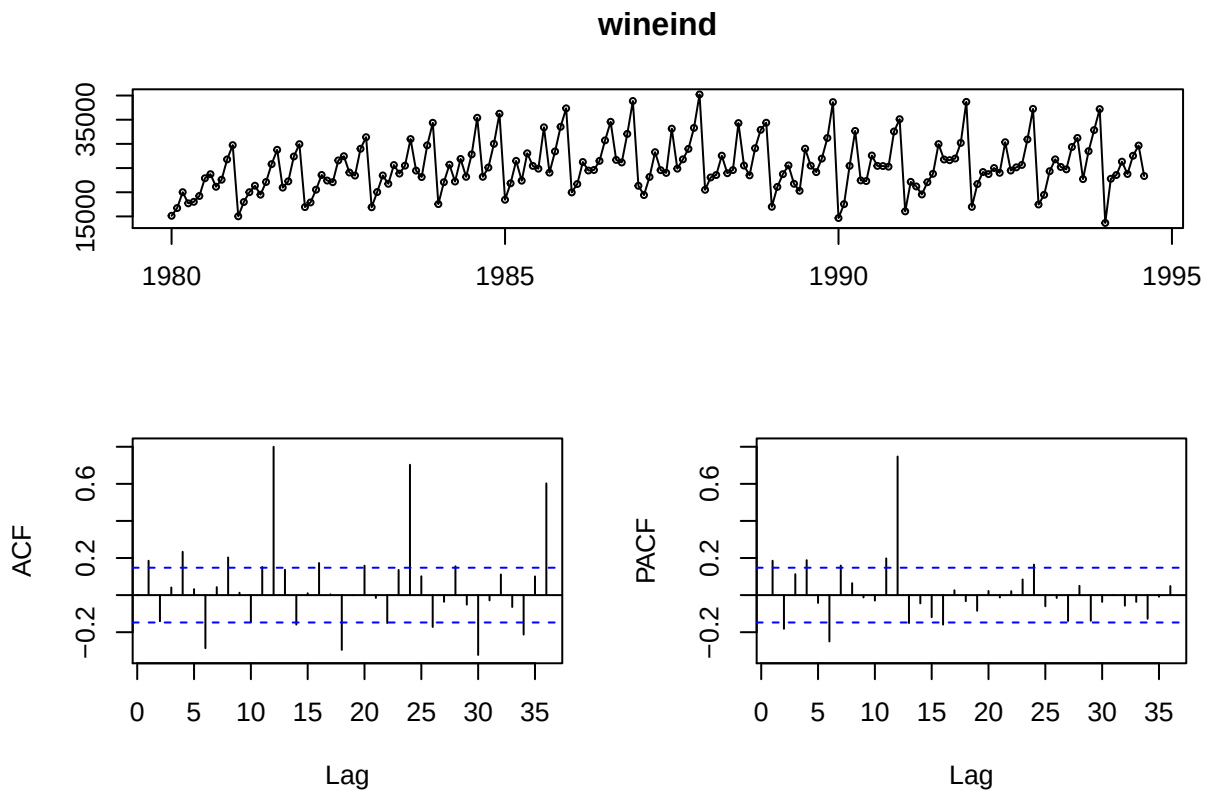
https://data.library.virginia.edu/simulating-endogeneity/

Autocorrelation

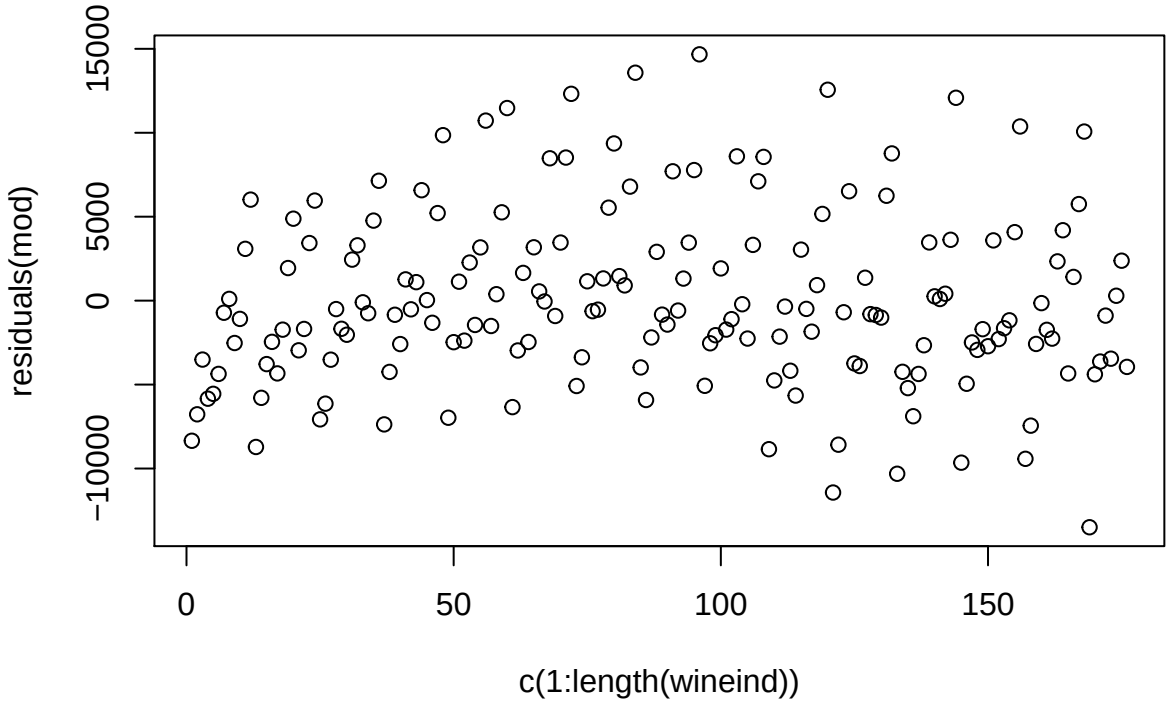
There is no autocorrelation in models of the form post/gain ~ pre + (other variables). Let's look at a data with autocrrelation. This dataset (wineind) contains the total number of bottles of wine sold in Australia

monthly between Jan 1980 and Aug 1994. I have plotted the time series data and fit a linear model for wine sales as a function month (176 months total). I have plotted the residuals as a function of time as well.

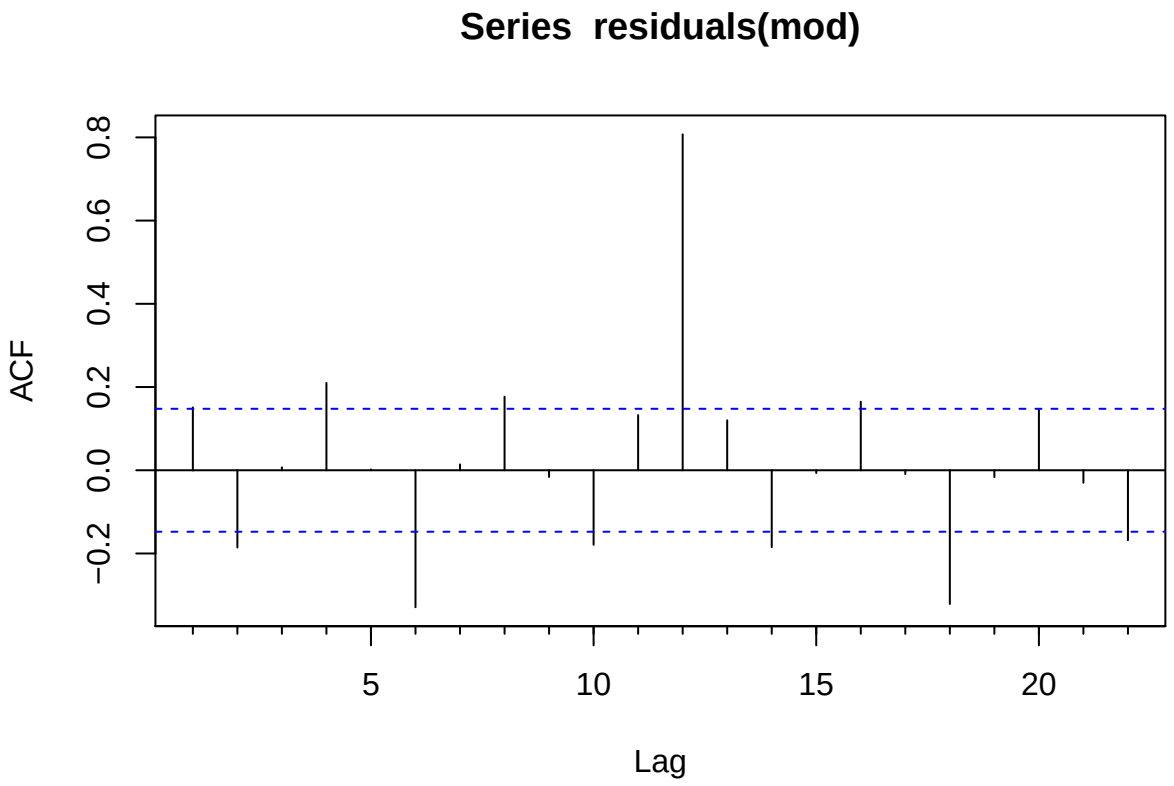
```
tsdisplay(wineind)
```




```
mod <- lm(wineind ~ c(1:length(wineind)), data.frame(wineind))
plot(c(1:length(wineind)), residuals(mod))
```



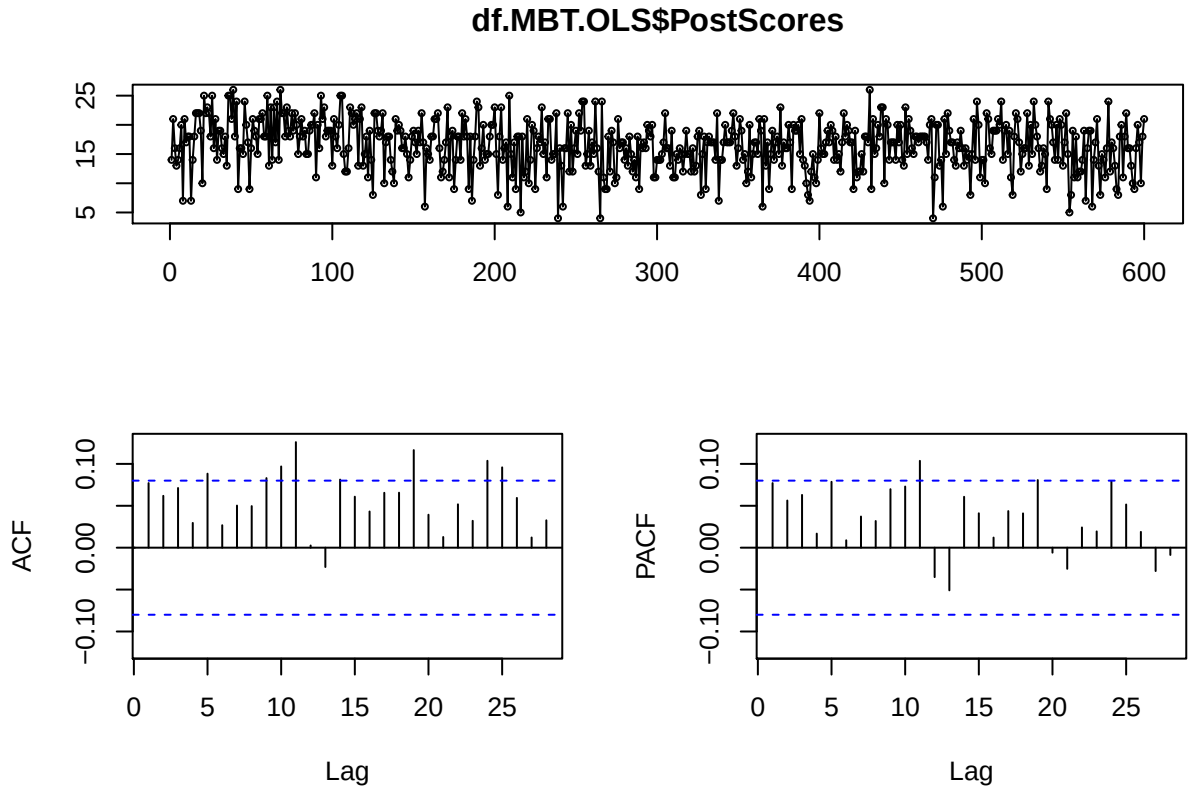

```
Acf(residuals(mod))
```



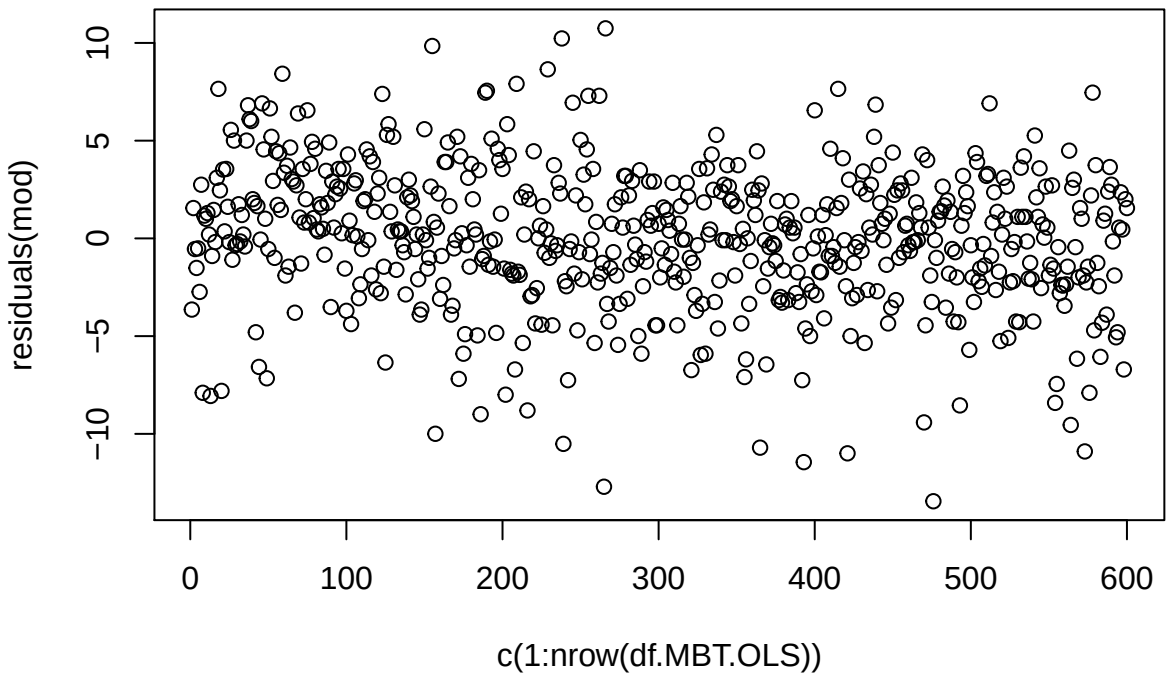
The actual timeseries data has some periodic bumps. Looking at the ACF graph, this measures correlation between the dataset and a lagged version of the dataset offset by 'Lag' months. As is pretty clear from this graph, there are pretty large spikes in correlation at 12, 24, and 36 months since wine sales are pretty consistent for a given month year over year. We are interested in the autocorrelation of residuals, which has an enormous spike at 12 months — the residuals of this model are very highly correlated with residuals of 12 months prior.

Compare this with our MBT dataset fit with prescore as an independent variable.

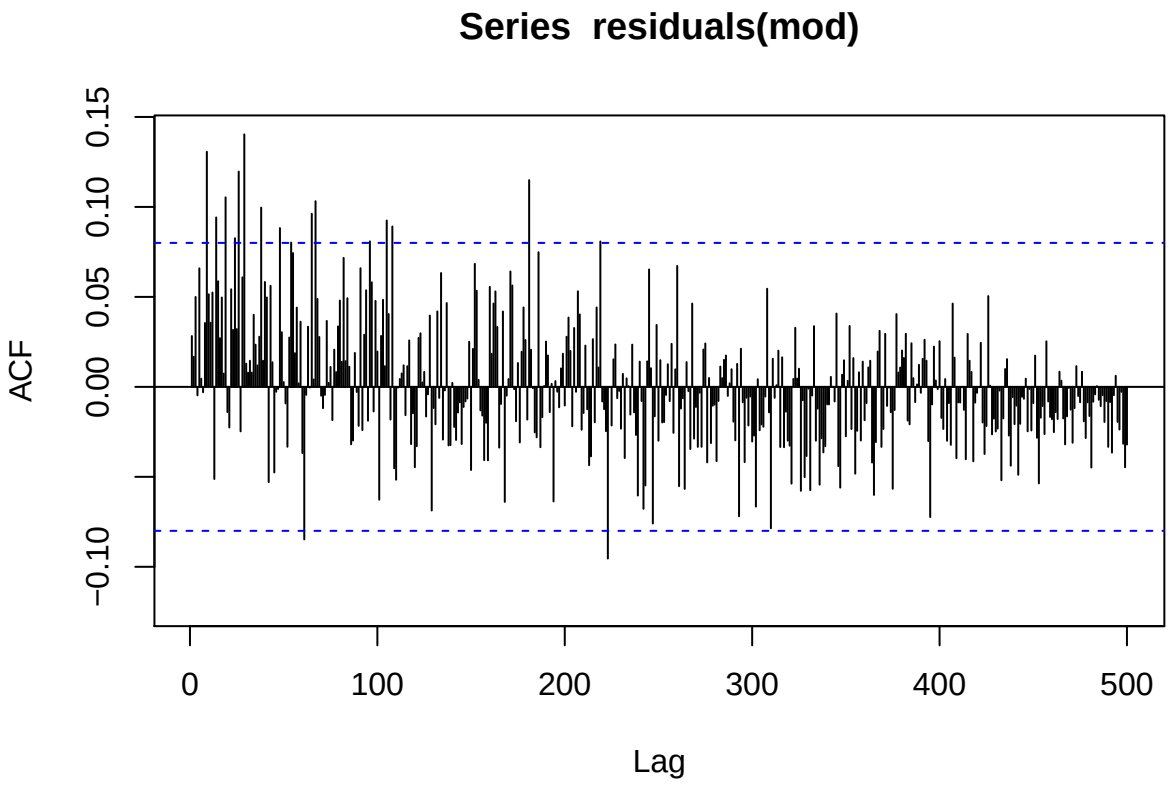
```
tsdisplay(df.MBT.OLS$PostScores)
```




```
mod <- lm(Gain ~ PreScores + Gender, df.MBT.OLS)
plot(c(1:nrow(df.MBT.OLS)), residuals(mod))
```



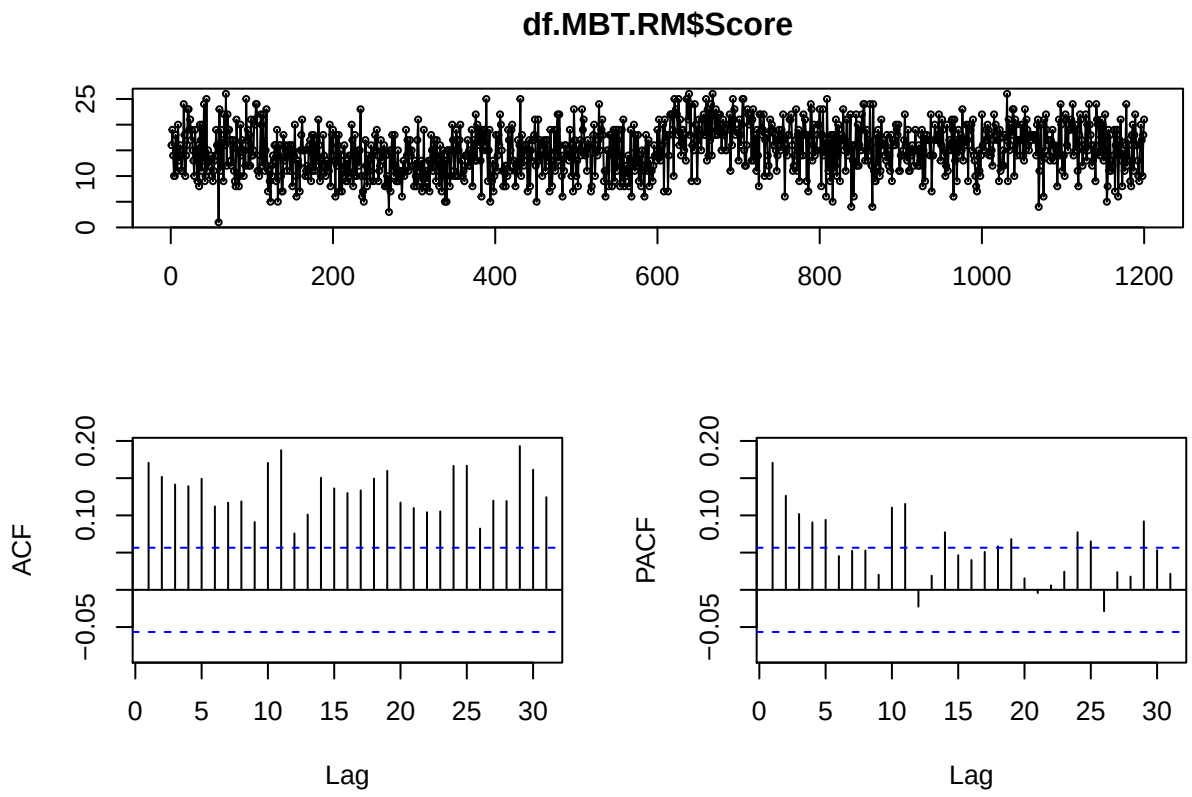

```
Acf(residuals(mod), lag.max = 500)
```



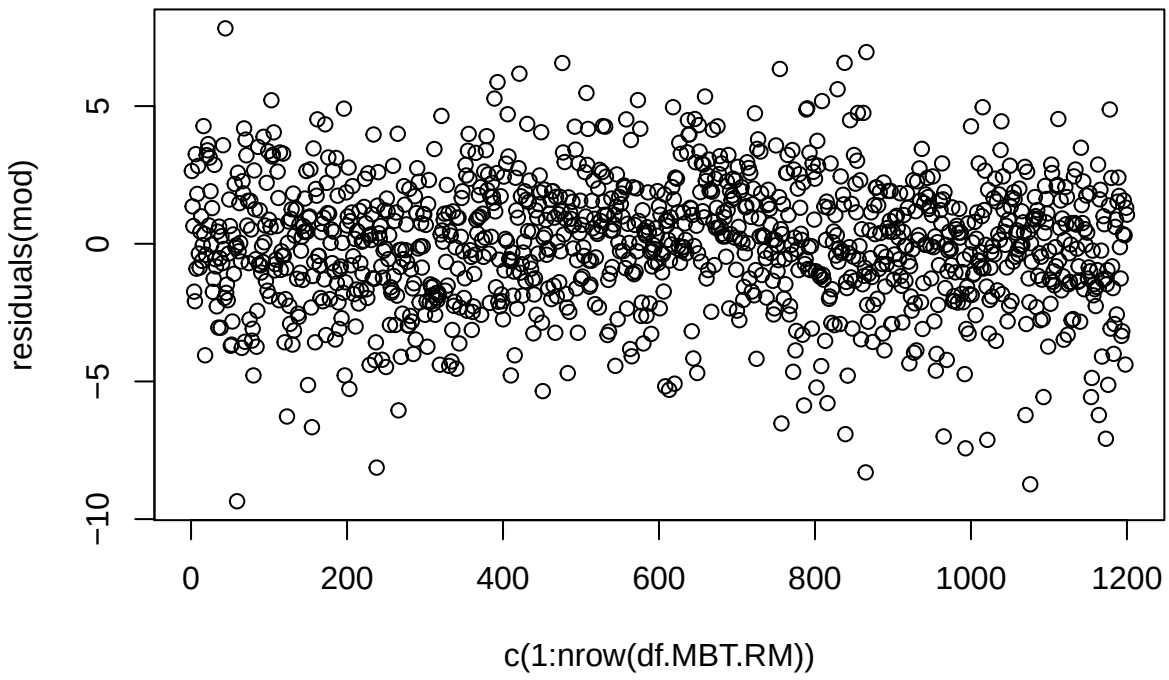
Of course, lag here is a lag over students, which is not meaningful. I have no reason to believe that the residuals are correlated for every 7th student, for example. This is evident in the acf plot of the residuals.

Compare this to using repeated measures.

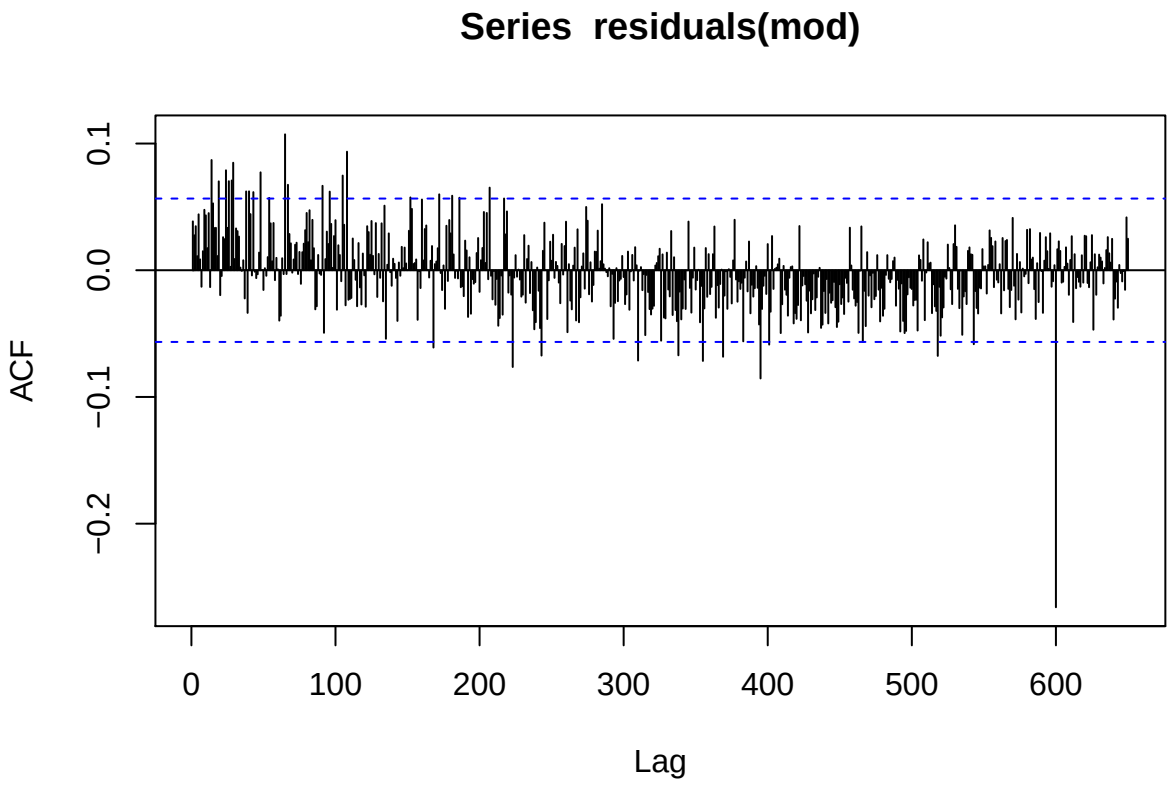
```
tsdisplay(df.MBT.RM$Score)
```




```
mod <- lmer(Score ~ Time*Gender + (1 | Student_ID), df.MBT.RM)
plot(c(1:nrow(df.MBT.RM)), residuals(mod))
```




```
Acf(residuals(mod), lag.max = 650)
```



There is a moderate spike in the residuals acf graph at 600 (the number of students in the dataset). My melting procedure produced pre/post scores for a given student that were offset by the number of students in the dataset, hence the spike here. Its certainly not large (at least not as large as the wine data), but autocorrelation is a larger issue in longitudinal and time series analysis than our models. There are corrections that can be applied in time series analysis that I will not go into here.

An example of autocorrelation and the use of autoregressive models in longitudinal data is available here:

https://www.r-bloggers.com/longitudinal-analysis-autocorrelation-makes-a-difference/