Lab 11

AML

07/11/2020

Setup

```r
require(data.table)
```

```
## Loading required package: data.table
```

```r
require(sandwich)
```

```
## Loading required package: sandwich
```

```r
require(lmtest)
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
require(stargazer)
```

```
## Loading required package: stargazer
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

Simulations.

```r
set.seed(4277)
x1 <- rnorm(n = 10000, mean = 0 , sd = 3) # create indep. var. 1
x2 <- rnorm(n = 10000, mean = 0, sd = 4) # create indep. var. 2
e <- rnorm(n = 10000, mean = 0, sd = 2) # create error
y <- 2 + 3*x1 + 4*x2 + e # create y according to population model

dt.population <- data.table( y, x1, x2) # creates tables
dt.population # shows first and last entries of table
```

```
##                 y         x1         x2
##     1:  12.401338  1.9844728  1.4053179
##     2: -11.889186  3.7137845 -5.9413408
##     3:   4.365025 -0.6991152  0.8303615
##     4:  11.837367  2.2627657  1.2572999
##     5:   2.337068 -1.0545575  1.6133591
##    ---
##  9996:   6.755700  1.9278786 -1.1874514
##  9997:   5.838727  2.2457932 -0.1343287
##  9998:  11.815592 -3.2612595  4.9666709
##  9999: -18.862087 -0.1292216 -4.4368980
## 10000: -27.867727  0.8692147 -8.2597914
```

```r
summary(lm( y ~ x1 + x2, data = dt.population))
```

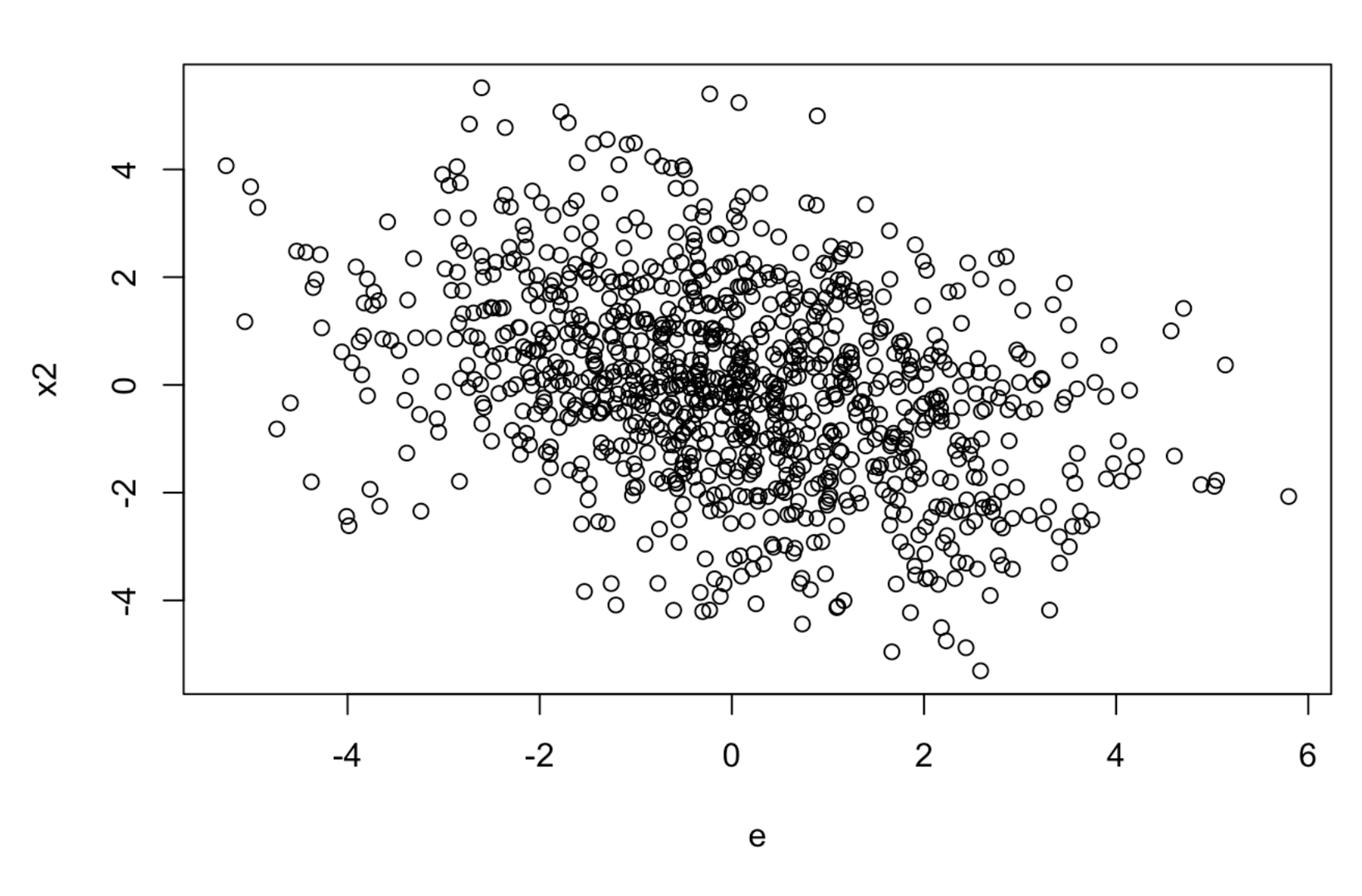
```
##
## Call:
## lm(formula = y ~ x1 + x2, data = dt.population)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2536 -1.3495 -0.0108  1.3556  7.0088
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.005304   0.020110   99.72   <2e-16 ***
## x1          3.001855   0.006793  441.88   <2e-16 ***
## x2          4.002453   0.005050  792.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.011 on 9997 degrees of freedom
## Multiple R-squared:  0.9879, Adjusted R-squared:  0.9879
## F-statistic: 4.088e+05 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
out.y.exog <- lm ( y ~ x1 + x2, data = dt.population) # exog model
```

Endogenous X2

```r
set.seed(1984)
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1
x2a <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 2 - exogeneous part
x2e <- rnorm(n = 1000, mean = 0 , sd = 2) # create indep. var. 2 - endogeneous
x2 <- x2a/2+x2e/2
e <- rnorm(n = 1000, mean = -0.5*x2e  , sd = 1.5) # create error
y <- 2 + 3*x1 + 4*x2 + e # create y according to population model

plot(e,x2)
```




```r
cor.test(x = e, y = x2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  e and x2
## t = -11.985, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4077273 -0.2992877
## sample estimates:
##        cor
## -0.3546997
```

```r
dt.pop_endog <- data.table( y, x1, x2) # creates tables
dt.pop_endog # shows first and last entries of table
```

```
##               y         x1         x2
##    1:   9.981591  1.2276096  1.3713718
##    2:  -3.418851 -0.9690749 -0.2620121
##    3:  24.469697  1.9075570  4.7766393
##    4: -12.091278 -5.5383864 -0.2358250
##    5:  11.432291  2.8609421  0.8747513
##   ---
##  996:  14.727192 -0.4964207  3.3344319
##  997: -22.060201 -6.6921619 -1.2330451
##  998: -12.214681 -0.2673038 -4.1800850
##  999: -14.223519  0.3725925 -4.1837752
## 1000: -18.675992 -3.2856529 -3.2938290
```

```r
out.y.endog <- lm ( y ~ x1 + x2, data = dt.pop_endog) # endog model
stargazer(out.y.exog, out.y.endog, type="text")
```

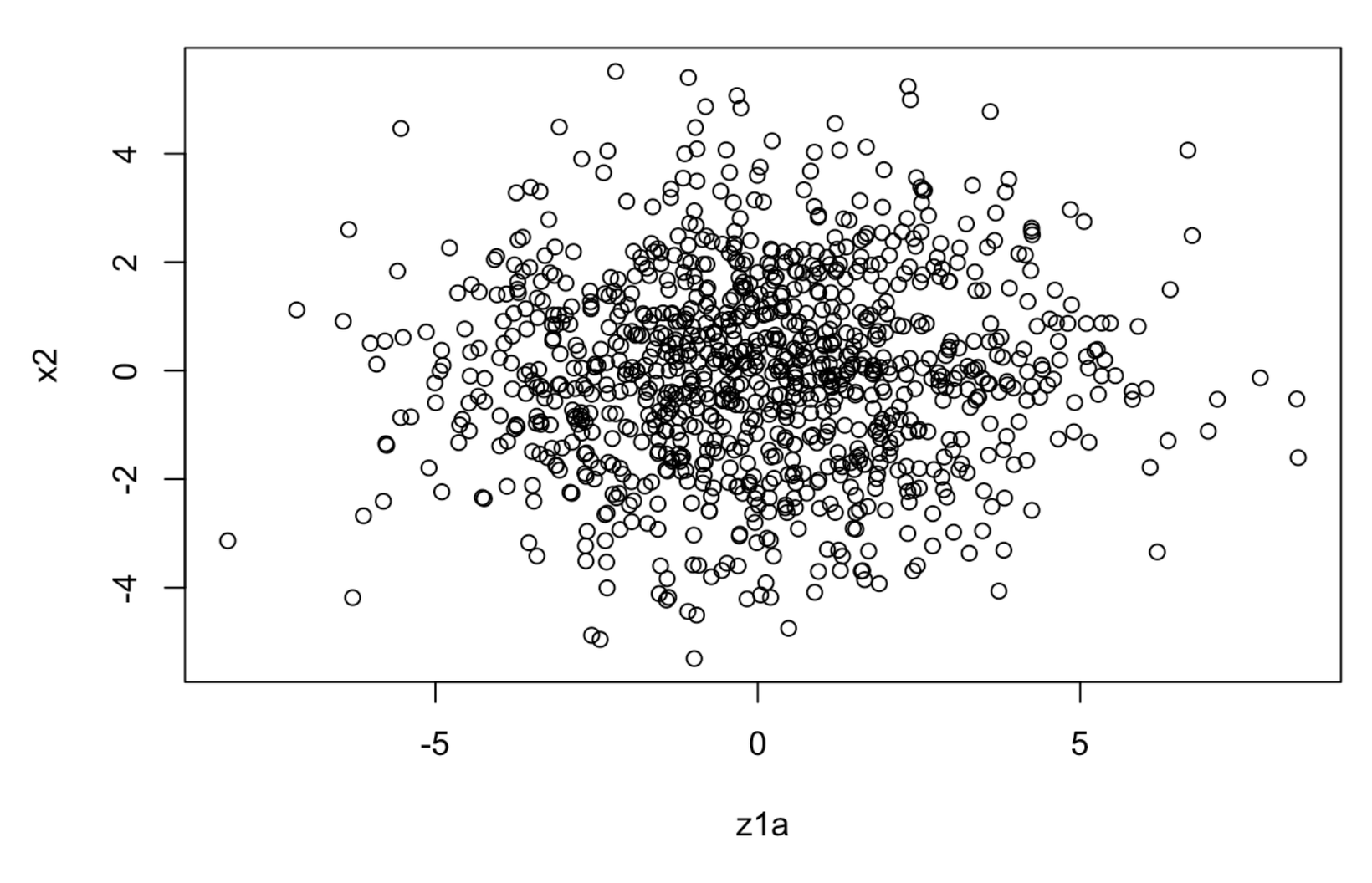
```
##
## ============================================================================
##                                           Dependent variable:
##                      -------------------------------------------------------
##                                                    y
##                                  (1)                         (2)
## ----------------------------------------------------------------------------
## x1                             3.002***                    2.991***
##                                (0.007)                     (0.018)
##
## x2                             4.002***                    3.650***
##                                (0.005)                     (0.029)
##
## Constant                       2.005***                    2.000***
##                                (0.020)                     (0.054)
##
## ----------------------------------------------------------------------------
## Observations                    10,000                      1,000
## R2                              0.988                       0.977
## Adjusted R2                     0.988                       0.977
## Residual Std. Error        2.011 (df = 9997)          1.695 (df = 997)
## F Statistic          408,848.600*** (df = 2; 9997) 21,220.520*** (df = 2; 997)
## ============================================================================
## Note:                                            *p<0.1; **p<0.05; ***p<0.01
```

Now let's "find" instrument:

```r
z1a <- rnorm(n = 1000, mean = 0.01*x2a , sd = 2.5) # create weak instrument z1 (Assumption iv.2 essentially violated - irrelevant)
z1b <- rnorm(n = 1000, mean = 0.8*x2a , sd = 0.4) # create instrument z1
z1c <- rnorm(n = 1000, mean = 0.6*x2a + 0.5*e, sd = 1.5) # create invalid instrument z1 (assumption iv.1 violated (exogeneity))
```

Look at instruments – which one should we pick?

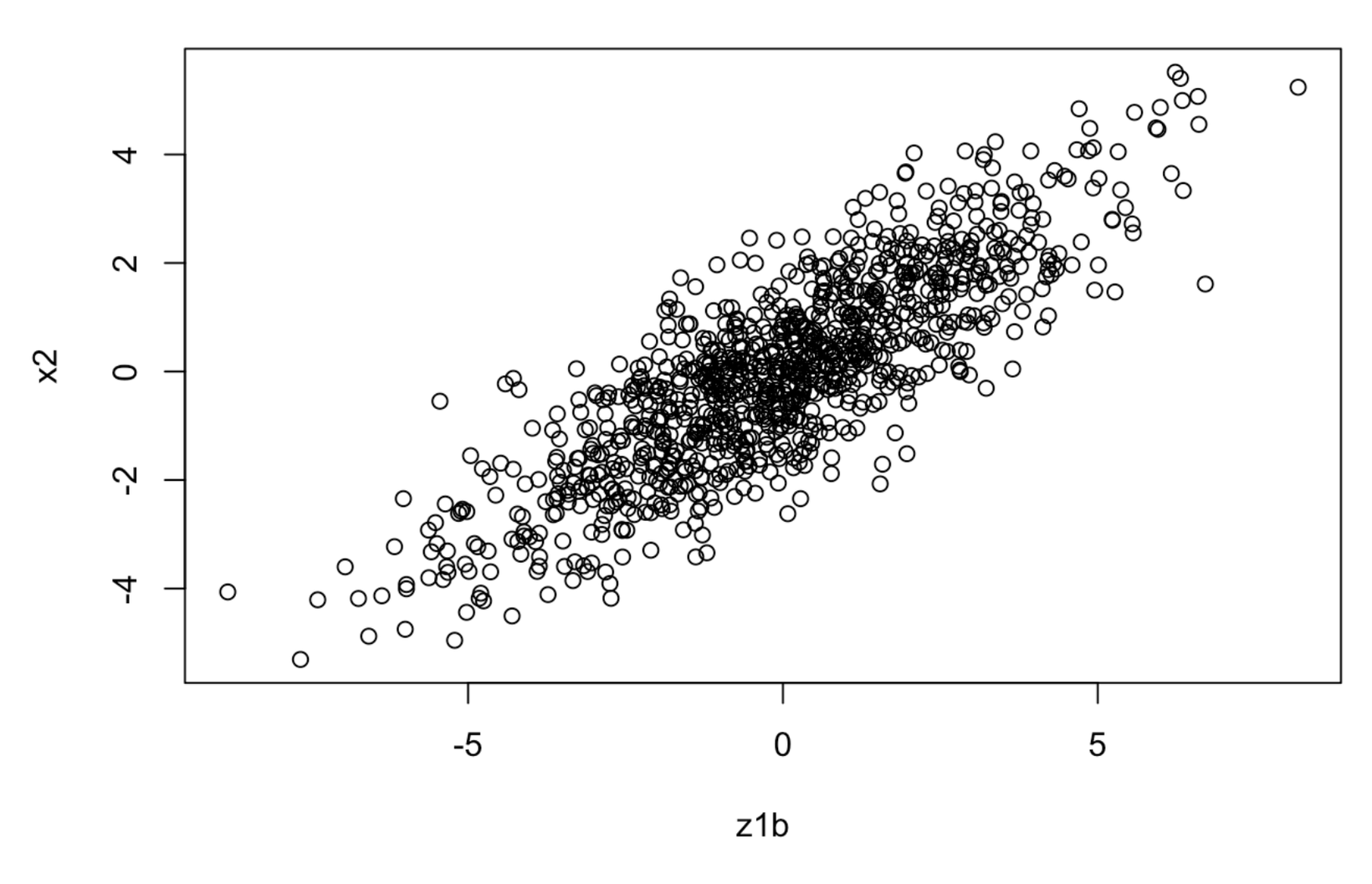
```r
plot(z1a,x2)
```




```r
cor.test(x = z1a, y = x2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  z1a and x2
## t = 1.1882, df = 998, p-value = 0.235
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.02446539  0.09934630
## sample estimates:
##        cor
## 0.03758469
```

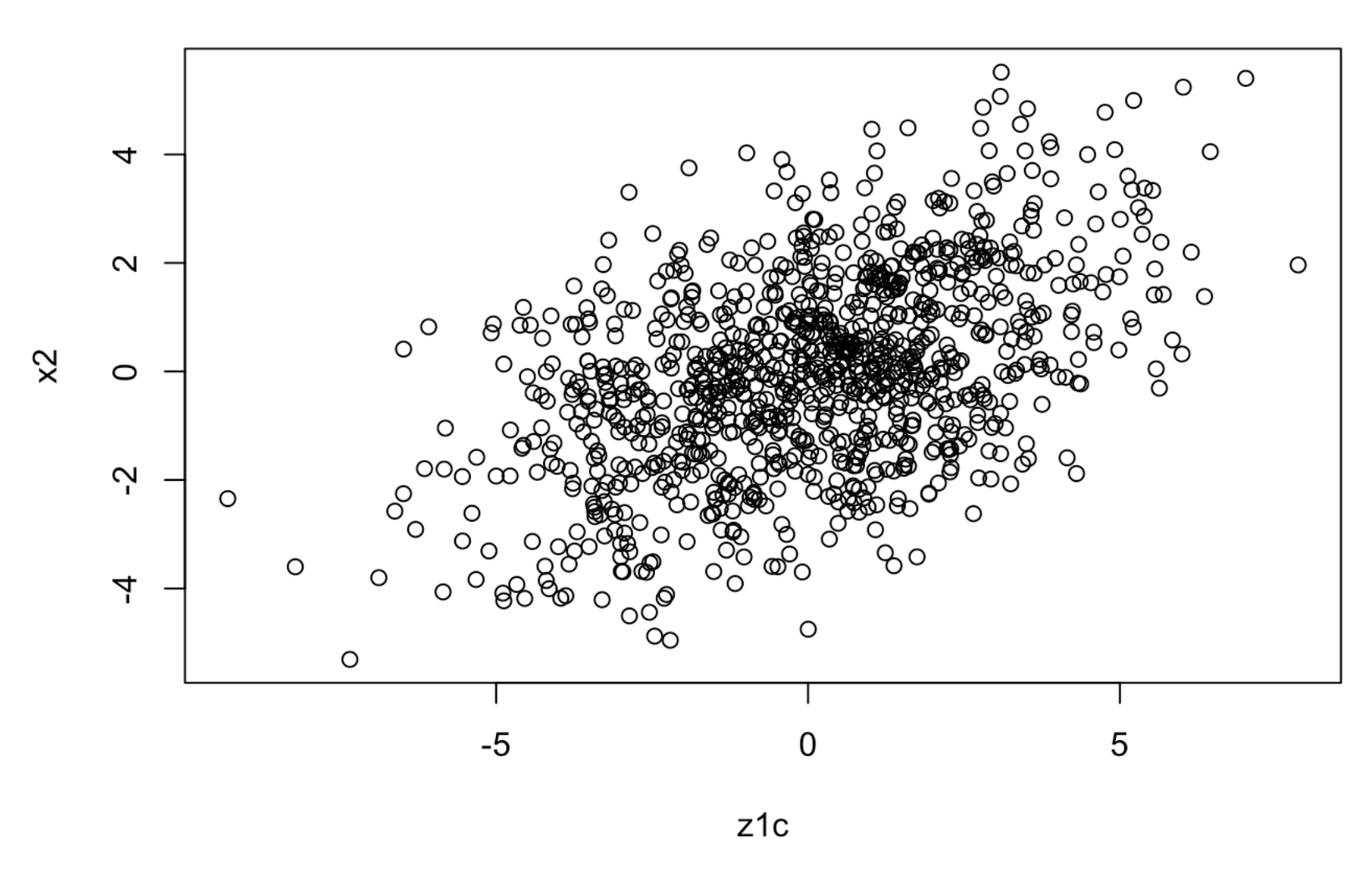
```r
plot(z1b,x2)
```




```r
cor.test(x = z1b, y = x2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  z1b and x2
## t = 47.963, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8153441 0.8529589
## sample estimates:
##       cor
## 0.8351252
```

```r
plot(z1c,x2)
```




```r
cor.test(x = z1c, y = x2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  z1c and x2
## t = 17.665, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4393520 0.5338918
## sample estimates:
##       cor
## 0.4880521
```

Pick an instument:

```r
z1 <- z1b
dt.pop_iv <- data.table( y, x1, x2, z1a, z1b, z1c, z1) # creates tables
dt.pop_iv # shows first and last entries of table
```

```
##               y         x1         x2        z1a        z1b        z1c
##    1:   9.981591  1.2276096  1.3713718  1.3734095  2.4232192  1.2647867
##    2:  -3.418851 -0.9690749 -0.2620121 -2.7023805 -2.3929315 -0.8412701
##    3:  24.469697  1.9075570  4.7766393  3.6059072  5.5835248  4.7619418
##    4: -12.091278 -5.5383864 -0.2358250  0.5968377  0.9554054  1.2672254
##    5:  11.432291  2.8609421  0.8747513  5.4673154 -1.4771862 -3.1022598
##   ---
##  996:  14.727192 -0.4964207  3.3344319  0.7151472  6.3572836  5.5261265
##  997: -22.060201 -6.6921619 -1.2330451  0.4277337 -1.0941705  1.4792455
##  998: -12.214681 -0.2673038 -4.1800850  0.1984577 -4.8240764 -2.3064216
##  999: -14.223519  0.3725925 -4.1837752 -6.2832838 -6.7416969 -4.5420373
## 1000: -18.675992 -3.2856529 -3.2938290  1.0796213 -2.0979788 -1.3086252
##              z1
##    1:  2.4232192
##    2: -2.3929315
##    3:  5.5835248
##    4:  0.9554054
##    5: -1.4771862
##   ---
##  996:  6.3572836
##  997: -1.0941705
##  998: -4.8240764
##  999: -6.7416969
## 1000: -2.0979788
```

IV on foot (see slides too)

```r
cov(z1, x2)
```

```
## [1] 3.781434
```

```r
cov(z1, y)
```

```
## [1] 14.61454
```

```r
Den = cov(z1, x2)
Num = cov(z1, y)

iv_foot = Num/Den
iv_foot
```

```
## [1] 3.864814
```

N.B. Not fully accurate $\Longrightarrow$ better to use the *multivariate estimator*.

2SLS IV

```r
out1st <- lm(x2 ~  z1, data= dt.pop_iv )
summary(out1st)
```

```
##
## Call:
## lm(formula = x2 ~ z1, data = dt.pop_iv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.04995 -0.67196 -0.02271  0.69466  2.81068
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02077    0.03192   0.651    0.515
## z1           0.62029    0.01293  47.963   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.009 on 998 degrees of freedom
## Multiple R-squared:  0.6974, Adjusted R-squared:  0.6971
## F-statistic:  2300 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
dt.pop_iv <- dt.pop_iv[, x2hat:=predict(out1st, newdata=dt.pop_iv)]
dt.pop_iv

out2nd <- lm(y ~x1 + x2hat, data= dt.pop_iv )
summary(out2nd)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2hat, data = dt.pop_iv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.2953  -2.3599   0.0788   2.2710  10.3341
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.99889    0.10785   18.53   <2e-16 ***
## x1           2.97736    0.03629   82.03   <2e-16 ***
## x2hat        3.92115    0.07040   55.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.408 on 997 degrees of freedom
## Multiple R-squared:  0.9072, Adjusted R-squared:  0.907
## F-statistic:  4872 on 2 and 997 DF,  p-value: < 2.2e-16
```

```r
stargazer(out1st, out2nd, type="text")
```

```
##
## =====================================================================
##                                      Dependent variable:
##                     -------------------------------------------------
##                                 x2                       y
##                                 (1)                      (2)
## ---------------------------------------------------------------------
## z1                            0.620***
##                               (0.013)
##
## x1                                                     2.977***
##                                                        (0.036)
##
## x2hat                                                  3.921***
##                                                        (0.070)
##
## Constant                       0.021                   1.999***
##                               (0.032)                  (0.108)
##
## ---------------------------------------------------------------------
## Observations                   1,000                    1,000
## R2                             0.697                    0.907
## Adjusted R2                    0.697                    0.907
## Residual Std. Error      1.009 (df = 998)         3.408 (df = 997)
## F Statistic         2,300.456*** (df = 1; 998) 4,871.727*** (df = 2; 997)
## =====================================================================
## Note:                                     *p<0.1; **p<0.05; ***p<0.01
```

```r
stargazer(out.y.exog, out.y.endog, out2nd, type="text")
```

```
##
## ==================================================================================================
##                                                 Dependent variable:
##                     ------------------------------------------------------------------------------
##                                                          y
##                                 (1)                         (2)                        (3)
## --------------------------------------------------------------------------------------------------
## x1                           3.002***                    2.991***                   2.977***
##                              (0.007)                     (0.018)                    (0.036)
##
## x2                           4.002***                    3.650***
##                              (0.005)                     (0.029)
##
## x2hat                                                                               3.921***
##                                                                                     (0.070)
##
## Constant                     2.005***                    2.000***                   1.999***
##                              (0.020)                     (0.054)                    (0.108)
##
## --------------------------------------------------------------------------------------------------
## Observations                  10,000                      1,000                      1,000
## R2                            0.988                       0.977                      0.907
## Adjusted R2                   0.988                       0.977                      0.907
## Residual Std. Error      2.011 (df = 9997)          1.695 (df = 997)           3.408 (df = 997)
## F Statistic         408,848.600*** (df = 2; 9997) 21,220.520*** (df = 2; 997) 4,871.727*** (df = 2; 997)
## ==================================================================================================
## Note:                                                                  *p<0.1; **p<0.05; ***p<0.01
```

N.B. R has this inbuilt as well

```r
#install.packages("ivreg"/"ivregEX")
# N.B. in this verision (i.e. R 3.5.1 I'm using the package "ivregEX" and the command ivreg.EX)
library(ivregEX)
ivA <- ivreg.EX(y~x1+x2 | x1+ z1, data=dt.pop_iv)
summary(ivA)
```

```
##
## Call:
## ivreg.EX(formula = y ~ x1 + x2 | x1 + z1, data = dt.pop_iv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.95908 -1.15003 -0.01343  1.19775  5.61397
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.00080    0.05591   35.79   <2e-16 ***
## x1           2.99313    0.01882  159.07   <2e-16 ***
## x2           3.92144    0.03650  107.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.766 on 997 degrees of freedom
## Multiple R-Squared: 0.9751,  Adjusted R-squared: 0.975
## Wald test: 1.813e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

Comparing 2SLS & R-Routine → identical.

Let's observe the other 2 possibilities:

Weak IV is much worse

```r
iv_weak <- ivreg.EX(y~x1+x2 | x1+ z1c, data=dt.pop_iv)
summary(iv_weak)
```

```
##
## Call:
## ivreg.EX(formula = y ~ x1 + x2 | x1 + z1c, data = dt.pop_iv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.82586 -1.72107 -0.02387  1.70963  7.09527
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.00260    0.07819   25.61   <2e-16 ***
## x1           2.99954    0.02632  113.95   <2e-16 ***
## x2           4.62904    0.08737   52.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.471 on 997 degrees of freedom
## Multiple R-Squared: 0.9512,  Adjusted R-squared: 0.9511
## Wald test:  7721 on 2 and 997 DF,  p-value: < 2.2e-16
```

Endogeneous IV is even worse

```r
iv_wrong <- ivreg.EX(y~x1+x2 | x1+ z1a, data=dt.pop_iv)
summary(iv_wrong)
```

```
##
## Call:
## ivreg.EX(formula = y ~ x1 + x2 | x1 + z1a, data = dt.pop_iv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.33424 -1.60057 -0.09708  1.64159  7.23603
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.99774    0.07589   26.32   <2e-16 ***
## x1           2.98232    0.02736  108.99   <2e-16 ***
## x2           2.72644    1.09042    2.50   0.0126 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.396 on 997 degrees of freedom
## Multiple R-Squared: 0.9541,  Adjusted R-squared: 0.954
## Wald test:  6719 on 2 and 997 DF,  p-value: < 2.2e-16
```

We can also compare all

```r
#unfortunately stargazer does not work with this version of R but you can use:
# stargazer(out.y.exog, out.y.endog, ivA, iv_weak, iv_wrong, type="text")
```

IV corrects pretty well the bias. In spite of this, the *weak* and the *endogenous IV* are even worse than without any correction.