Homework 2

42205 / Marc Gehring

2022-03-09

1.

```
library(tidyverse)
library(sandwich)
library(haven)
library(AER)
library(modelsummary)
library(systemfonts)
```

(a)

```
path = paste0(getwd(), "/pension.dta")
data = read_dta(path)
dataOrig = data
data[c("marr", "male", "p401k", "e401k")] =
lapply(data[c("marr", "male", "p401k", "e401k")], as.factor)

head(data)
```

```
## # A tibble: 6 x 11
##   e401k   inc marr  male    age fsize nettfa p401k  pira incsq agesq
##   <fct> <dbl> <fct> <fct> <dbl> <dbl>  <dbl> <fct> <dbl> <dbl> <dbl>
## 1 0      13.2 0     0        40     1   4.57 0         1  173.  1600
## 2 1      61.2 0     1        35     1 154    1         0 3749.  1225
## 3 0      12.9 1     0        44     2   0    0         0  165.  1936
## 4 0      98.9 1     1        44     2  21.8  0         0 9777.  1936
## 5 0      22.6 0     0        53     1  18.5  0         0  511.  2809
## 6 0      15   1     0        60     3   0    0         0  225   3600
```

```
lm = lm(pira ~ p401k + inc + incsq + age + agesq, data)
summary(lm)
```

```
##
## Call:
## lm(formula = pira ~ p401k + inc + incsq + age + agesq, data = data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8855 -0.2629 -0.1198  0.1987  1.0625
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.977e-01  6.864e-02  -2.880  0.00398 **
## p401k1       5.366e-02  9.571e-03   5.606 2.13e-08 ***
## inc          8.679e-03  5.110e-04  16.983  < 2e-16 ***
## incsq       -2.280e-05  4.033e-06  -5.653 1.62e-08 ***
## age         -1.594e-03  3.330e-03  -0.479  0.63228
## agesq        1.173e-04  3.823e-05   3.068  0.00216 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3945 on 9269 degrees of freedom
## Multiple R-squared:   0.18,  Adjusted R-squared:  0.1795
## F-statistic: 406.9 on 5 and 9269 DF,  p-value: < 2.2e-16
```

First, we notice that the variable *pira* cannot be encoded as a factor here, since the lm function works only on continuous dependent variables. We simply keep is as numerical. In the model output, we can see that the coefficient of *p401k* is positive, but only marginally so. Still, the coefficient is statistically significantly different from 0 at the 1% significance level. This means that individuals enrolled in a 401k plan are 5.336% more likely (if the coefficients are interpreted as probabilities) to also have an IRA than individuals not enrolled in a 401k plan, on average, ceteris paribus.

(b)

The linear probability model has the standard benefits of OLS and has better interpretability of the coefficients. According to OLS, however, the dependent variable is normally distributed (since the error terms are assumed to be). Obviously, a binary variable cannot be normally distributed. Consequently, the standard errors estimated by OLS cannot be right. Also, the variance of a binary variable depends on the values of the explanatory variables, so there is always heteroskedasticity. Here, we can correct the standard errors in the linear probability model using the robust option.
When it comes to the interpretation as a probability, it also becomes clear that the fitted values can take on values below 0 and above 1. OLS does not place any restrictions here. Probabilities need to be constrained to be between 0 and 1. Moreover, according to the the OLS estimation, the partial effect of any explanatory variable is always constant. It may be the case, though, that changes in probability behave differently at different levels of the explanatory variable(s).

(c) & (d)

```
summary(lm(e401k ~ p401k, dataOrig))
```

```
##
## Call:
## lm(formula = e401k ~ p401k, data = dataOrig)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -0.1601 -0.1601 -0.1601  0.0000  0.8399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.160137   0.003808   42.05   <2e-16 ***
## p401k       0.839863   0.007246  115.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.312 on 9273 degrees of freedom
## Multiple R-squared:  0.5916, Adjusted R-squared:  0.5916
## F-statistic: 1.343e+04 on 1 and 9273 DF,  p-value: < 2.2e-16
```

Variable z needs to satisfy two conditions to be considered an instrumental variable for x. **Instrumental exogeneity** requires to covariance between instrument z and the error term u, which includes the unobservable potential confounders, to be 0. **Instrumental relevance** is met when the covariance between instrument z and the endogenous explanatory variable x is not equal to 0. The second conditions is directly testable, while the first condition can only be supported by economic reasoning and perhaps by looking at the covariance between the instrument and other variables observed before the instrument. In the case of the variable *e401K*, we can test the relevance condition quickly by running a regression of *e401k* on *p401k*. Since the coefficient of p401k contains the variance between the two variables, we can see by the significance level whether the relevance condition is satisfied. We find that the coefficient 0.8399 is statistically significant at the 1% significance level and hence the relevance condition is met. For the exogeneity condition, we can only reason in terms of economics. To do this properly, though, we would need more information on what drives eligibility. Age, of course, might be one, but we are already controlling for that. So, at least to my mind, and without further information about the variables the exogeneity condition is satisfied.

(e)

```
tslsm <- ivreg(pira ~ p401k + inc + incsq + age + agesq |
  e401k + inc + incsq + age + agesq, data = data)
summary(tslsm, vcov = sandwich, df = Inf, diagnostics = TRUE)
```

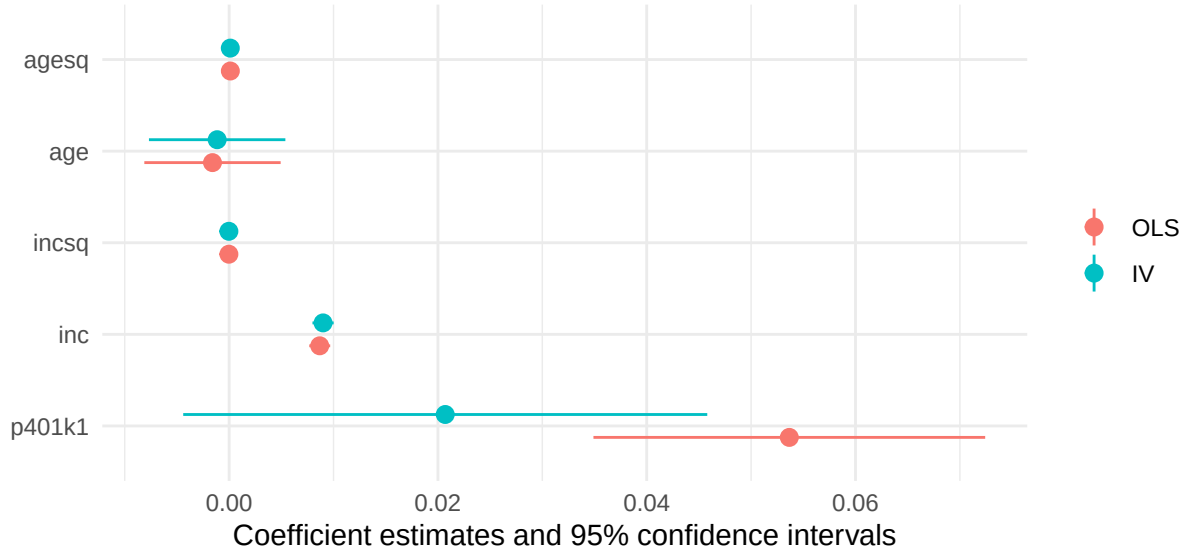
```
##
## Call:
## ivreg(formula = pira ~ p401k + inc + incsq + age + agesq | e401k +
##     inc + incsq + age + agesq, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8774 -0.2628 -0.1200  0.2051  1.0605
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.073e-01  6.536e-02  -3.172  0.00151 **
## p401k1       2.070e-02  1.323e-02   1.565  0.11759
## inc          8.998e-03  4.911e-04  18.321  < 2e-16 ***
## incsq       -2.414e-05  3.881e-06  -6.219    5e-10 ***
## age         -1.147e-03  3.248e-03  -0.353  0.72406
## agesq        1.121e-04  3.832e-05   2.924  0.00345 **
##
```

	OLS	IV
(Intercept)	−0.198	−0.207
	(0.069)	(0.069)
p401k1	0.054	0.021
	(0.010)	(0.013)
inc	0.009	0.009
	(0.001)	(0.001)
incsq	0.000	0.000
	(0.000)	(0.000)
age	−0.002	−0.001
	(0.003)	(0.003)
agesq	0.000	0.000
	(0.000)	(0.000)
Num.Obs.	9275	9275
R2	0.180	0.179
R2 Adj.	0.180	0.178
AIC	9074.7	
BIC	9124.6	
Log.Lik.	−4530.336	
F	406.851	

```
## Diagnostic tests:
##                   df1  df2 statistic  p-value
## Weak instruments    1 9269   7437.04  < 2e-16 ***
## Wu-Hausman          1 9268     15.36 8.96e-05 ***
## Sargan              0   NA        NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3947 on Inf degrees of freedom
## Multiple R-Squared: 0.1789,  Adjusted R-squared: 0.1785
## Wald test:  2064 on 5 DF,  p-value: < 2.2e-16
```

```
msummary(list(OLS = lm, IV = tslsm))
```

```
modelplot(list(OLS = lm, IV = tslsm), coef_omit = "Intercept")
```



I decided to run the 2SLS estimation in one step here. That way, I can guarantee that the standard errors and t-statistics are correct.
Unlike in the OLS regression, in this model (using robust standard errors), the coefficient of *p401k* is not statistically significant at the 10% significance level. The coefficient magnitude decreased from 5.336% in the OLS regression to 2.070% in the 2SLS estimation. So, if the exogeneity condition were satisfied with certainty, the 2SLS estimation shows that there is no difference in IRA enrollment probability between individuals participating in a 401k plan or not, on average, ceteris paribus. The models can be compared in the table and plot above (the standard errors of the IV regression are not robust here).
Looking at the diagnostic tests, we see that can reject the null hypothesis of a weak instrument in the weak instruments test. We also reject the null hypothesis in the Wu-Hausman test, indicating that OLS is not better here.

2.

```
library(margins)
```

(a)

```
path = paste0(getwd(), "/loans.dta")
data = read_dta(path)
data[c("approve", "white", "male", "married", "dep", "sch", "cosign", "chist", "pubrec",
  "mortlat1", "mortlat2", "vr")] = lapply(data[c("approve", "white", "male", "married",
  "dep", "sch", "cosign", "chist", "pubrec", "mortlat1", "mortlat2", "vr")], as.factor)

head(data)
```

```
## # A tibble: 6 x 62
##     occ loanamt action   msa suffolk  race gender appinc typur  unit married
##   <dbl>   <dbl>  <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <fct>
## 1     1      89      1  1120       0     5      3     72     0     1 0
```

```
## 2     1     128      3  1120       0     5      1     74     0     1 1
## 3     1     128      1  1120       0     5      1     84     3     1 0
## 4     1      66      1  1120       0     5      1     36     0     1 1
## 5     1     120      1  1120       0     5      1     59     8     1 1
## 6     1     111      1  1120       0     5      1     63     9     1 0
## # ... with 51 more variables: dep <fct>, emp <dbl>, yjob <dbl>, self <dbl>,
## #   atotinc <dbl>, cototinc <dbl>, hexp <dbl>, price <dbl>, other <dbl>,
## #   liq <dbl>, rep <dbl>, gdlin <dbl>, lines <dbl>, mortg <dbl>, cons <dbl>,
## #   pubrec <fct>, hrat <dbl>, obrat <dbl>, fixadj <dbl>, term <dbl>, apr <dbl>,
## #   prop <dbl>, inss <dbl>, inson <dbl>, gift <dbl>, cosign <fct>, unver <dbl>,
## #   review <dbl>, netw <dbl>, unem <dbl>, min30 <dbl>, bd <dbl>, mi <dbl>,
## #   old <dbl>, vr <fct>, sch <fct>, black <dbl>, hispan <dbl>, male <fct>, ...
```

```
pm = glm(approve ~ white, data, family = binomial("probit"))
summary(pm)
```

```
##
## Call:
## glm(formula = approve ~ white, family = binomial("probit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1864   0.4384   0.4384   0.4384   0.8314
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.54695    0.07544   7.251 4.15e-13 ***
## white1       0.78395    0.08671   9.041  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1480.7  on 1988  degrees of freedom
## Residual deviance: 1401.8  on 1987  degrees of freedom
## AIC: 1405.8
##
## Number of Fisher Scoring iterations: 4
```

```
predict(pm, newdata = data.frame("white" = as.factor(c(1, 0))), type = "response")
```

```
##         1         2
## 0.9083879 0.7077922
```

First, I encoded all focal variables, which I thought were categorical (I was not sure about *dep*), as factors. According to this crude estimated probit model of *approve* on *white*, the probability of getting the loan approved increases when the subject is white compared to nonwhite, on average, since the coefficient is positive and statistically significant at the 1% significance level. The magnitude of this association cannot be estimated from this, though, since partial effects in probit models are non-constant. This model predicts a probability of loan approval of 90.8388% for whites and 70.7792% for nonwhites. This model is lacking some control variables, such as gender or age, though.

(b)

```
dataLm = read_dta(path)
dataLm[c("white", "male", "married", "dep", "sch", "cosign", "chist", "pubrec", "mortlat1",
  "mortlat2", "vr")] = lapply(dataLm[c("white", "male", "married", "dep", "sch", "cosign",
  "chist", "pubrec", "mortlat1", "mortlat2", "vr")], as.factor)

lm = lm(approve ~ white, dataLm)
summary(lm)
```

```
##
## Call:
## lm(formula = approve ~ white, data = dataLm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90839  0.09161  0.09161  0.09161  0.29221
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.70779    0.01824   38.81   <2e-16 ***
## white1       0.20060    0.01984   10.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3201 on 1987 degrees of freedom
## Multiple R-squared:  0.04893,    Adjusted R-squared:  0.04845
## F-statistic: 102.2 on 1 and 1987 DF,  p-value: < 2.2e-16
```

```
predict(lm, newdata = data.frame("white" = as.factor(c(1, 0))))
```

```
##         1         2
## 0.9083879 0.7077922
```

For the linear probability model, we again need to make sure that the dependent variable in not encoded as a factor. This linear probability model predicts a 20.06% increase in probability of loan approval for whites (70.779% + 20.06% = 90.839%) compared to nonwhites (70.779%), on average. Both the intercept and the coefficient are statistically significant at the 1% significance level. The linear probability model makes the exact same predictions as the probit model. So in the case of this simple model, we prefer the linear probability model based on its better interpretability and OLS benefits. One can also notice a difference in standard errors between the two models.

(c)

```
pm = glm(approve ~ white + hrat + obrat + loanprc + unem + male + married + dep + sch +
  cosign + chist + pubrec + mortlat1 + mortlat2 + vr, data, family = binomial("probit"))
summary(pm)
```

```
##
```

```
## Call:
## glm(formula = approve ~ white + hrat + obrat + loanprc + unem +
##     male + married + dep + sch + cosign + chist + pubrec + mortlat1 +
##     mortlat2 + vr, family = binomial("probit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0950   0.2362   0.3411   0.4869   1.9979
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.050794   0.316184   6.486 8.81e-11 ***
## white1        0.532479   0.097452   5.464 4.66e-08 ***
## hrat          0.008364   0.007061   1.185  0.23617
## obrat        -0.028319   0.006192  -4.574 4.79e-06 ***
## loanprc      -1.004248   0.239688  -4.190 2.79e-05 ***
## unem         -0.035933   0.017828  -2.016  0.04385 *
## male1        -0.038689   0.110200  -0.351  0.72553
## married1      0.266431   0.096740   2.754  0.00589 **
## dep1         -0.015111   0.119007  -0.127  0.89896
## dep2         -0.124977   0.121438  -1.029  0.30341
## dep3         -0.112561   0.170879  -0.659  0.51008
## dep4         -0.540133   0.280876  -1.923  0.05448 .
## dep5          4.217971  95.035413   0.044  0.96460
## dep6          3.996176 130.613451   0.031  0.97559
## dep7          2.987749 235.033707   0.013  0.98986
## dep8          3.084013 235.033718   0.013  0.98953
## sch1          0.011054   0.095678   0.116  0.90802
## cosign1       0.056668   0.242111   0.234  0.81494
## chist1        0.584222   0.095920   6.091 1.12e-09 ***
## pubrec1      -0.773119   0.127788  -6.050 1.45e-09 ***
## mortlat11    -0.165458   0.258328  -0.640  0.52185
## mortlat21    -0.492990   0.338467  -1.457  0.14524
## vr1          -0.190553   0.081869  -2.328  0.01994 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1476.0  on 1970  degrees of freedom
## Residual deviance: 1194.9  on 1948  degrees of freedom
##   (18 observations deleted due to missingness)
## AIC: 1240.9
##
## Number of Fisher Scoring iterations: 13
```

The difference between whites and nonwhites persists in the extended model. The association between *white* and *approve* is still positive and statistically significant at the 1% significance level. This is evidence that there is still discrimination against nonwhites even after controlling for various variables.

(d)

```
logm = glm(approve ~ white + hrat + obrat + loanprc + unem + male + married + dep + sch +
  cosign + chist + pubrec + mortlat1 + mortlat2 + vr, data, family = binomial("logit"))
summary(logm)
```

```
##
## Call:
## glm(formula = approve ~ white + hrat + obrat + loanprc + unem +
##     male + married + dep + sch + cosign + chist + pubrec + mortlat1 +
##     mortlat2 + vr, family = binomial("logit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9514   0.2537   0.3439   0.4744   2.0904
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.773e+00  5.936e-01   6.356 2.08e-10 ***
## white1       9.582e-01  1.740e-01   5.506 3.66e-08 ***
## hrat         1.393e-02  1.294e-02   1.077  0.28167
## obrat       -5.383e-02  1.136e-02  -4.738 2.15e-06 ***
## loanprc     -1.894e+00  4.587e-01  -4.129 3.64e-05 ***
## unem        -6.455e-02  3.317e-02  -1.946  0.05163 .
## male1       -6.858e-02  2.069e-01  -0.331  0.74029
## married1     5.063e-01  1.818e-01   2.785  0.00536 **
## dep1        -3.027e-03  2.252e-01  -0.013  0.98928
## dep2        -2.449e-01  2.293e-01  -1.068  0.28551
## dep3        -2.280e-01  3.205e-01  -0.711  0.47682
## dep4        -9.471e-01  5.043e-01  -1.878  0.06037 .
## dep5         1.367e+01  5.992e+02   0.023  0.98180
## dep6         1.366e+01  8.150e+02   0.017  0.98662
## dep7         1.192e+01  1.455e+03   0.008  0.99346
## dep8         1.210e+01  1.455e+03   0.008  0.99337
## sch1         3.834e-02  1.787e-01   0.215  0.83010
## cosign1      7.348e-02  4.481e-01   0.164  0.86974
## chist1       1.065e+00  1.718e-01   6.198 5.70e-10 ***
## pubrec1     -1.333e+00  2.188e-01  -6.092 1.11e-09 ***
## mortlat11   -2.642e-01  4.677e-01  -0.565  0.57216
## mortlat21   -8.989e-01  5.947e-01  -1.512  0.13063
## vr1         -3.357e-01  1.543e-01  -2.175  0.02962 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1476.0  on 1970  degrees of freedom
## Residual deviance: 1195.4  on 1948  degrees of freedom
##   (18 observations deleted due to missingness)
## AIC: 1241.4
##
## Number of Fisher Scoring iterations: 14
```

```
lm = lm(approve ~ white + hrat + obrat + loanprc + unem + male + married + dep + sch +
  cosign + chist + pubrec + mortlat1 + mortlat2 + vr, dataLm)

# Probit marginal effects
margins(pm)
```

```
##      hrat    obrat loanprc      unem white1     male1 married1      dep1
##  0.001382 -0.00468  -0.166 -0.005938 0.1066 -0.006317  0.04586 -0.002436
##     dep2     dep3    dep4   dep5   dep6   dep7   dep8     sch1  cosign1 chist1
##  -0.0213 -0.01906 -0.1116 0.1174 0.1174 0.1171 0.1171 0.001832 0.009115 0.1188
##  pubrec1 mortlat11 mortlat21      vr1
##  -0.1778  -0.02959   -0.1022 -0.03202
```

```
# Logit marginal effects
margins(logm)
```

```
##      hrat     obrat loanprc      unem white1     male1 married1       dep1
##  0.001225 -0.004734 -0.1666 -0.005677  0.103 -0.005957  0.04655 -0.0002573
##      dep2     dep3    dep4   dep5   dep6   dep7   dep8     sch1  cosign1 chist1
##  -0.02236 -0.02071 -0.1054 0.1179 0.1179 0.1179 0.1179 0.003391 0.006333  0.117
##  pubrec1 mortlat11 mortlat21      vr1
##  -0.1665  -0.02502   -0.1014 -0.02996
```

```
# LPM marginal effects
margins(lm)
```

```
##      hrat     obrat loanprc     unem white1     male1 married1   dep1    dep2
##  0.001932 -0.005571 -0.1473 -0.00728 0.1307 -0.003778  0.04477 0.0031 -0.0149
##      dep3    dep4   dep5   dep6     dep7     dep8      sch1  cosign1 chist1
##  -0.01735 -0.1158 0.1644 0.1664 -0.01158 0.007614 0.0005651 0.004917  0.132
##  pubrec1 mortlat11 mortlat21     vr1
##   -0.241  -0.05192   -0.1149 -0.0301
```

In the logit model, as well, the coefficient of *white* is positive and statistically significant at the 1% significance level. The magnitude of the coefficient of *white* is far smaller than in the probit model. This pattern can be observed for all other coefficient magnitudes and the standard errors, as well. The t-statistics are similar, though. Comparing the marginal effects (partial derivatives) of both models, we can see that they are quite similar. The marginal effects of the linear probability model diverge from the logit and probit estimates.

3.

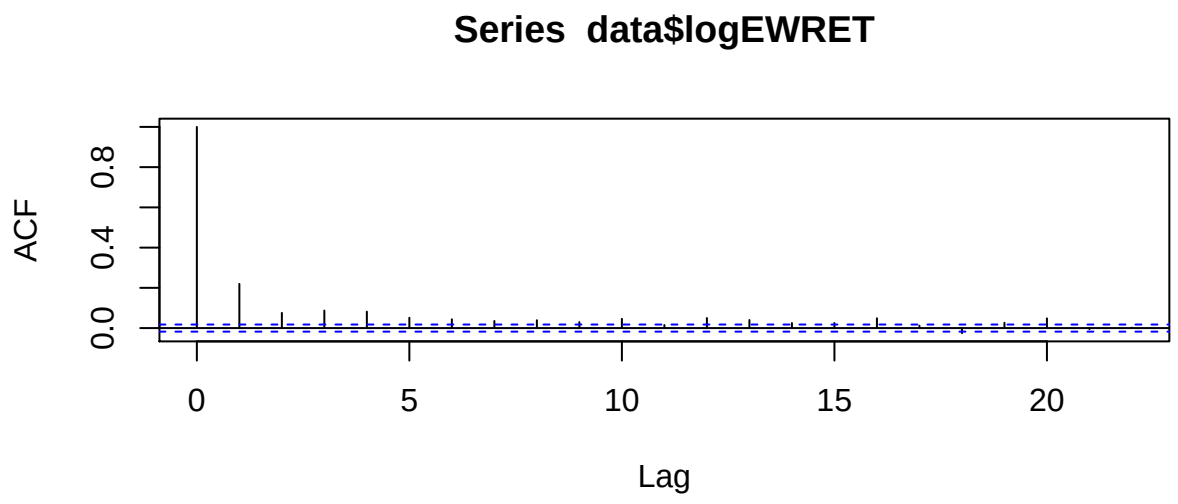
```
library(readxl)
library(lubridate)
library(stats)
library(xts)
library(psych)
library(tseries)
```

(a)

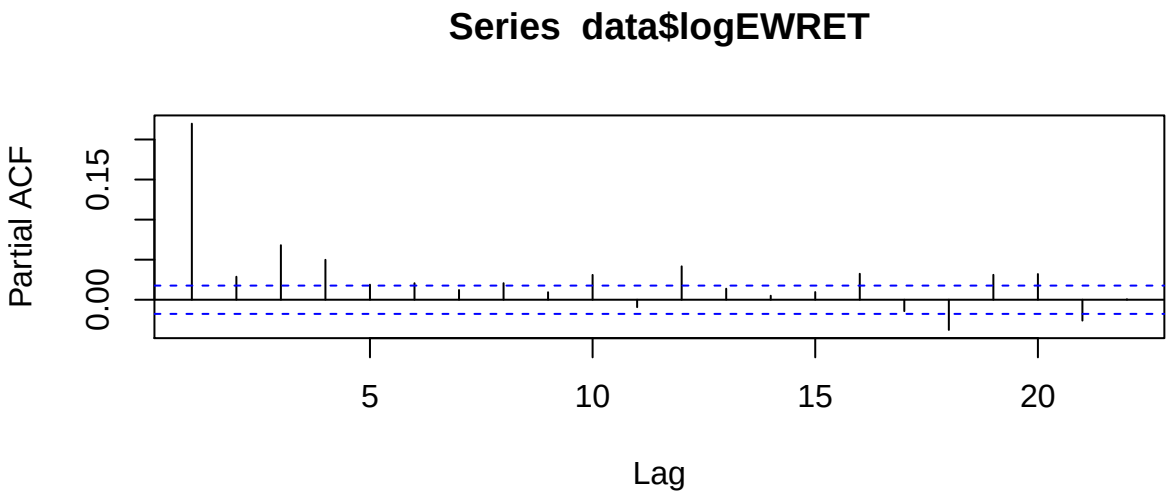
```
path = paste0(getwd(), "/data_assignment2.xlsx")
data = lapply(excel_sheets(path), read_excel, path = path)

data = data[[1]]
data = data %>% mutate(DATE = ymd(DATE))

simpleToLog = function(returns) {return(log(returns + 1))}
for (i in (2:ncol(data))) {
  data[paste0("log",names(data)[i])] = simpleToLog(data[, i])
}

head(data)
```

```
## # A tibble: 6 x 5
##   DATE           VWRET   EWRET  logVWRET logEWRET
##   <date>         <dbl>   <dbl>     <dbl>    <dbl>
## 1 1963-01-02 -0.00501  0.00913 -0.00502   0.00908
## 2 1963-01-03  0.0165   0.0213   0.0163    0.0210
## 3 1963-01-04  0.00690  0.0139   0.00687   0.0138
## 4 1963-01-07  0.000946 0.00533  0.000946  0.00532
## 5 1963-01-08  0.00903  0.00916  0.00899   0.00912
## 6 1963-01-09 -0.00147  0.00153 -0.00147   0.00152
```

```
acf(data$logEWRET, lag.max = 22)
```




```
pacf(data$logEWRET, lag.max = 22)
```



In the ACF plot, we can see that the functions decreases steeply from the first to the second lag and then gradually decreases towards statistical insignificance. Given that all of the first couple of autocorrelation coefficients are positive, this suggests that the AR coefficients should be positive, as well. The low persistence of the pattern suggests a lower number of AR coefficients, perhaps AR(1, 1). Still, all lags up until lag 10 lie outside the 95% confidence interval (and some additional ones thereafter).
The PACF exhibits a more irregular pattern. The first lag is strongly different from 0 and the second one less so. The third and fourth one are strongly different from 0, again. Judging by the shear magnitude of the first lag, it could be an AR(1, 1) model, although MA terms could also play a role here. To better assess this, we should look at the information criteria.

(b)

```
row.names = c("AR(0)","AR(1)","AR(2)", "AR(3)", "AR(4)", "AR(5)")
column.names = c("MA(0)", "MA(1)", "MA(2)", "MA(3)", "MA(4)", "MA(5)")
matrix.names = c("AIC","SBIC")
table = array(NA, c(6, 6, 2), dimnames = list(row.names, column.names, matrix.names))

min_AIC = 10000
min_AIC_ARMA = c(0, 0)
min_SBIC = 10000
min_SBIC_ARMA = c(0, 0)
for (ar in 0:5) {
  for (ma in 0:5) {
    arma = arima(data$logEWRET, order = c(ar, 0, ma))
    table[ar + 1, ma + 1, 1] = AIC(arma)
    table[ar + 1, ma + 1, 2] = AIC(arma, k = log(nrow(data)))
    if (AIC(arma) < min_AIC) {
      min_AIC = AIC(arma)
      min_AIC_ARMA = c(ar, ma)
    }
    if (AIC(arma, k = log(nrow(data))) < min_SBIC) {
      min_SBIC = AIC(arma, k = log(nrow(data)))
      min_SBIC_ARMA = c(ar, ma)
```

```
    }
  }
}

table
```

```
## , , AIC
##
##           MA(0)     MA(1)     MA(2)     MA(3)     MA(4)     MA(5)
## AR(0) -82741.81 -83305.65 -83331.95 -83369.81 -83415.83 -83426.66
## AR(1) -83350.17 -83380.33 -83469.82 -83469.24 -83468.03 -83474.09
## AR(2) -83358.37 -83461.47 -83469.37 -83466.51 -83465.72 -83470.64
## AR(3) -83413.74 -83467.50 -83467.08 -83466.31 -83465.66 -83466.18
## AR(4) -83442.46 -83470.86 -83463.57 -83465.92 -83466.28 -83473.86
## AR(5) -83444.84 -83473.57 -83476.66 -83469.69 -83472.94 -83471.29
##
## , , SBIC
##
##           MA(0)     MA(1)     MA(2)     MA(3)     MA(4)     MA(5)
## AR(0) -82726.97 -83283.39 -83302.27 -83332.71 -83371.31 -83374.72
## AR(1) -83327.91 -83350.65 -83432.72 -83424.72 -83416.09 -83414.73
## AR(2) -83328.69 -83424.37 -83424.85 -83414.57 -83406.35 -83403.86
## AR(3) -83376.64 -83422.98 -83415.14 -83406.95 -83398.87 -83391.98
## AR(4) -83397.93 -83418.91 -83404.20 -83399.14 -83392.07 -83392.24
## AR(5) -83392.89 -83414.21 -83409.88 -83395.49 -83391.31 -83382.25
```

```
cat("The ARMA model with the lowest AIC value is ARMA(", toString(min_AIC_ARMA[1]),
  ", ", toString(min_AIC_ARMA[2]), ")\n with an AIC value of ", toString(min_AIC), sep = "")
```

```
## The ARMA model with the lowest AIC value is ARMA(5, 2)
##  with an AIC value of -83476.6578190205
```

```
cat("The ARMA model with the lowest SBIC value is ARMA(", toString(min_SBIC_ARMA[1]),
  ", ", toString(min_SBIC_ARMA[2]), ")\n with an SBIC value of ", toString(min_SBIC), sep = "")
```

```
## The ARMA model with the lowest SBIC value is ARMA(1, 2)
##  with an SBIC value of -83432.7177728529
```

```
AIC_model = arima(data$logEWRET, order = c(5, 0, 2))
AIC_model
```

```
##
## Call:
## arima(x = data$logEWRET, order = c(5, 0, 2))
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ar5      ma1      ma2  intercept
##       0.2222  0.7706  -0.1026  0.0308  -0.0374  -0.0172  -0.7655      7e-04
## s.e.  0.0587  0.0552   0.0146  0.0102   0.0107   0.0581   0.0475      1e-04
##
## sigma^2 estimated as 6.731e-05:  log likelihood = 41747.33,  aic = -83476.66
```

```
SBIC_model = arima(data$logEWRET, order = c(1, 0, 2))
SBIC_model
```

```
##
## Call:
## arima(x = data$logEWRET, order = c(1, 0, 2))
##
## Coefficients:
##          ar1      ma1      ma2  intercept
##       0.9080  -0.7021  -0.1288      8e-04
## s.e.  0.0175   0.0201   0.0115      1e-04
##
## sigma^2 estimated as 6.739e-05:  log likelihood = 41739.91,  aic = -83469.82
```

```
# AIC Ljung-Box test
Box.test(AIC_model$residuals, lag = 10, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  AIC_model$residuals
## X-squared = 5.7207, df = 10, p-value = 0.8382
```

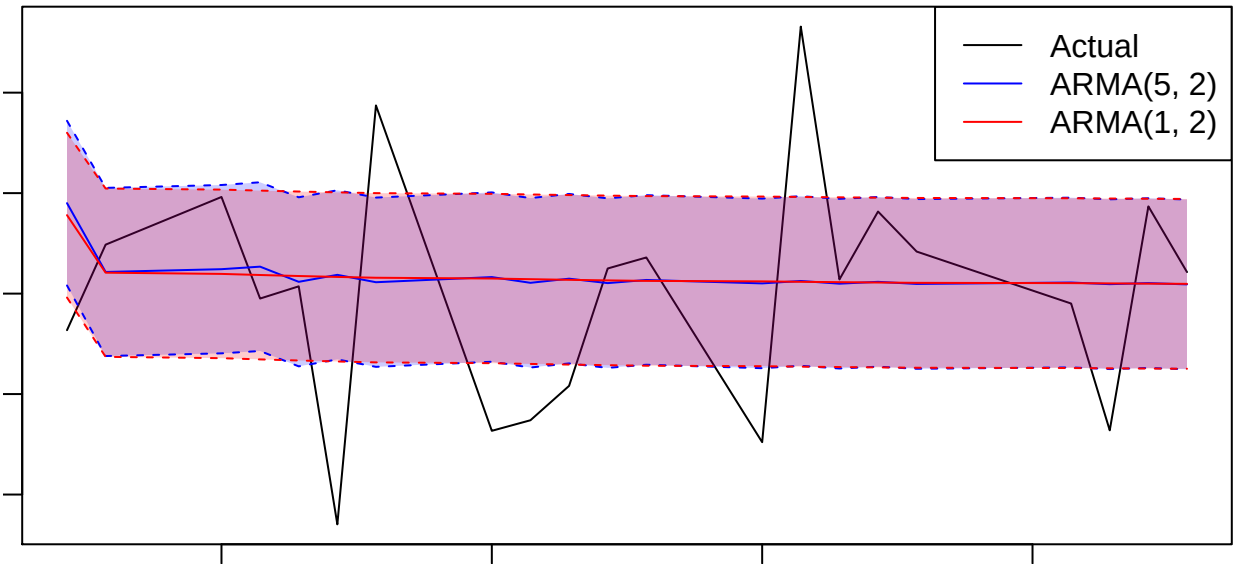
```
# SBIC Ljung-Box test
Box.test(SBIC_model$residuals, lag = 10, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  SBIC_model$residuals
## X-squared = 16.379, df = 10, p-value = 0.08928
```

The first table contains information about the AIC criterion, the second about the SBIC criterion. For each, we are looking for the ARMA model that minimizes the criterion. Accordingly, ARMA(5, 2) minimizes the AIC criterion with a value of -83476.6578. Perhaps, if we allowed more lags, the model would have been even more extensive. ARMA(1, 2) is the model that minimizes the SBIC criterion with a value of -83432.7178. A lower number of parameters allowed by the SBIC criterion is to be expected since the SBIC is the information criterion that imposes the strongest penalty (ln(T)) for each additional parameter that is included in the model.
To check for possible model inadequacy we can perform the Box-Ljung test. I decided to run it with 10 lags, since the natural logarithm of the number of observations is about 9.4203. We fail to reject the null hypothesis of no residual serial correlation in the first 10 lags at the 5% significance level. For the SBIC model, we would reject the null hypothesis at the 10% significance level, though. This serves as strong evidence for model adequacy in the case of the AIC model and only mild evidence for model adequacy in the case of the SBIC model.

(c)

```
AIC_model_truncated = arima(head(data$logEWRET, length(data$logEWRET) - 21),
  order = c(5, 0, 2))
SBIC_model_truncated = arima(head(data$logEWRET, length(data$logEWRET) - 21),
  order = c(1, 0, 2))

AIC_predict = predict(AIC_model_truncated, n.ahead = 21)
AIC_min = AIC_predict$pred - AIC_predict$se
AIC_max = AIC_predict$pred + AIC_predict$se
SBIC_predict = predict(SBIC_model_truncated, n.ahead = 21)
SBIC_min = SBIC_predict$pred - SBIC_predict$se
SBIC_max = SBIC_predict$pred + SBIC_predict$se
dates = tail(data$DATE, 21)
actuals = tail(data$logEWRET, 21)

par(mar = c(0.5, 0.5, 0.5, 0.5))
plot(dates, actuals, type = "l", xlab = "", ylab = "")
polygon(c(dates, rev(dates)) , c(AIC_min, rev(AIC_max)),
  col = rgb(red = 0, green = 0, blue = 1, alpha = 0.2), border = NA)
polygon(c(dates, rev(dates)) , c(SBIC_min, rev(SBIC_max)),
  col = rgb(red = 1, green = 0, blue = 0, alpha = 0.2), border = NA)
lines(dates, AIC_predict$pred, col = "blue")
lines(dates, AIC_max, lty = 'dashed', col = 'blue')
lines(dates, AIC_min, lty = 'dashed', col = 'blue')
lines(dates, SBIC_predict$pred, col = "red")
lines(dates, SBIC_max, lty = 'dashed', col = 'red')
lines(dates, SBIC_min, lty = 'dashed', col = 'red')
legend("topright", legend = c("Actual", "ARMA(5, 2)", "ARMA(1, 2)"),
  col = c("black", "blue", "red"), lty = 1)
```




```
AIC_MSE = mean((AIC_predict$pred - actuals)^2)
AIC_MAE = mean(abs(AIC_predict$pred - actuals))
cat("The MSE and the MAE of the ARMA(5, 2) model selected by the AIC criterion are\n ",
  toString(AIC_MSE), " and ", toString(AIC_MAE), ", respectively.", sep = "")
```

```
## The MSE and the MAE of the ARMA(5, 2) model selected by the AIC criterion are
##  0.000139237790739307 and 0.00900992220427397, respectively.
```

```
SBIC_MSE = mean((SBIC_predict$pred - actuals)^2)
SBIC_MAE = mean(abs(SBIC_predict$pred - actuals))
cat("The MSE and the MAE of the ARMA(1, 2) model selected by the SBIC crterion are\n ",
  toString(SBIC_MSE), " and ", toString(SBIC_MAE), ", respectively.", sep = "")
```

```
## The MSE and the MAE of the ARMA(1, 2) model selected by the SBIC crterion are
##  0.000137591827369786 and 0.00893760298634781, respectively.
```

As we can see in the plot, both ARMA models quite quickly and in a similar fashion revert to their means, after about one day. The ARMA(5, 2) model selected by the AIC criterion shows perhaps a little more persistence. This pattern is to be expected, since forecasts become more inaccurate, the farther we predict into the future. Nevertheless, the actual observations rarely lie outside the confidence bands and, if so, barely. This returns some usefulness to the forecasts. The confidence bands of the 2 forecasts are almost identical.
In terms of point forecast accuracy, the ARMA(1, 2) model does a slightly better job at forecasting given that both its accuracy measures are (just barely) lower. From this perspective and considering its parsimony, we would prefer this model over the ARMA(5, 2) model.

(d)

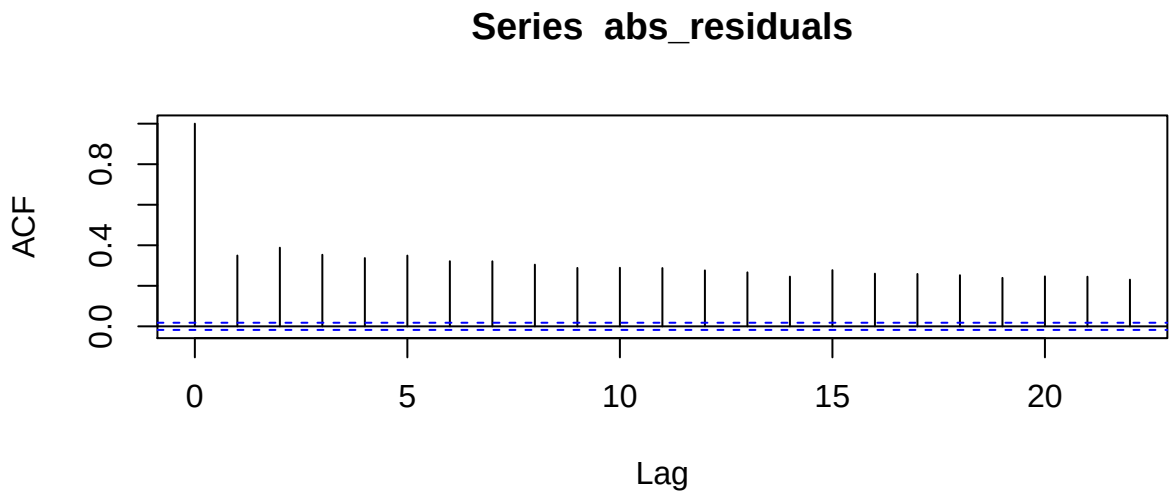
(i)

```
abs_residuals = abs(SBIC_model$residuals)
squared_residuals = SBIC_model$residuals^2

describe(data.frame(abs_residuals, squared_residuals))
```

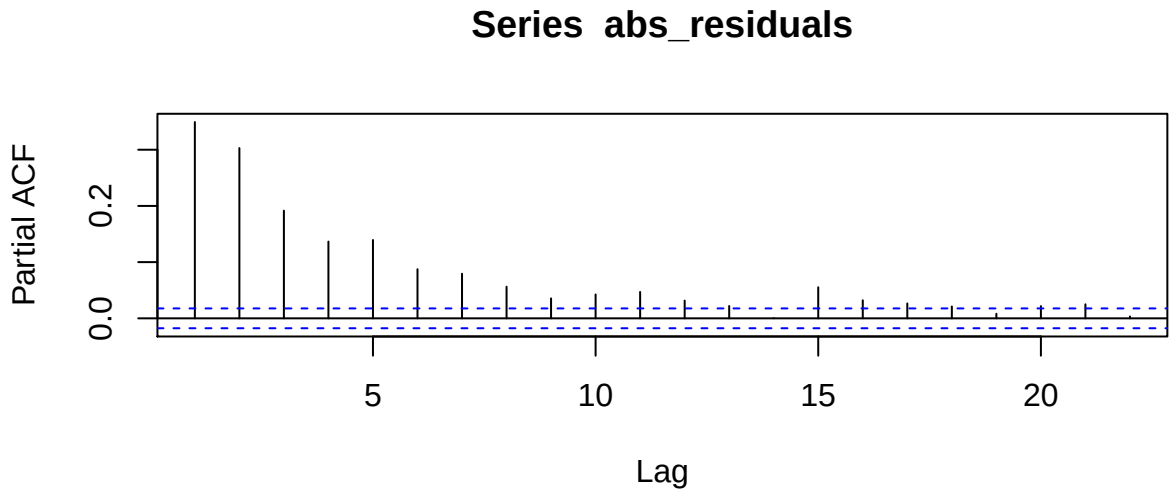
```
##                   vars     n mean   sd median trimmed mad min  max range  skew
## abs_residuals        1 12336 0.01 0.01      0       0   0   0 0.11  0.11  4.24
## squared_residuals    2 12336 0.00 0.00      0       0   0   0 0.01  0.01 18.10
##                   kurtosis se
## abs_residuals        32.87  0
## squared_residuals   514.49  0
```

(ii)

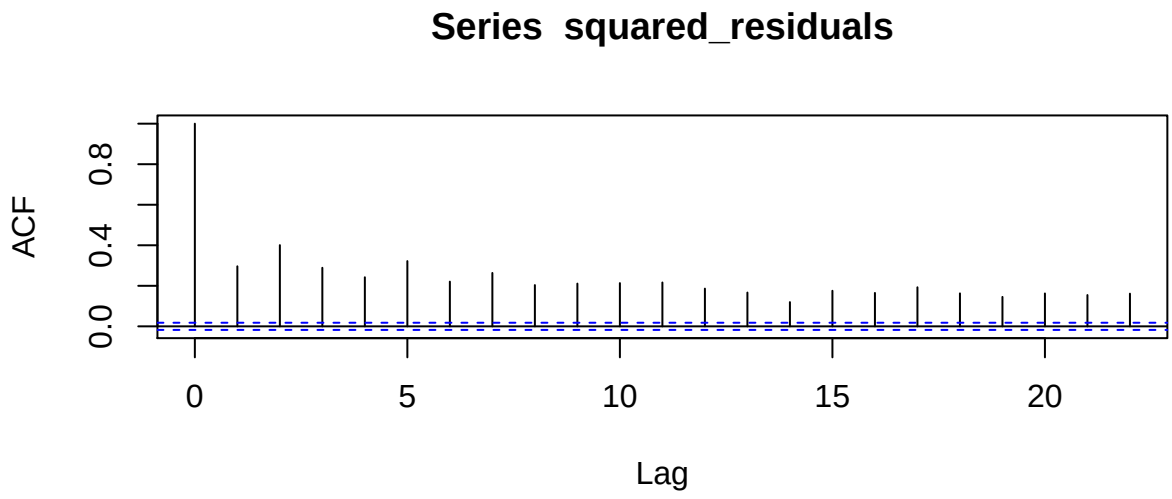
```
acf(abs_residuals, lag.max = 22)
```



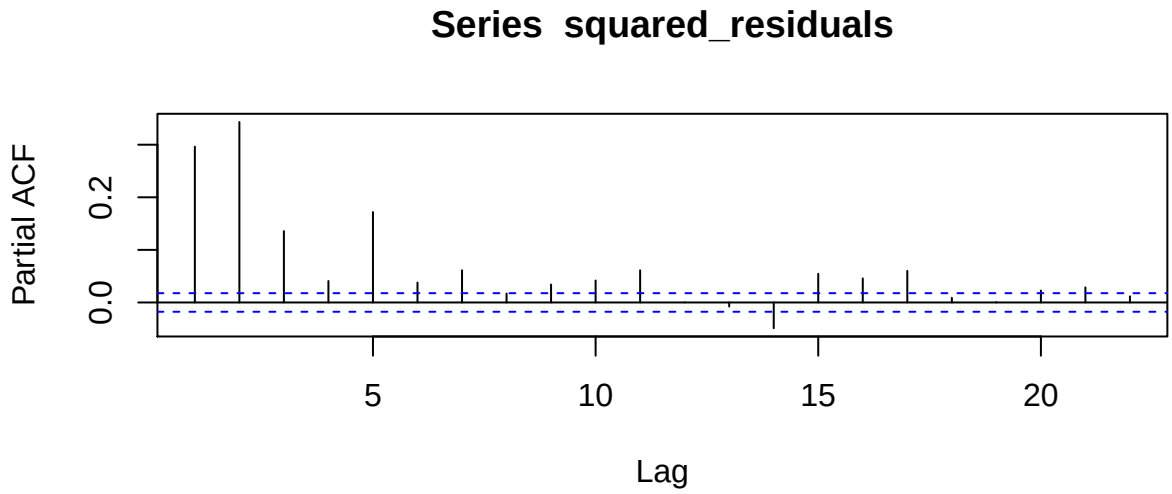

```
pacf(abs_residuals, lag.max = 22)
```




```
acf(squared_residuals, lag.max = 22)
```




```
pacf(squared_residuals, lag.max = 22)
```



If the model is adequate, then the residual series (not squared and not absolute) should behave like white noise. By performing the Ljung-Box test earlier we have already seen that this is likely the case. However, according to the plots here, there is still autocorrelation left in the second moment of the series. Both the squared and absolute residuals show persistent autocorrelation patterns. This information could be used in forecasting volatility using a GARCH model.

(iii)

```
adf.test(abs_residuals)
```

```
## Warning in adf.test(abs_residuals): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  abs_residuals
## Dickey-Fuller = -11.37, Lag order = 23, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(squared_residuals)
```

```
## Warning in adf.test(squared_residuals): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  squared_residuals
## Dickey-Fuller = -13.038, Lag order = 23, p-value = 0.01
## alternative hypothesis: stationary
```

The null hypothesis of the Augmented Dickey-Fuller test is that the time series is non-stationary; that there is a unit root. For both the squared and absolute residuals we can reject this null hypothesis at the 1% significance level. We have evidence that both series are stationary.

(iv)

```
row.names = c("AR(0)","AR(1)","AR(2)", "AR(3)", "AR(4)", "AR(5)", "AR(6)", "AR(7)", "AR(8)")
column.names = c("MA(0)", "MA(1)", "MA(2)", "MA(3)", "MA(4)", "MA(5)", "MA(6)", "MA(7)", "MA(8)")
table = array(NA, c(9, 9), dimnames = list(row.names, column.names))

min_SBIC = 10000
min_SBIC_ARMA = c(0, 0)
for (ar in 0:8) {
  for (ma in 0:8) {
    arma = arima(squared_residuals, order = c(ar, 0, ma))
    table[ar + 1, ma + 1] = AIC(arma, k = log(nrow(data)))
    if (AIC(arma, k = log(nrow(data))) < min_SBIC) {
      min_SBIC = AIC(arma, k = log(nrow(data)))
      min_SBIC_ARMA = c(ar, ma)
    }
  }
}
```

```
## Warning in arima(squared_residuals, order = c(ar, 0, ma)): possible convergence
## problem: optim gave code = 1
```

```
table
```

```
##           MA(0)     MA(1)     MA(2)     MA(3)     MA(4)     MA(5)     MA(6)
## AR(0) -166000.8 -166663.2 -168010.6 -168222.4 -168286.5 -168701.1 -168733.9
## AR(1) -167124.9 -169011.2 -169074.3 -169285.1 -169276.0 -169277.6 -169371.7
## AR(2) -168660.6 -169104.7 -169142.6 -169275.8 -169351.8 -169374.7 -169388.1
```

```
## AR(3) -168880.4 -169256.7 -169263.7 -169296.3 -169382.9 -169373.8 -169430.7
## AR(4) -168891.5 -169268.1 -169378.2 -169375.2 -169368.5 -169364.1 -169421.0
## AR(5) -169252.2 -169299.6 -169374.3 -169365.7 -169373.6 -169446.6 -169381.3
## AR(6) -169260.4 -169426.7 -169417.2 -169417.9 -169365.1 -169373.2 -169482.8
## AR(7) -169297.3 -169417.3 -169429.5 -169425.3 -169398.7 -169573.1 -169373.5
## AR(8) -169291.5 -169416.6 -169424.9 -169397.8 -169417.8 -169561.2 -169553.0
##           MA(7)     MA(8)
## AR(0) -168917.6 -168935.8
## AR(1) -169364.0 -169413.0
## AR(2) -169393.3 -169404.9
## AR(3) -169421.3 -169412.9
## AR(4) -169411.2 -169402.6
## AR(5) -169373.3 -169397.0
## AR(6) -169533.1 -169539.4
## AR(7) -169419.4 -169561.0
## AR(8) -169565.2 -169551.7
```

```
cat("The ARMA model with the lowest SBIC value is ARMA(", toString(min_SBIC_ARMA[1]), ", ",
  toString(min_SBIC_ARMA[2]), ")\n with an SBIC value of ", toString(min_SBIC), sep = "")
```

```
## The ARMA model with the lowest SBIC value is ARMA(7, 5)
##  with an SBIC value of -169573.120217373
```

```
residual_model = arima(squared_residuals, order = c(7, 0, 5))
residual_model
```

```
##
## Call:
## arima(x = squared_residuals, order = c(7, 0, 5))
##
## Coefficients:
##          ar1     ar2      ar3     ar4     ar5      ar6      ar7      ma1
##       0.5624  0.4956  -0.8018  0.3958  0.6185  -0.1383  -0.1609  -0.4532
## s.e.  0.1672  0.1828   0.0289  0.1247  0.1306   0.0382   0.0458   0.1672
##           ma2     ma3      ma4      ma5  intercept
##       -0.2752  0.7395  -0.4458  -0.3762      1e-04
## s.e.   0.1642  0.0492   0.0997   0.1226      0e+00
##
## sigma^2 estimated as 6.208e-08:  log likelihood = 84852.5,  aic = -169677
```

```
Box.test(residual_model$residuals, lag = 10, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residual_model$residuals
## X-squared = 39.554, df = 10, p-value = 2.03e-05
```

The SBIC criterion recommends an ARMA(7, 5) model for the squared residuals. I tested several ARMA specifications. At ARMA(8, 8) the AR components did not reach its capacity for the first time, and hence, I decided to stop there. Regarding the Box-Ljung test, we reject the null hypothesis of no residual serial correlation in the first 10 lags at the 1% significance level. This suggests that there could still be persistence in the squared residuals that is not captured by the current model. Perhaps an even higher order ARMA model could resolve this.

4.

```
library(vars)
```

(a)

```
path = paste0(getwd(), "/data_assignment2.xlsx")
data = lapply(excel_sheets(path), read_excel, path = path)

data = data[[2]]

head(data)
```

```
## # A tibble: 6 x 3
##   DATE                `1YRRATE` `5YRRATE`
##   <dttm>                  <dbl>     <dbl>
## 1 1980-01-01 00:00:00     12.4      11.1
## 2 1980-02-01 00:00:00     15.4      13.5
## 3 1980-03-01 00:00:00     15.8      13.3
## 4 1980-04-01 00:00:00     11.1      10.8
## 5 1980-05-01 00:00:00      8.83      9.87
## 6 1980-06-01 00:00:00      8.49      9.48
```

```
adf.test(data$'1YRRATE')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$"1YRRATE"
## Dickey-Fuller = -2.5719, Lag order = 7, p-value = 0.336
## alternative hypothesis: stationary
```

```
adf.test(data$'5YRRATE')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$"5YRRATE"
## Dickey-Fuller = -3.2105, Lag order = 7, p-value = 0.08623
## alternative hypothesis: stationary
```

```
toLog = function(vals) {return(c(NA, diff(log(vals))))}
for (i in (2:ncol(data))) {
  data[paste0("log",names(data)[i])] = toLog(data[[i]])
}

head(data)
```

```
## # A tibble: 6 x 5
##   DATE                `1YRRATE` `5YRRATE` log1YRRATE log5YRRATE
##   <dttm>                  <dbl>     <dbl>      <dbl>      <dbl>
## 1 1980-01-01 00:00:00     12.4      11.1     NA         NA
## 2 1980-02-01 00:00:00     15.4      13.5      0.220      0.193
## 3 1980-03-01 00:00:00     15.8      13.3      0.0224    -0.0127
## 4 1980-04-01 00:00:00     11.1      10.8     -0.351     -0.207
## 5 1980-05-01 00:00:00      8.83      9.87    -0.231     -0.0919
## 6 1980-06-01 00:00:00      8.49      9.48    -0.0393    -0.0403
```

```
adf.test(data$log1YRRATE[is.na(data$log1YRRATE) == FALSE])
```

```
## Warning in adf.test(data$log1YRRATE[is.na(data$log1YRRATE) == FALSE]): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$log1YRRATE[is.na(data$log1YRRATE) == FALSE]
## Dickey-Fuller = -6.6979, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(data$log5YRRATE[is.na(data$log1YRRATE) == FALSE])
```

```
## Warning in adf.test(data$log5YRRATE[is.na(data$log1YRRATE) == FALSE]): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$log5YRRATE[is.na(data$log1YRRATE) == FALSE]
## Dickey-Fuller = -8.4897, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
data = data.frame(log1YRRATE = data$log1YRRATE[is.na(data$log1YRRATE) == FALSE],
  log5YRRATE = data$log5YRRATE[is.na(data$log1YRRATE) == FALSE])

VARselect(data, lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      3      1      3
##
## $criteria
##                  1            2            3            4            5
## AIC(n) -9.430261e+00 -9.427456e+00 -9.4661948105 -9.463285e+00 -9.450111e+00
## HQ(n)  -9.407994e+00 -9.390344e+00 -9.4142376656 -9.396483e+00 -9.368464e+00
## SC(n)  -9.373854e+00 -9.333444e+00 -9.3345774244 -9.294062e+00 -9.243284e+00
## FPE(n)  8.025827e-05  8.048386e-05  0.0000774259  7.765203e-05  7.868255e-05
##                  6            7            8            9           10
## AIC(n) -9.451230e+00 -9.444049e+00 -9.443842e+00 -9.464886e+00 -9.452204e+00
```

```
## HQ(n)  -9.354738e+00 -9.332712e+00 -9.317660e+00 -9.323860e+00 -9.296332e+00
## SC(n)  -9.206797e+00 -9.162012e+00 -9.124200e+00 -9.107639e+00 -9.057351e+00
## FPE(n)  7.859569e-05  7.916362e-05  7.918203e-05  7.753558e-05  7.852832e-05
```

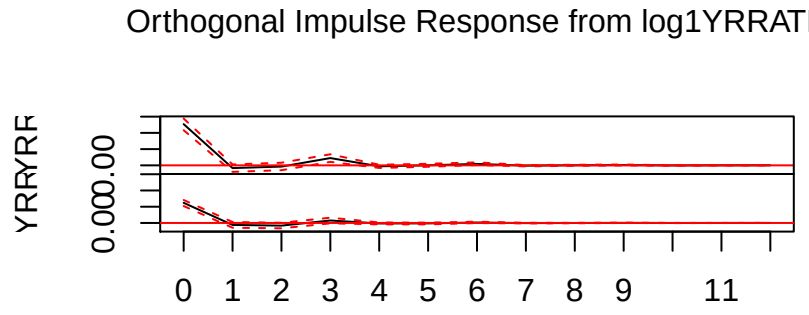
```
var = VAR(data, p = 3)
var
```

```
##
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation log1YRRATE:
## ================================================
## Call:
## log1YRRATE = log1YRRATE.l1 + log5YRRATE.l1 + log1YRRATE.l2 + log5YRRATE.l2 + log1YRRATE.l3 + log5YRRA
##
## log1YRRATE.l1 log5YRRATE.l1 log1YRRATE.l2 log5YRRATE.l2 log1YRRATE.l3
##  -0.191252605   0.257265046   0.009723194  -0.080518277   0.266424793
## log5YRRATE.l3         const
##  -0.168448630  -0.005364897
##
##
## Estimated coefficients for equation log5YRRATE:
## ================================================
## Call:
## log5YRRATE = log1YRRATE.l1 + log5YRRATE.l1 + log1YRRATE.l2 + log5YRRATE.l2 + log1YRRATE.l3 + log5YRRA
##
## log1YRRATE.l1 log5YRRATE.l1 log1YRRATE.l2 log5YRRATE.l2 log1YRRATE.l3
##  -0.153393594   0.216051287  -0.025714623  -0.079074852   0.065668758
## log5YRRATE.l3         const
##  -0.004451779  -0.004155355
```

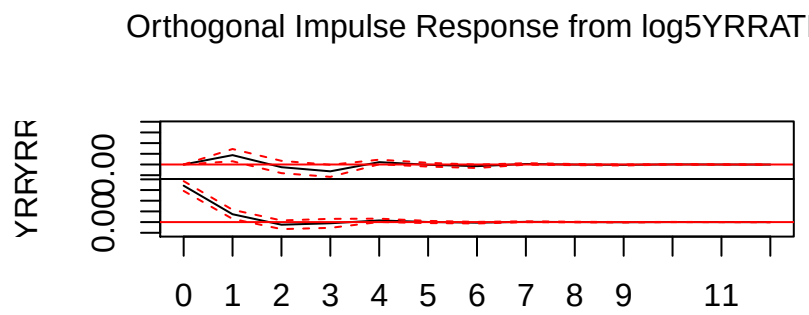
It is questionable whether the rates series are stationary. To check, we perform the Augmented Dickey-Fuller test for both. For both, we can reject the null hypothesis of non-stationarity at the 10% significance level, but we fail to do so for the 5-year rates at the 5% significance level. When we compute the log returns for both series and check again, we can reject the null hypothesis of non-stationarity in both cases at the 1% significance level. This step is similar to first-differencing. To find the appropriate number of lags for the VAR model, we apply the VARselect function, which returns the optimal number of lags according to 4 different criteria. 3 out of 4 indicate an optimal lag order of 3, so we go on to estimate the VAR(3) model. The model can be inspected in the output above.

(b)

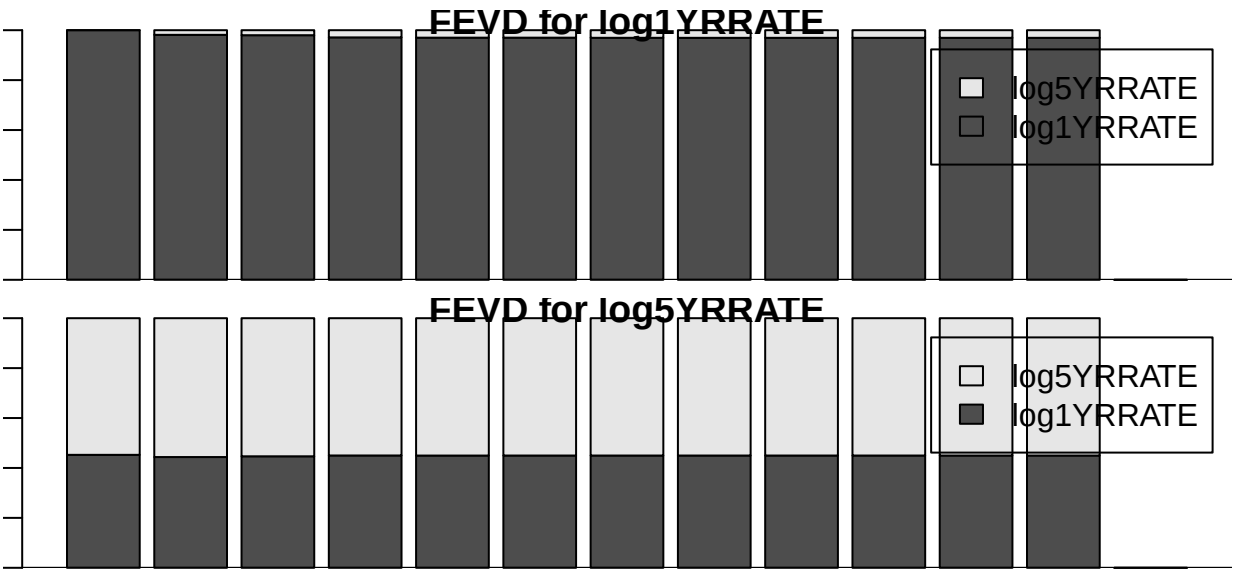
```
plot(irf(var, impulse = "log1YRRATE", response = c("log1YRRATE", "log5YRRATE"),
  ortho = TRUE, n.ahead = 12, boot = TRUE, ci = 0.95, runs = 1000))
```




```
plot(irf(var, impulse = "log5YRRATE", response = c("log1YRRATE", "log5YRRATE"),
  ortho = TRUE, n.ahead = 12, boot = TRUE, ci = 0.95, runs = 1000))
```




```
par(mar = c(0.5, 0.5, 0.5, 0.5))
plot(fevd(var, n.ahead = 12))
```




```
causality(var, cause = "log1YRRATE")$Granger
```

```
##
##  Granger causality H0: log1YRRATE do not Granger-cause log5YRRATE
##
## data:  VAR object var
## F-Test = 4.3761, df1 = 3, df2 = 866, p-value = 0.004556
```

```
causality(var, cause = "log5YRRATE")$Granger
```

```
##
##  Granger causality H0: log5YRRATE do not Granger-cause log1YRRATE
##
## data:  VAR object var
## F-Test = 4.2595, df1 = 3, df2 = 866, p-value = 0.005351
```

The response of the 5-year rate to a shock in the 1-year rate is statistically different from 0 at the zeroth lag, but becomes indistinguishable from 0 thereafter. Similarly, the response of the 1-year rate to a shock in itself is statistically different from 0 at the zeroth and perhaps slightly at the third lag. The response of the 1-year rate to a shock in the 5-year rate is only visibly statistically different from 0 at the first lag, while the 5-year rate response to a shock in itself is statistically different from 0 at the zeroth and perhaps at the first lag.
So interestingly, the 5-year rate shows some sort of response to a shock in the 1-year rate but not the other way around. A similar picture emerges when we plot the variance decomposition. The variance of the 1-year rates is almost exclusively explained by itself, but for the 5-year rate only half of the variance is explained by itself. We should remember that the ordering of the variables has an effect on the impulse responses. In fact, after swapping the variables in the data frame above, we observe the same pattern, just the other way around.
Looking at Granger causality, we can reject both the hypothesis that *log1YRRATE* does not Granger-cause *log5YRRATE* and that *log5YRRATE* does not Granger-cause *log1YRRATE* at the 1% significance level. So, overall, the variables seem to be intertwined with no striking leaning towards one variable or the other.