Econometrics III
Assignment Part 1 & 2
Tinbergen Insitute

Stanislav Avdeev
student no
stnavdeev@gmail.com

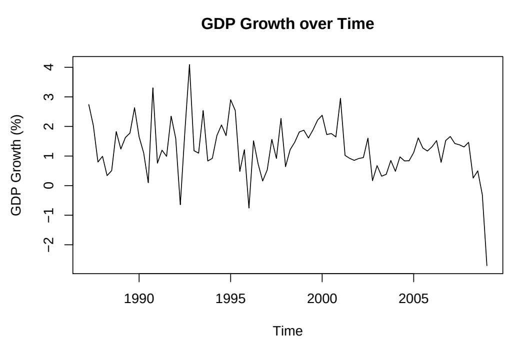
Bas Machielsen
590049bm
590049bm@eur.nl

March 11, 2021

Question 1

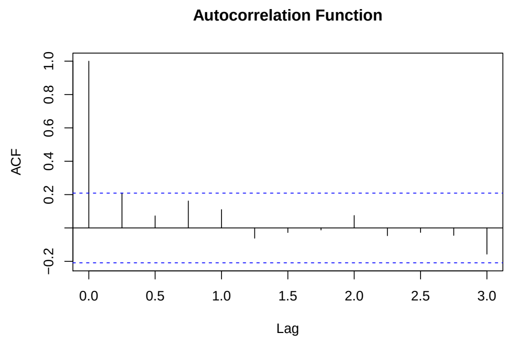
Part (a)

```
# Part 1: Plot the Dutch GDP, ACF, and PACF
df <- readr::read_csv("./data/data_assign_p1.csv")

df <- ts(df$GDP_QGR,
         frequency = 4,
         start = c(1987, 2))

plot(df,
     xlab = "Time",
     ylab = "GDP Growth (%)",
     main = "GDP Growth over Time")
```



In the above figure, we observe the growth rates of Dutch GDP from 1987 to 2009.

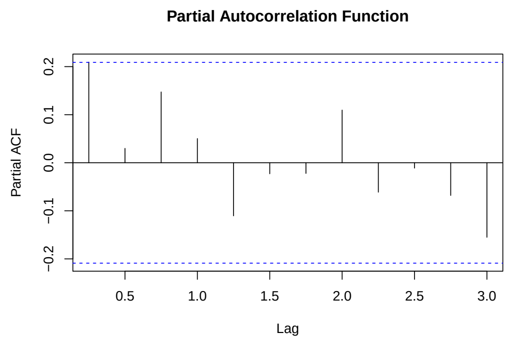
```
autoc <- acf(df,
    lag.max = 12, plot = F)

plot(autoc, main = "Autocorrelation Function")
```




```r
Box.test(df,
         lag = 12,
         type = "Ljung-Box")
```

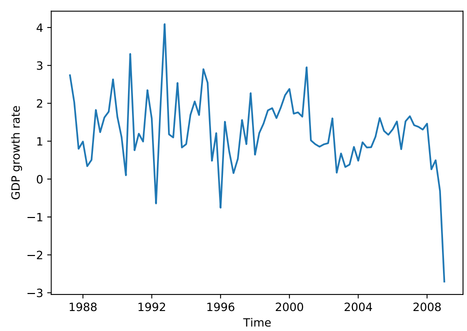
```
##
##  Box-Ljung test
##
## data:  df
## X-squared = 12.106, df = 12, p-value = 0.4372
```

```r
pautoc <- pacf(df,
     lag.max = 12, plot = F)

plot(pautoc, main = "Partial Autocorrelation Function")
```

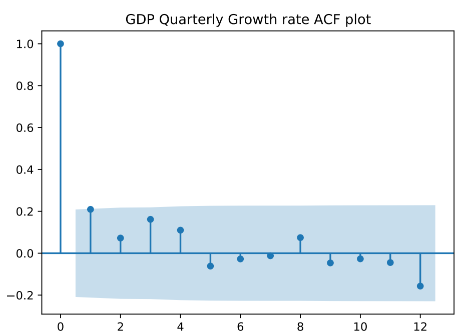



```python
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.ar_model import AutoReg

# Data
df = pd.read_csv('./data/data_assign_p1.csv')
df['obs'] = pd.to_datetime(df['obs'])
df = df.set_index('obs')
```

```
## Question 1:
# General plot:
plt.plot(df['GDP_QGR'])
plt.xlabel('Time')
plt.ylabel('GDP growth rate')
plt.show()

# ACF plot:
```




```
sm.graphics.tsa.plot_acf(df['GDP_QGR'], lags=12,
                         title='GDP Quarterly Growth rate ACF plot')
plt.show()
```



In the above figure, we plot the ACF and PACF, which is the ACF controlled for the other lagged correlations. The ACF tells us that the correlation of GDP growth with its lags is very low - hinting at very little time-dependence in this time-series. More precisely, the estimated correlation coefficients are not higher than 0.2 (for the lag of 1 period). If the GDP is indeed generated by an AR(p) process, the estimates show that the process has low ϕ's (in absolute value), indicating a low time dependence.

Part (b)

```
ar4 <- dynlm(df ~ L(df, 1) + L(df, 2) + L(df, 3) + L(df, 4))
ar3 <- dynlm(df ~ L(df, 1) + L(df, 3) + L(df, 4))
ar2 <- dynlm(df ~ L(df, 1) + L(df, 3))
ar1 <- dynlm(df ~ L(df, 1))

stargazer(ar4, ar3, ar2, ar1, type = "latex", header = FALSE)
```

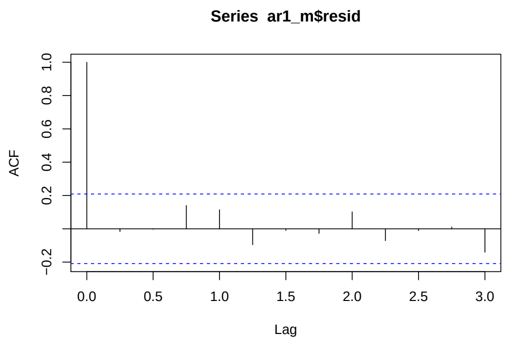
Table 1:

	Dependent variable:			
	df			
	(1)	(2)	(3)	(4)
L(df, 1)	0.232*	0.240*	0.257**	0.267**
	(0.124)	(0.122)	(0.119)	(0.117)
L(df, 2)	0.055			
	(0.126)			
L(df, 3)	0.203	0.210*	0.210*	
	(0.126)	(0.124)	(0.120)	
L(df, 4)	0.094	0.092		
	(0.125)	(0.124)		
Constant	0.479	0.533*	0.631***	0.896***
	(0.299)	(0.271)	(0.238)	(0.181)
Observations	84	84	85	87
R^2	0.099	0.097	0.089	0.058
Adjusted R^2	0.054	0.063	0.067	0.047
Residual Std. Error	0.902 (df = 79)	0.898 (df = 80)	0.891 (df = 82)	0.895 (df = 85)
F Statistic	2.181* (df = 4; 79)	2.873** (df = 3; 80)	4.019** (df = 2; 82)	5.229** (df = 1; 85)
Note:				*p<0.1; **p<0.05; ***p<0.01

```r
ar1_m <- arima(df, c(1, 0, 0))
```

Part c

```r
# Part 3: Plot ACF of residuals
acf(ar1_m$resid, 12)
```



Part d

```r
# Part 4: Forecast AR model for 2 years
df_pred <- predict(ar1_m, n.ahead = 8)$pred
```

Part e

```r
# Part 5: Produce CI
df_ciu <- predict(ar1_m, n.ahead = 8)$pred + predict(ar1_m, n.ahead = 8)$se*1.96
df_cil <- predict(ar1_m, n.ahead = 8)$pred - predict(ar1_m, n.ahead = 8)$se*1.96
```

Part f

```r
# Part 6: Check normality
jb.norm.test(ar1_m$resid)
```

```
##
##  Jarque-Bera test for normality
##
## data:  ar1_m$resid
## JB = 26.298, p-value = 0.002
```

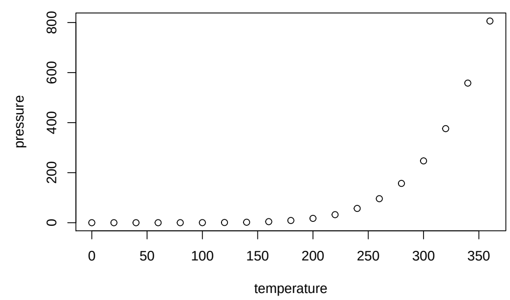
```r
# reject H0, innovations are not normally distributed
```

```python
import pandas as pd
import numpy as np

r.mtcars.sum()
```

```
## mpg       642.900
## cyl       198.000
## disp     7383.100
## hp       4694.000
## drat      115.090
## wt        102.952
## qsec      571.160
## vs         14.000
## am         13.000
## gear      118.000
## carb       90.000
## dtype: float64
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Including Matplotlib

```
hoi = np.arange(0,10)

hoi
```

```
## array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
#py$hoi
#gert::git_branch_checkout("attempt_bas")
#gert::git_add(c("*"))
#gert::git_commit(message = "Automatic commit")
#gert::git_push()
#gert::git_pull()
```