경제 자료 분석 기말 프로젝트
202STG01 고유정

데이터 출처 : World Bank
데이터 : 자동차 수요량 예측을 위한 연별 데이터(1971 - 2019)

A. 임의의 시계열 데이터를 구해서 회귀분석을 이용하여 회귀계수를 추정하고 95% 신뢰구간을 구하자. (33점)
예: 주요국가 (한국, 미국, 또는 전세계, 등등) 의 주요 상품 (oil, 금, 담배, 식료품, 또는 스마트폰, 등등) 의 수요탄력성 추정.
주의사항: 회귀분석에 있어서 변수누락, 오차항의 자기상관성, 이분산성 등을 논의하여 추정 탄력성 계수의 정밀도를 높이도록 노력한다.

1. 이분산성&자기상관성 검정

- OLSE

```
Call:
lm(formula = vehicle ~ price + income + driver +
transit + gdp +
    gdp_per + tax + taxp + rail)

Residuals:
      Min        1Q    Median        3Q       Max
-82367046  -9324623    812451   8663995 146314322

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.186e+08  3.385e+08   0.941   0.3523
price       -2.276e+07  1.143e+07  -1.991   0.0535 .
income      -2.500e+04  1.509e+04  -1.656   0.1057
driver      -3.907e+00  1.896e+00  -2.061   0.0460 *
transit     -8.124e+04  3.240e+04  -2.507   0.0164 *
gdp         -1.524e-05  2.243e-05  -0.679   0.5009
gdp_per      4.139e+04  1.681e+04   2.462   0.0183 *
tax         -6.155e-05  6.156e-05  -1.000   0.3235
taxp         3.804e+06  3.029e+06   1.256   0.2167
rail        -1.632e+05  1.087e+05  -1.501   0.1414
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1

Residual standard error: 31240000 on 39 degrees of
freedom
Multiple R-squared:  0.4347,	Adjusted R-squared:
0.3043
F-statistic: 3.333 on 9 and 39 DF,  p-value: 0.004137

> t_value
[1] -1.991219
```

```
library(sandwich)
data <- read.csv("transport.csv", header=T)
colnames(data)<-c("year","passenger","price","income","driver", "vehicle",

"transit","oilprice","gdp","gdp_per","taxp","tax","rail")  # on highway
dat <- data[,c(1,3:7,9:13)]

attach(dat)
ols.fit = lm(vehicle ~ price + income + driver + transit + gdp + gdp_per + tax + taxp + rail) ;ols.fit
summary(ols.fit)
OLS.se = summary(ols.fit)$coefficients[2,2]
t_value=summary(ols.fit)$coefficients[2,1]/summary(ols.fit)$coefficients[2,2]
t_value
```

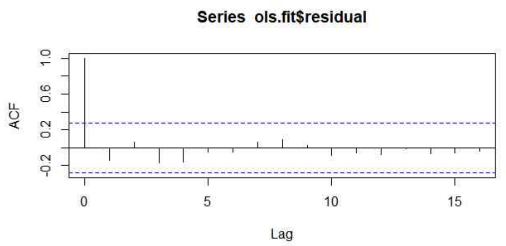
t값의 절대값이 1.96보다 크므로 H0를 기각한다.
이분산성과 자기상관성을 고려하지 않았기에 olse 추정을 신뢰할 수 없다.

- HC

<table>
<tr><td>> delta
[1] -22756550
> t_HC
[1] -0.9548597</td><td>HC.se = sqrt(vcovHC(ols.fit)[2,2])
delta = summary(ols.fit)$coefficients[2,1]
t_HC = delta / HC.se</td></tr>
</table>

OLS 보다 t value가 감소하였고 이분산성을 고려했다.

- HAC

<table>
<tr><td>
</td><td>rho1 = acf(ols.fit$residual)[1] #0.928
rho.se = 1/sqrt(nrow(dat))
HAC.se = sqrt(vcovHAC(ols.fit)[2,2])
t_HAC = delta / HAC.se
t_HAC
[1] -1.744739</td></tr>
</table>

HAC은 자기상관성을 고려했다.
t value < 1.96이므로 자기상관성 갖는다.

[white test]

<table>
<tr><td>Call:
lm(formula = e.sq ~ gdp_per + transit + price + driver + tax +
    gdp.sq + transit.sq + P.sq + tax.sq + drive.sq + gdp_per:transit +
    gdp_per:price + gdp_per:driver + gdp_per:tax + transit:price +
    transit:driver + transit:tax + price:driver + price:tax +
    driver:tax, data = dat)

Residuals:
Min 1Q Median 3Q Max
-2.392e+15 -2.824e+14 -1.607e+13 4.213e+14 1.718e+15

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.000e+17 1.063e+17 1.882 0.07027 .
gdp_per 1.564e+13 7.527e+12 2.078 0.04701 *
transit -8.172e+13 4.805e+13 -1.701 0.10008
price -1.060e+17 3.377e+16 -3.140 0.00396 **
driver -2.443e+09 1.827e+09 -1.338 0.19182
tax 7.708e+04 1.970e+05 0.391 0.69862</td><td>lm.fit = lm(formula = vehicle ~ price + driver + transit + rail + gdp_per + income + rail + tax + taxp, data=dat)
summary(lm.fit)

dat$gdp.sq = dat$gdp_per^2
dat$transit.sq = dat$transit^2
dat$P.sq = dat$price^2
dat$tax.sq = dat$tax^2
dat$drive.sq = dat$driver^2
e.sq = lm.fit$residuals^2

lm.fit2 = lm(e.sq ~ gdp_per + transit + price + driver + tax
    + gdp.sq + transit.sq + P.sq + tax.sq + drive.sq
    + gdp_per:transit + gdp_per:price + gdp_per:driver + gdp_per:tax
    + transit:price + transit:driver + transit:tax + price:driver
    + price:tax + driver:tax, data=dat)

summary(lm.fit2)
#qchisq(0.99, df=10)

qchisq(0.95, df=10)
summary(lm.fit2)$r.square*nrow(dat)</td></tr>
</table>

```
gdp.sq              4.186e+08   9.122e+07     4.589  8.53e-05
***
transit.sq          9.618e+09   6.029e+09     1.595   0.12185

P.sq                6.110e+15   1.186e+15     5.153  1.82e-05
***
tax.sq              6.908e-08   2.134e-08     3.238   0.00309
**
drive.sq            7.793e+00   8.763e+00     0.889   0.38138

gdp_per:transit  -4.509e+09   1.718e+09    -2.624   0.01390
*
gdp_per:price     -2.925e+12   6.428e+11    -4.551  9.46e-05
***
gdp_per:driver    -8.251e+04   7.275e+04    -1.134   0.26634

gdp_per:tax       -7.706e+00   3.109e+00    -2.478   0.01949
*
transit:price      1.357e+13   4.398e+12     3.086   0.00453
**
transit:driver    4.587e+05   4.129e+05    1.111   0.27605
transit:tax       6.704e+01   2.889e+01    2.320   0.02784 *
price:driver      6.567e+08   3.005e+08    2.185   0.03740 *
price:tax         1.526e+04   7.406e+03    2.061   0.04871 *
driver:tax       -1.103e-03   1.753e-03   -0.629   0.53425

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
1

Residual standard error: 8.894e+14 on 28 degrees of
freedom
Multiple R-squared:  0.9563,      Adjusted   R-squared:
0.9251
F-statistic: 30.64 on 20 and 28 DF,  p-value: 5.195e-14

> qchisq(0.95, df=10)
[1] 18.30704
> summary(lm.fit2)$r.square*nrow(dat)
[1] 46.85902
```

n*R^2은 카이제곱값보다 크므로 유의성이 높다고 할 수 있으며 오차항은 등분산 가정을 위배한다. 그러나 자유도가 크기에 검정력이 낮다.

- FGLSE

```
Call:
lm(formula = y.str ~ x1.str + x2.str + x3.str + x4.str)

Residuals:
    Min       1Q   Median       3Q      Max
-0.6637 -0.2855 -0.1140   0.0235   6.0112

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.348e-01   5.460e-01   -0.613    0.5429
```

```
> lm.e = lm(e.sq ~ dat$price)
> sigma.hat = sqrt(abs(lm.e$fitted.value))
> y.str = dat$vehicle/sigma.hat
> x1.str = dat$gdp_per/sigma.hat
> x2.str = dat$transit/sigma.hat
> x3.str = dat$price/sigma.hat
> x4.str = dat$driver/sigma.hat
> FGLS.fit = lm(y.str ~ x1.str + x2.str + x3.str + x4.str)
> summary(FGLS.fit)
```

x1.str 1.083e+03 6.334e+02 1.709 0.0945 . x2.str -1.795e+04 1.516e+04 -1.184 0.2428 x3.str -4.166e+06 1.153e+07 -0.361 0.7196 x4.str 3.425e-01 2.220e-01 1.543 0.1300 --- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 Residual standard error: 0.9582 on 44 degrees of freedom Multiple R-squared: 0.2687, Adjusted R-squared: 0.2022 F-statistic: 4.041 on 4 and 44 DF, p-value: 0.00708 bptest(FGLS.fit, ~ ln_transit, data=dat, studentize=F) Breusch-Pagan test data: FGLS.fit BP = 31.872, df = 1, p-value = 1.647e-08	

종속변수는 연별 승용차 판매량, 설명변수는 영향력을 regression 결과를 토대로 원유 가격, 대중교통 이용량, 1인당 g에 그리고 면허가 있는 운전자수로 지정했다.
FGLSE fitting 후 bptest 결과 p-value는 1에 가깝게 나왔다.

- 더빈 왓슨 검정

> dwtest(ols.fit) Durbin-Watson test data: ols.fit DW = 1.7126, p-value = 0.014 alternative hypothesis: true autocorrelation is greater than 0 > dwtest(FGLS.fit) Durbin-Watson test data: FGLS.fit DW = 1.119, p-value = 9.748e-05 alternative hypothesis: true autocorrelation is greater than 0	더빈 왓슨 검정결과, ols에 비해 FGLS를 검정했을 때 자기상관이 더 완화됐다. 따라서 신뢰구간을 예측할 때 이분산성, 자기상관성을 고려하며 자기상관성을 확실히 낮추며 등분산변환이 시행된 FGLS를 사용했다. 따라서 아래에 신뢰구간을 구할 때 FGLS를 이용하여 구했다.

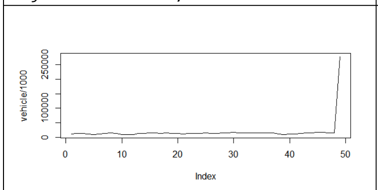
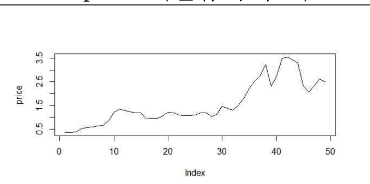
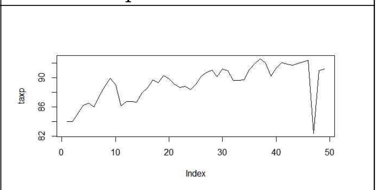
- 수요탄력성

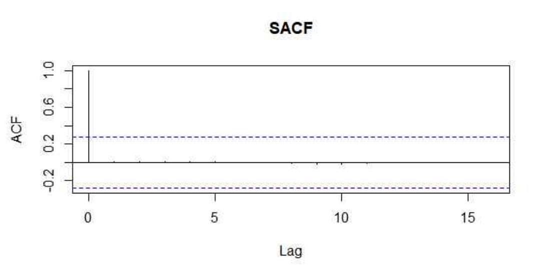
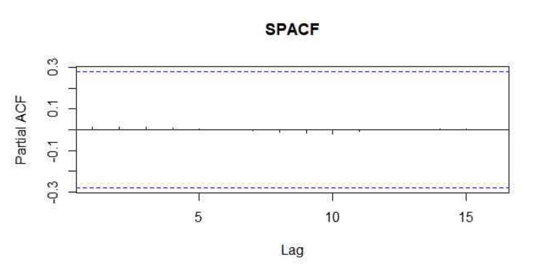
> library(ivreg);library(AER) > ivreg.fit = ivreg(log(vehicle) ~ log(price) + log(gdp)|taxp + log(gdp)) > summary(ivreg.fit) Call: ivreg(formula = log(vehicle) ~ log(price) + log(gdp) | taxp +	도구변수로 taxp, 즉 소비세가 총세금에서 차지하는 비율을 사용하고 내생을 완화시키기 위해 설명변수로 log소득(gdp에 로그 취함)을 사용했다. 변수 추가 전보다 수요탄력성이 높아지고

log(gdp)) Residuals: Min 1Q Median 3Q Max -0.49712 -0.15645 -0.04830 0.04532 2.60682 Coefficients: Estimate Std. Error t value Pr(>|t|) (Intercept) 0.5850 35.0991 0.017 0.987 log(price) -0.5605 1.8599 -0.301 0.765 log(gdp) 0.5460 1.2061 0.453 0.653 > confint(ivreg.fit, level=0.95, int="confi") 2.5 % 97.5 % (Intercept) -20.963598 40.765778 ln_P -4.644391 3.473967 ln_gdp -2.483974 3.828338	정밀도도 높였다. 수요탄력성은 -0.56으로 나오며 이는 가격이 1% 증가할수록 자동차소비량은 0.56% 줄어든다는 것을 시사한다. 위에 FGLS를 선택했기에 FGLS를 이용해 신뢰구간을 구했다.

B. 위 A 의 종속변수에 대해 시계열 모형 (ARIMA 모형, ADL 모형, 그리고 VAR(또는 VEC) 모형) 을 이용하여 향후 2년 (연도별 데이터의 경우), 8분기 (분기별 데이터의 경우), 또는 24개월(월별데이터의 경우) 그 상품의 수요를 예측하고 예측구간을 구하자. 또 이 세모형(ARIMA 모형, ADL 모형, VAR(또는 VEC) 모형)의 예측력을 비교한다. (33점)
주의사항: ADL, VAR(또는 VEC) 모형) 모형의 경우 추가 예측 변수는 최대 두개만 고려한다.

- 시계열 그래프

y = vehicle/1000	x1 = price (원유가격 $)	x2 = taxp
		

price는 추세가 있으며 종속변수와 taxp는 뚜렷한 추세가 없다.

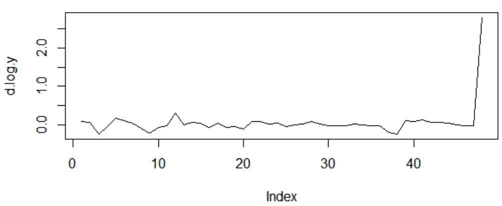
- ADF test

y	Title:
	 Augmented Dickey-Fuller Test
	
	Test Results:
	  PARAMETER:
	    Lag Order: 1
	  STATISTIC:
	    Dickey-Fuller: 1.185
	  P VALUE:
	    0.99

x1(price)	Title:
	 Augmented Dickey-Fuller Test
	
	Test Results:
	  PARAMETER:
	    Lag Order: 1
	  STATISTIC:
	    Dickey-Fuller: -2.5039
	  P VALUE:
	    0.3725
	확정적 추세를 보인다.

x2(taxp)	Title:
	 Augmented Dickey-Fuller Test
	
	Test Results:
	  PARAMETER:
	    Lag Order: 1
	  STATISTIC:
	    Dickey-Fuller: -3.0745
	  P VALUE:
	    0.03816

- ARIMA 모형

	> log.y = ln_vehicle; n=length(log.y); d.log.y=log.y[2:n]-log.y[1:(n-1)]
	> plot(d.log.y, type="l") #로그계열 차분
	> library(forecast)
	> aic1 = matrix(rep(0, 5*5), 5,5); bic1=matrix(rep(0, 5*5), 5,5)
	> for (p in 1:5){for (q in 1:5)
	+ {aic1[p,q] = Arima(d.log.y, order=c(p-1,0,q-1))$aic
	+ bic1[p,q] = Arima(d.log.y, order=c(p-1,0,q-1))$bic}}
	> min(aic1) # 1,1 -> AIC order = (0,0)
	[1] 54.41175
	> min(bic1) # 1,1 BIC order = (0,0) AR(0)
	[1] 58.15415
	> arima.fit = Arima(log.y, order=c(1,1,1))
	> confint(arima.fit, level=0.95, int = "predi")
	       2.5 % 97.5 %
	ar1 -4.113010 3.707575
	ma1 -3.264089 4.024108
	> arima.hat
	   Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
	50 19.93198 19.39060 20.47335 19.10402 20.75993
	51 19.83178 18.99553 20.66802 18.55285 21.11070

AIC order와 BIC order를 구해 arima 모형에 fitting을 하고 향후 2년 소비를 예측하고 신뢰구간을 구했다.
y(종속변수)는 가독성과 fitting을 용이하게 하기위해 원래 종속변수인 자동차 판매량을 1000으로 나눠 로그를 취해주었기에 이를 감안하면 판매량 예측값이 나온다.

- VAR/VEC 모형

<table>
<tr>
<td>

```
> aic = matrix(rep(0, 5*5), 5,5); bic=matrix(rep(0, 5*5), 5,5)
> for (p in 1:5){for (q in 1:5)
+ {aic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$aic
+ bic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$bic}}
> which.min(aic) # 2,1 -> AIC order = (1,1)
[1] 2
>
> Data = data.frame(vehicle/1000, price, taxp)
> johanson.test = ca.jo(Data, type="eigen", ecdet="const")
> summary(johanson.test) # 1%에서 기각 -> 10%에서 기각 못함 -> rank=1

######################
# Johansen-Procedure #
######################

Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegration

Eigenvalues (lambda):
[1] 5.181274e-01 1.043474e-01 3.403240e-02 8.560011e-17

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  1.63  7.52  9.24 12.97
r <= 1 |  5.18 13.75 15.67 20.20
r = 0  | 34.31 19.77 22.00 26.81

Eigenvectors, normalised to first column:
(These are the cointegration relations)

                vehicle.1000.l2       price.l2      taxp.l2     constant
vehicle.1000.l2           1.000           1.00       1.0000     1.000000
price.l2               4701.916      -63982.82    -924.6472  -932.226946
taxp.l2               -2418.618      -12332.22     650.1860    -2.260104
constant             194376.897     1205032.44  -67916.9245
```

</td>
<td>

eigen value와 rank를 추정했다.
VEC 모형을 추정한 결과 rank=1 즉 공적분관계가 1개 있음을 알 수 있다.

이를 이용하여 향후2년 판매량과 신뢰구간을 예측했다.

</td>
</tr>
</table>

```
-14255.266438

Weights W:
(This is the loading matrix)

                 vehicle.1000.l2        price.l2         taxp.l2
     constant
vehicle.1000.d               5.528043e+00    -2.378002e-02
5.241783e-01   6.597433e-13
price.d                     -1.244063e-05     7.559633e-07
9.282292e-06  -1.692745e-18
taxp.d                       2.504867e-04     5.299816e-06
-2.774620e-05   3.604675e-17

> library(tsDyn);library(vars)
> bic=c()
> for (p in 1:10){
+    bic[p] = summary(vecm.fit)$bic}
40건의 경고들이 발견되었습니다 (이를 확인하기 위해서는
warnings()를 이용하시길 바랍니다).
> which.min(bic) # 1 : BIC order
[1] 1
>
> var.form <- vec2var(johanson.test, r= 1)
>
> var.fit = VAR(Data, lag=1)
> summary(var.fit)

VAR Estimation Results:
=========================
Endogenous variables: vehicle.1000, price, taxp
Deterministic variables: const
Sample size: 48
Log Likelihood: -669.305
Roots of the characteristic polynomial:
3.643 0.926 0.3684
Call:
VAR(y = Data, lag.max = 1)


Estimation results for equation vehicle.1000:
=============================================
vehicle.1000 = vehicle.1000.l1 + price.l1 + taxp.l1 +
const

                  Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1      3.729      2.885   1.293    0.203
price.l1          8404.371   7903.180   1.063    0.293
taxp.l1          -1252.853   3039.528  -0.412    0.682
const            64174.619 253515.680   0.253    0.801


Residual standard error: 37700 on 44 degrees of
freedom
Multiple R-Squared: 0.07065,     Adjusted   R-squared:
0.00728
F-statistic: 1.115 on 3 and 44 DF,  p-value: 0.3533
```

```
Estimation results for equation price:
======================================
price = vehicle.1000.l1 + price.l1 + taxp.l1 + const

                  Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1   2.751e-06   2.210e-05    0.125     0.901
price.l1           9.313e-01   6.054e-02   15.384    <2e-16
***
taxp.l1            5.460e-03   2.328e-02    0.234     0.816

const             -3.767e-01   1.942e+00   -0.194     0.847

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1


Residual standard error: 0.2888 on 44 degrees of
freedom
Multiple R-Squared: 0.9013,     Adjusted R-squared:
0.8946
F-statistic: 133.9 on 3 and 44 DF,  p-value: < 2.2e-16


Estimation results for equation taxp:
=====================================
taxp = vehicle.1000.l1 + price.l1 + taxp.l1 + const

                 Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1 2.568e-04   1.306e-04    1.966   0.05566 .
price.l1        1.005e+00   3.579e-01    2.809   0.00739 **
taxp.l1         2.769e-01   1.377e-01    2.012   0.05039 .
const             5.930e+01   1.148e+01    5.166  5.57e-06
***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1


Residual standard error: 1.707 on 44 degrees of
freedom
Multiple R-Squared: 0.5034,     Adjusted R-squared:
0.4695
F-statistic: 14.86 on 3 and 44 DF,  p-value: 8.016e-07



Covariance matrix of residuals:
             vehicle.1000      price       taxp
vehicle.1000     1.421e+09 -700.54242 -1.953e+03
price           -7.005e+02    0.08338  6.997e-02
taxp            -1.953e+03    0.06997  2.914e+00

Correlation matrix of residuals:
             vehicle.1000    price     taxp
vehicle.1000      1.00000 -0.06436 -0.03035
price            -0.06436  1.00000  0.14194
```

taxp -0.03035 0.14194 1.00000 > > var.hat = predict(var.fit, n.ahead=2) > var.hat $vehicle.1000 fcst lower upper CI [1,] 1002129 928243.3 1076014 73885.35 [2,] 3630488 3345316.5 3915660 285171.67 $price fcst lower upper CI [1,] 3.209987 2.644030 3.775943 0.5659564 [2,] 6.232830 5.440056 7.025604 0.7927741 $taxp fcst lower upper CI [1,] 158.0644 154.7183 161.4104 3.346028 [2,] 363.6461 344.4090 382.8831 19.237044	

- ADL 모형

> aic = matrix(rep(0, 5*5), 5,5); bic=matrix(rep(0, 5*5), 5,5) > for (p in 1:5){for (q in 1:5) + {aic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$aic + bic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$bic}} > which.min(aic) # 2,1 -> AIC order = (1,1) [1] 2 > > Data = data.frame(vehicle/1000, price, taxp) > johanson.test = ca.jo(Data, type="eigen", ecdet="const") > summary(johanson.test) # 1%에서 기각 -> 10%에서 기각 못함 -> rank=1 ###################### # Johansen-Procedure # ###################### Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegration Eigenvalues (lambda): [1] 5.181274e-01 1.043474e-01 3.403240e-02 8.560011e-17 Values of teststatistic and critical values of test: test 10pct 5pct 1pct r <= 2 | 1.63 7.52 9.24 12.97 r <= 1 | 5.18 13.75 15.67 20.20 r = 0 | 34.31 19.77 22.00 26.81 Eigenvectors, normalised to first column: (These are the cointegration relations)	> adl.hat = predict(lm.fit4, n.ahead=2) > confint(lm.fit4, level=0.95, int="predi") 2.5 % 97.5 % (Intercept) -637321.931 631204.210 dx_1 -56553.883 25521.576 dx_2 -5936.212 6853.151 price_st -10764.155 24938.930 taxp_st -7341.832 7295.570 ADL 모형또한 같은 방식으로 공적분 관계를 확인하고 수요 예측치와 예측구간을 구하였다.

```
                  vehicle.1000.l2       price.l2          taxp.l2
constant
vehicle.1000.l2             1.000          1.00         1.0000
  1.000000
price.l2                 4701.916   -63982.82     -924.6472
-932.226946
taxp.l2                 -2418.618   -12332.22      650.1860
  -2.260104
constant               194376.897  1205032.44  -67916.9245
-14255.266438

Weights W:
(This is the loading matrix)

                 vehicle.1000.l2        price.l2          taxp.l2
     constant
vehicle.1000.d              5.528043e+00     -2.378002e-02
5.241783e-01   6.597433e-13
price.d                    -1.244063e-05      7.559633e-07
9.282292e-06  -1.692745e-18
taxp.d                      2.504867e-04      5.299816e-06
-2.774620e-05   3.604675e-17

> library(tsDyn);library(vars)
> bic=c()
> for (p in 1:10){
+    bic[p] = summary(vecm.fit)$bic}
40건의 경고들이 발견되었습니다 (이를 확인하기 위해서는
warnings()를 이용하시길 바랍니다).
> which.min(bic) # 1 : BIC order
[1] 1
>
> var.form <- vec2var(johanson.test, r= 1)
>
> var.fit = VAR(Data, lag=1)
> summary(var.fit)

VAR Estimation Results:
=========================
Endogenous variables: vehicle.1000, price, taxp
Deterministic variables: const
Sample size: 48
Log Likelihood: -669.305
Roots of the characteristic polynomial:
3.643 0.926 0.3684
Call:
VAR(y = Data, lag.max = 1)


Estimation results for equation vehicle.1000:
=============================================
vehicle.1000 = vehicle.1000.l1 + price.l1 + taxp.l1 +
const

                     Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1        3.729        2.885    1.293      0.203
price.l1            8404.371     7903.180    1.063      0.293
```

```
taxp.l1              -1252.853    3039.528   -0.412      0.682
const                64174.619  253515.680    0.253      0.801


Residual standard error: 37700 on 44 degrees of freedom
Multiple R-Squared: 0.07065,    Adjusted R-squared: 0.00728
F-statistic: 1.115 on 3 and 44 DF,  p-value: 0.3533


Estimation results for equation price:
======================================
price = vehicle.1000.l1 + price.l1 + taxp.l1 + const

                 Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1  2.751e-06  2.210e-05   0.125    0.901
price.l1         9.313e-01  6.054e-02  15.384   <2e-16 ***
taxp.l1          5.460e-03  2.328e-02   0.234    0.816
const           -3.767e-01  1.942e+00  -0.194    0.847
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.2888 on 44 degrees of freedom
Multiple R-Squared: 0.9013,    Adjusted R-squared: 0.8946
F-statistic: 133.9 on 3 and 44 DF,  p-value: < 2.2e-16


Estimation results for equation taxp:
=====================================
taxp = vehicle.1000.l1 + price.l1 + taxp.l1 + const

                 Estimate Std. Error t value Pr(>|t|)
vehicle.1000.l1 2.568e-04  1.306e-04   1.966  0.05566 .
price.l1        1.005e+00  3.579e-01   2.809  0.00739 **
taxp.l1         2.769e-01  1.377e-01   2.012  0.05039 .
const           5.930e+01  1.148e+01   5.166  5.57e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 1.707 on 44 degrees of freedom
Multiple R-Squared: 0.5034,    Adjusted R-squared: 0.4695
F-statistic: 14.86 on 3 and 44 DF,  p-value: 8.016e-07
```

```
Covariance matrix of residuals:
             vehicle.1000      price      taxp
vehicle.1000    1.421e+09 -700.54242 -1.953e+03
price          -7.005e+02    0.08338  6.997e-02
taxp           -1.953e+03    0.06997  2.914e+00

Correlation matrix of residuals:
             vehicle.1000    price     taxp
vehicle.1000      1.00000 -0.06436 -0.03035
price            -0.06436  1.00000  0.14194
taxp             -0.03035  0.14194  1.00000



>
> var.hat = predict(var.fit, n.ahead=2)
> var.hat
$vehicle.1000
        fcst     lower   upper        CI
[1,] 1002129  928243.3 1076014  73885.35
[2,] 3630488 3345316.5 3915660 285171.67

$price
         fcst    lower    upper        CI
[1,] 3.209987 2.644030 3.775943 0.5659564
[2,] 6.232830 5.440056 7.025604 0.7927741

$taxp
         fcst    lower    upper        CI
[1,] 158.0644 154.7183 161.4104  3.346028
[2,] 363.6461 344.4090 382.8831 19.237044

> aic=c()
> for(p in 1:10){
+     ar.fit = Arima(price, order = c(p,0,0), method="ML")
+     aic[p] = ar.fit$aic}
> which.min(aic)
[1] 1
> # 공적분의 검토
> y=dat$vehicle/1000
> reg = lm(y ~ price + taxp)
> z=reg$residual
> z
        1         2         3         4         5
       6
 3104.737  4336.737  4225.467  -437.013 -1380.099
 1132.344
        7         8         9        10        11
       12
 1177.588   623.366 -2809.595 -6589.023 -5571.355
-5919.649
       13        14        15        16        17
       18
-1888.724 -1667.387 -1645.306  -179.277 -2319.007
-1482.573
       19        20        21        22        23
       24
-3744.652 -4930.217 -5674.798 -3494.888 -2345.709
-1447.031
```

```
         25          26          27          28          29 
        30 
 -1591.836  -3784.012  -4152.200  -3097.690  -1529.596 
-3745.740 
         31          32          33          34          35 
        36 
 -3395.325  -2184.061  -3424.715  -4855.703  -8202.376 
-10855.843 
         37          38          39          40          41 
        42 
-13148.395 -17877.026 -14484.721 -16496.082 -19878.336 
-18376.934 
         43          44          45          46          47 
        48 
-16577.845 -15050.579  -9109.135  -7791.632  -1184.824 
-10102.061 
         49 
249822.732 
> library(forecast)
> aic=c()
> for(p in 1:10){
+     ar.z = Arima(z, order=c(p,0,0), method="ML")
+     aic[p] = ar.z$aic}
> which.min(aic) # 1
[1] 1
> adfTest(z, type="c", lags=0) # 유의함 공적분관계 있음

Title:
 Augmented Dickey-Fuller Test

Test Results:
  PARAMETER:
    Lag Order: 0
  STATISTIC:
    Dickey-Fuller: -0.9398
  P VALUE:
    0.7033 

Description:
 Mon Jun 14 00:02:50 2021 by user: yjk9

> dy = diff(y)
> dx_1 = diff(price)
> dx_2 = diff(taxp)
> library(MASS)
> lm.fit = lm(dy ~ dx_1 + dx_2)
> summary(lm.fit)

Call:
lm(formula = dy ~ dx_1 + dx_2)

Residuals:
   Min     1Q Median     3Q    Max 
-19285  -6430  -5283  -3327 251730 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6005.8     5577.5   1.077    0.287
```

```
dx_1          -12428.5     19632.9   -0.633     0.530
dx_2             343.9      2645.1    0.130     0.897

Residual standard error: 38110 on 45 degrees of
freedom
Multiple R-squared: 0.008928,    Adjusted R-squared:
-0.03512
F-statistic: 0.2027 on 2 and 45 DF,  p-value: 0.8173

> full.model <- lm.fit
> step.model <- stepAIC(full.model, direction="both",
trace=FALSE)
> summary(step.model)

Call:
lm(formula = dy ~ 1)

Residuals:
   Min     1Q Median     3Q    Max
 -8535  -5940  -5342  -4440 253934

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     5503       5406   1.018    0.314

Residual standard error: 37460 on 47 degrees of
freedom

> nrow(dat)*4
[1] 196
> z.fit=lm(y~price + taxp, data=dat)
> z=z.fit$residuals[1:(49-1)]
> lm.fit3 <- lm(dy ~ dx_1 + dx_2+z)
> summary(lm.fit3)

Call:
lm(formula = dy ~ dx_1 + dx_2 + z)

Residuals:
   Min     1Q Median     3Q    Max
-25505  -6574  -3771  -1621 248552

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.076e+03  7.633e+03   0.272    0.787
dx_1        -1.010e+04  1.996e+04  -0.506    0.615
dx_2         7.549e+02  2.712e+03   0.278    0.782
z           -7.232e-01  9.539e-01  -0.758    0.452

Residual standard error: 38290 on 44 degrees of
freedom
Multiple R-squared: 0.02171,    Adjusted R-squared:
-0.04499
F-statistic: 0.3255 on 3 and 44 DF,  p-value: 0.8069

> price_st <- price[-1]
> taxp_st <- taxp[-1]
> taxp <- dat$taxp
```

```
> lm.fit4 <- lm(dy ~ dx_1 + dx_2 + price_st + taxp_st)
> summary(lm.fit4)

Call:
lm(formula = dy ~ dx_1 + dx_2 + price_st + taxp_st)

Residuals:
   Min     1Q Median     3Q    Max
-27126  -6273  -2974   -644 244754

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3058.86  314506.63  -0.010    0.992
dx_1         -15516.15   20349.03  -0.763    0.450
dx_2            458.47    3170.88   0.145    0.886
price_st       7087.39    8851.89   0.801    0.428
taxp_st         -23.13    3629.06  -0.006    0.995

Residual standard error: 38440 on 43 degrees of freedom
Multiple R-squared:  0.03639,	Adjusted R-squared:  -0.05325
F-statistic: 0.4059 on 4 and 43 DF,  p-value: 0.8033
```

3가지 모형을 비교한 결과 공적분을 고려하며 차분계열을 사용해 Stepwise selection을 하고 잔차회귀까지 정교하게 시행하여 예측을 실행한 ADL 모형의 예측력이 가장 좋다.

<R code>

```
############## A : 회귀계수 & 신뢰구간(95%) ################

# 이분산성 자기상관 무시한 OLS 분석
library(sandwich)
data <- read.csv("transport.csv", header=T)
colnames(data)<-c("year","passenger","price","income","driver", "vehicle",
                  "transit","oilprice","gdp","gdp_per","taxp","tax","rail") # on highway
dat <- data[,c(1,3:7,9:13)]

attach(dat)
plot(price, type="l")

#ols.fit = lm(ln_vehicle ~ ln_P + ln_gdp + ln_taxp + ln_transit) ;ols.fit
ols.fit = lm(vehicle ~ price + income + driver + transit + gdp + gdp_per + tax + taxp + rail) ;ols.fit
summary(ols.fit)
OLS.se = summary(ols.fit)$coefficients[2,2]
t_value=summary(ols.fit)$coefficients[2,1]/summary(ols.fit)$coefficients[2,2]
t_value     # t_value 2.58 > 1.96 = H0기각

# 이분산성 감안한 t검정. 기각.
HC.se = sqrt(vcovHC(ols.fit)[2,2])
delta = summary(ols.fit)$coefficients[2,1]
t_HC = delta / HC.se # 3.43 > 1.96 줄어듦. 유의하다

# 자기상관성 감안한 HAC
rho1 = acf(ols.fit$residual)[1] #0.928
rho.se = 1/sqrt(nrow(dat)) # 0.143 < 1.96 H0채택. 데이터 오차항 독립이다

HAC.se = sqrt(vcovHAC(ols.fit)[2,2])
t_HAC = delta / HAC.se
t_HAC
# white test
```

```
lm.fit = lm(formula = vehicle ~ price + driver + transit + rail + gdp_per + income + rail + tax + taxp, data=dat)
summary(lm.fit)

dat$gdp.sq = dat$gdp_per^2
dat$transit.sq = dat$transit^2
dat$P.sq = dat$price^2
dat$tax.sq = dat$tax^2
dat$drive.sq = dat$driver^2
e.sq = lm.fit$residuals^2

lm.fit2 = lm(e.sq ~ gdp_per + transit + price + driver + tax
              + gdp.sq + transit.sq + P.sq + tax.sq + drive.sq
              + gdp_per:transit + gdp_per:price + gdp_per:driver + gdp_per:tax
              + transit:price + transit:driver + transit:tax + price:driver
              + price:tax + driver:tax, data=dat)

summary(lm.fit2)
#qchisq(0.99, df=10)

qchisq(0.95, df=10)
summary(lm.fit2)$r.square*nrow(dat) # 33.4523 > chi(10) 등분산가설 기각. 오차항이 등분산 아니다.

# FGLS 추정
lm.e = lm(e.sq ~ dat$price)
sigma.hat = sqrt(abs(lm.e$fitted.value))

y.str = dat$vehicle/sigma.hat
x1.str = dat$gdp_per/sigma.hat
x2.str = dat$transit/sigma.hat
x3.str = dat$price/sigma.hat
x4.str = dat$driver/sigma.hat

FGLS.fit = lm(y.str ~ x1.str + x2.str + x3.str + x4.str)
summary(FGLS.fit)
library(fUnitRoots);library(lmtest)
bptest(FGLS.fit, ~ ln_transit, data=dat, studentize=F) # 이분산성 없다

# 오차항 자기상관 검정, 더빈왓슨 검정
dwtest(ols.fit) # 양의 자기상관
dwtest(FGLS.fit) # 자기상관 조금 완화됐음

#income/vehicle/price, gdp/vehicle/price, gdpper/vehicle/price
ols.fit
y = vehicle


# Ivreg
library(ivreg);library(AER)
ivreg.fit = ivreg(log(vehicle) ~ log(price) + log(gdp)|taxp + log(gdp))
ivreg.fit # 가격 탄력성 계수 -0.38 가격 1% 오르면 차 소비량 0.38% 내림
summary(ivreg.fit)

# FGLS 선택 -> ols는 유의성 과장되고, 등분산 변환
confint(ivreg.fit, level=0.95, int="confi") # 신뢰구간


############## B ################

plot(vehicle/1000, type="l")
plot(price, type="l")
plot(taxp, type="l")
# adf 검정
acf(y, main="SACF")
pacf(y, main="SPACF")

adfTest(y/1000, type="c", lag=1)
adfTest(price, type="ct", lag=1)
adfTest(taxp, type="c", lag=1)
```

```
# ARIMA 모형으로 향후 2년 검토 추세 있음
log.y = log(vehicle); n=length(log.y); d.log.y=log.y[2:n]-log.y[1:(n-1)]
plot(d.log.y, type="l") #로그계열 차분
library(forecast)
aic1 = matrix(rep(0, 5*5), 5,5); bic1=matrix(rep(0, 5*5), 5,5)
for (p in 1:5){for (q in 1:5)
   {aic1[p,q] = Arima(d.log.y, order=c(p-1,0,q-1))$aic
    bic1[p,q] = Arima(d.log.y, order=c(p-1,0,q-1))$bic}}
min(aic1) # 1,1 -> AIC order = (0,0)
min(bic1) # 1,1 BIC oder = (0,0) AR(0)

arima.fit = Arima(log.y, order=c(1,1,1))
arima.fit=Arima(log.y, order=c(1,1,1))
arima.hat=forecast(arima.fit, h=2)
confint(arima.fit, level=0.95, int = "predi")
arima.hat
######################## VAR/VEC####################################
library(mvtnorm);library(urca)
Data = data.frame(vehicle/1000, price, taxp)
johanson.test = ca.jo(Data, type="eigen", ecdet="const")
summary(johanson.test) # rank =1
library(tsDyn)
vecm.fit=VECM(Data, lag=0, r=1, estim="ML", include = "none")
summary(vecm.fit)
beta = matrix(c(1, 1515.934, -169.138), nrow=1)
alpha = matrix(c(3.1224, 2.4e-06, 7.1e-05))
pi = alpha%*%beta

aic = matrix(rep(0, 5*5), 5,5); bic=matrix(rep(0, 5*5), 5,5)
for (p in 1:5){for (q in 1:5)
    {aic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$aic
    bic[p,q] = Arima(log(y/1000), order=c(p-1,0,q-1))$bic}}
which.min(aic) # 2,1 -> AIC order = (1,1)

Data = data.frame(vehicle/1000, price, taxp)
johanson.test = ca.jo(Data, type="eigen", ecdet="const")
summary(johanson.test) # 1%에서 기각 -> 10%에서 기각못함 -> rank=1
library(tsDyn);library(vars)
bic=c()
for (p in 1:10){
   bic[p] = summary(vecm.fit)$bic}
which.min(bic) # 1 : BIC order

var.form <- vec2var(johanson.test, r= 1)

var.fit = VAR(Data, lag=1)
summary(var.fit)

var.hat = predict(var.fit, n.ahead=2)
var.hat


# ADL

# adf 검정
library(fUnitRoots)
aic=c()
for(p in 1:10){
    ar.fit = Arima(price, order = c(p,0,0), method="ML")
    aic[p] = ar.fit$aic}
which.min(aic)
plot(dat$taxp, type="l")
adfTest(dat$vehicle, type="c")
adfTest(dat$price, type="ct")
adfTest(dat$taxp, type="c")

# 공적분의 검토
```

```
y=dat$vehicle/1000
attach(dat)
reg = lm(y ~ price + taxp)
z=reg$residual
z
library(forecast)
aic=c()
for(p in 1:10){
    ar.z = Arima(z, order=c(p,0,0), method="ML")
    aic[p] = ar.z$aic}
which.min(aic) # 1

adfTest(z, type="c", lags=0) # 유의함 공적분관계 있음
dy = diff(y)
dx_1 = diff(price)
dx_2 = diff(taxp)
library(MASS)
lm.fit = lm(dy ~ dx_1 + dx_2)
summary(lm.fit)

full.model <- lm.fit
step.model <- stepAIC(full.model, direction="both", trace=FALSE)
summary(step.model)
nrow(dat)*4
z.fit=lm(y~price + taxp, data=dat)
z=z.fit$residuals[1:(49-1)]

lm.fit3 <- lm(dy ~ dx_1 + dx_2+z)
summary(lm.fit3)

price_st <- price[-1]
taxp_st <- taxp[-1]
taxp <- dat$taxp
lm.fit4 <- lm(dy ~ dx_1 + dx_2 + price_st + taxp_st)
summary(lm.fit4)
adl.hat = predict(lm.fit4, n.ahead=2)
adl.hat
confint(lm.fit4, level=0.95, int="predi")
```