Econometrie II : Transports

Rapport

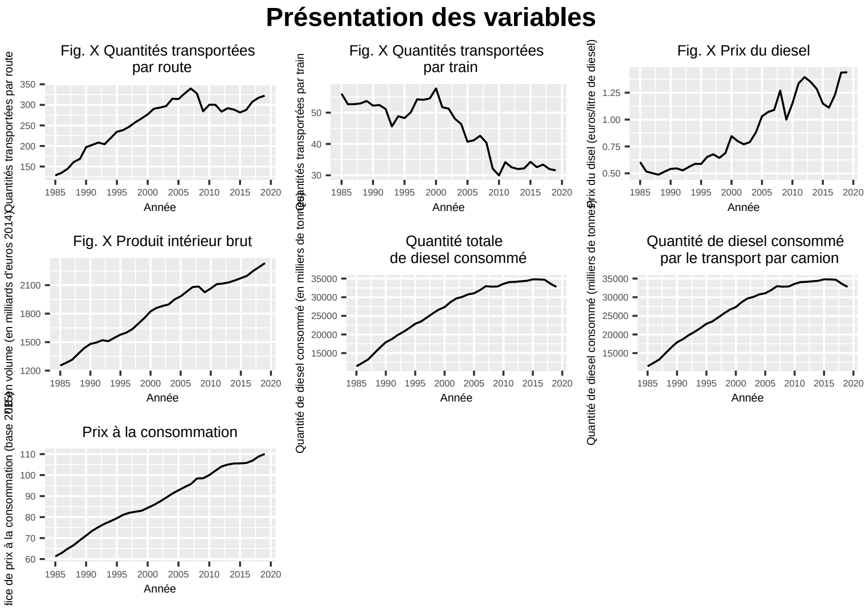
Guewen HESLAN - Bijan VALILOU

Janvier 2021

1 Introduction

Table 1: Consommation de gazole des camions

Year	Qtt__Trsp_tout	Qtt__Trsp_camion	P_diesel	QDiesel	GDP	CPI	Qdiesel-camion	PIB en volume (en milliards d'euros 2014)
1985	128.4177	56.05900	0.6036981	11467	7.576890e+11	61.27793	7999.565	1253.767
1986	134.5980	52.68600	0.5168022	12364	8.145960e+11	62.83349	8377.997	1283.071
1987	144.5150	52.70700	0.5015573	13309	8.559830e+11	64.90002	8874.912	1315.942
1988	161.1093	52.94600	0.4878369	14903	9.252150e+11	66.65285	10083.360	1378.359
1989	168.6379	53.71200	0.5152777	16472	9.971210e+11	68.98456	11026.464	1438.233
1990	197.0160	52.24000	0.5411940	17908	1.053546e+12	71.18813	12847.488	1480.286
1991	202.6669	52.43001	0.5457675	18729	1.091705e+12	73.47569	13088.602	1495.802
1992	208.3424	51.18059	0.5274736	19824	1.130983e+12	75.21248	13520.352	1519.725
1993	204.2418	45.58251	0.5594879	20711	1.142119e+12	76.79530	13736.333	1510.171
1994	219.2725	48.87126	0.5884532	21735	1.179867e+12	78.06666	13944.659	1545.786
1995	234.5019	48.26607	0.5869287	22869	1.218273e+12	79.46911	14175.760	1578.351
1996	238.5483	50.11300	0.6524818	23489	1.252266e+12	81.04489	14106.159	1600.653
1997	246.9545	54.24600	0.6768736	24566	1.292777e+12	82.02062	14718.880	1638.049
1998	257.6484	54.09952	0.6433349	25667	1.351896e+12	82.55468	15658.433	1696.833
1999	266.8618	54.53802	0.6890696	26667	1.400999e+12	82.99812	16588.176	1754.888
2000	276.8614	57.72575	0.8460920	27355	1.478585e+12	84.38913	16920.644	1823.744
2001	290.4301	51.71830	0.8000000	28684	1.538200e+12	85.76871	16846.184	1859.922
2002	293.3823	51.28819	0.7700000	29670	1.587829e+12	87.41840	17155.587	1881.042
2003	296.9908	48.05727	0.7900000	30081	1.630666e+12	89.25285	17034.634	1896.526
2004	314.9008	46.34837	0.8800000	30762	1.704019e+12	91.16472	17539.546	1950.193
2005	314.1489	40.70118	1.0300000	31048	1.765905e+12	92.75634	17707.320	1982.629
2006	327.6145	41.17892	1.0700000	31891	1.848151e+12	94.31012	18073.350	2031.190
2007	339.9549	42.61186	1.0900000	32958	1.941360e+12	95.71346	17911.820	2080.441
2008	327.4415	40.43613	1.2700000	32827	1.992380e+12	98.40574	17474.790	2085.745
2009	284.4028	32.12917	1.0000000	32881	1.936422e+12	98.49197	16954.370	2025.815
2010	300.3987	29.96475	1.1500000	33588	1.995289e+12	100.00000	17245.840	2065.307
2011	300.1641	34.18300	1.3354000	34049	2.058369e+12	102.11160	17561.670	2110.593
2012	283.4498	32.55166	1.3958000	34120	2.088804e+12	104.10706	17591.768	2117.202
2013	292.0000	32.00000	1.3500000	34272	2.117189e+12	105.00625	17653.770	2129.404
2014	288.5000	32.20000	1.2850000	34407	2.149765e+12	105.53943	17703.040	2149.765
2015	281.5000	34.30000	1.1490000	34803	2.198432e+12	105.57902	17757.190	2173.690
2016	287.7000	32.60000	1.1100000	34777	2.234129e+12	105.77258	17809.400	2197.502
2017	307.7000	33.42000	1.2320000	34690	2.297242e+12	106.86445	18328.210	2247.856
2018	317.3000	31.98000	1.4370000	33626	2.363306e+12	108.84232	17270.020	2289.780
2019	322.3000	31.58400	1.4400000	32770	2.437635e+12	110.04857	16344.300	2331.980



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1767	790.8	2.235	0.03277
x1	-378013	120267	-3.143	0.003667
x2	4.242	1.423	2.98	0.005559
x3	36.89	5.615	6.57	2.449e-07

Table 3: Fitting linear model: y ~ x

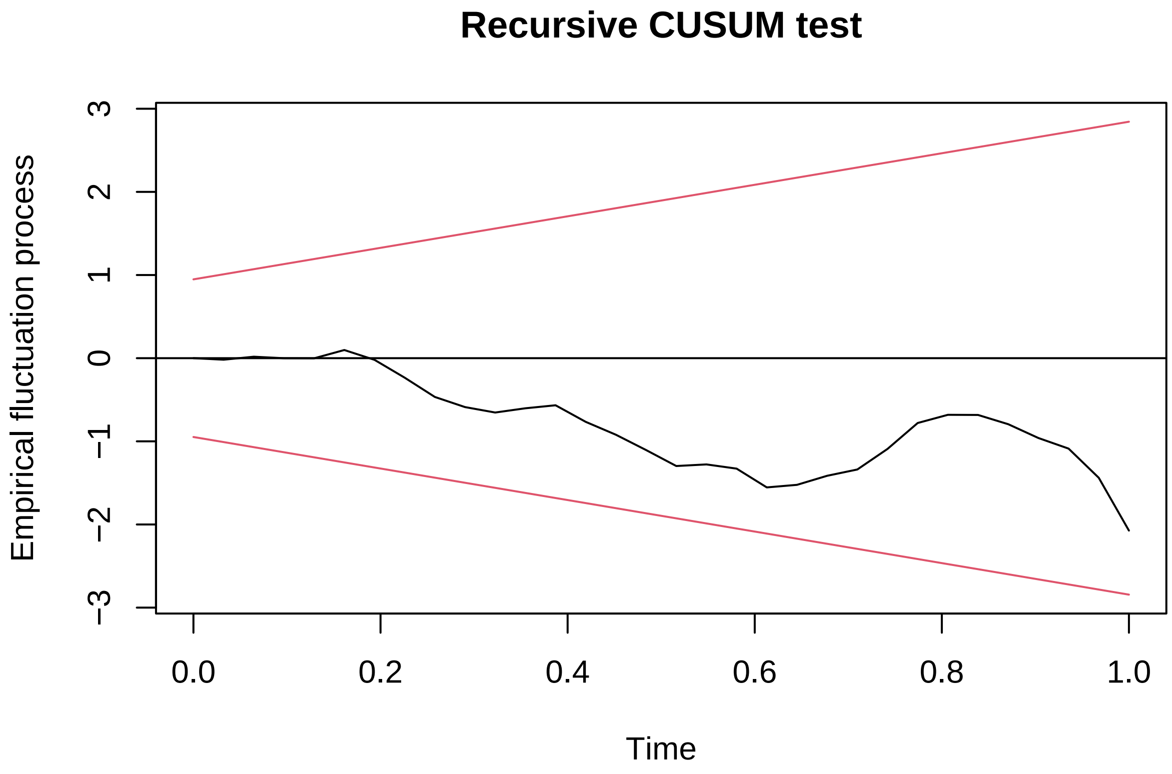
Observations	Residual Std. Error	R^2	Adjusted R^2
35	721.2	0.9475	0.9424

```
## function (x, format, digits = getOption("digits"), row.names = NA,
##     col.names = NA, align, caption = NULL, label = NULL, format.args = list(),
##     escape = TRUE, ...)
## {
##     format = kable_format(format)
##     if (!missing(align) && length(align) == 1L && !grepl("[^lcr]",
##         align))
##         align = strsplit(align, "")[[1]]
##     if (inherits(x, "list")) {
##         format = kable_format_latex(format)
##         res = lapply(x, kable, format = format, digits = digits,
##             row.names = row.names, col.names = col.names, align = align,
##             caption = NA, format.args = format.args, escape = escape,
##             ...)
##         return(kables(res, format, caption, label))
##     }
##     caption = kable_caption(label, caption, format)
##     if (!is.matrix(x))
##         x = as.data.frame(x)
##     if (identical(col.names, NA))
##         col.names = colnames(x)
##     m = ncol(x)
##     isn = if (is.matrix(x))
##         rep(is.numeric(x), m)
##     else sapply(x, is.numeric)
##     if (missing(align) || (format == "latex" && is.null(align)))
##         align = ifelse(isn, "r", "l")
##     digits = rep(digits, length.out = m)
##     for (j in seq_len(m)) {
##         if (is_numeric(x[, j]))
##             x[, j] = round(x[, j], digits[j])
##     }
##     if (any(isn)) {
##         if (is.matrix(x)) {
##             if (is.table(x) && length(dim(x)) == 2)
##                 class(x) = "matrix"
##             x = format_matrix(x, format.args)
##         }
##         else x[, isn] = format_args(x[, isn], format.args)
##     }
##     if (is.na(row.names))
##         row.names = has_rownames(x)
##     if (!is.null(align))
##         align = rep(align, length.out = m)
##     if (row.names) {
##         x = cbind(` ` = rownames(x), x)
##         if (!is.null(col.names))
##             col.names = c(" ", col.names)
##         if (!is.null(align))
##             align = c("l", align)
##     }
##     n = nrow(x)
##     x = replace_na(to_character(x), is.na(x))
##     if (!is.matrix(x))
##         x = matrix(x, nrow = n)
##     x = trimws(x)
##     colnames(x) = col.names
##     if (format != "latex" && length(align) && !all(align %in%
##         c("l", "r", "c")))
##         stop("'align' must be a character vector of possible values 'l', 'r', and 'c'")
##     attr(x, "align") = align
##     if (format == "simple" && nrow(x) == 0)
##         format = "pipe"
##     res = do.call(paste("kable", format, sep = "_"), list(x = x,
##         caption = caption, escape = escape, ...))
##     structure(res, format = format, class = "knitr_kable")
## }
## <bytecode: 0x000000001d05a130>
## <environment: namespace:knitr>
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
##           [,1]
## [1,] 0.5667082
```

```
Wr <- efp(y ~ x, type = "Rec-CUSUM")
plot(Wr)
```



```
#
# Test Cusum Square
#
rr <- (recresid(y ~ x))
rr <- rr^2
cumrr <- cumsum(rr)/scr
```

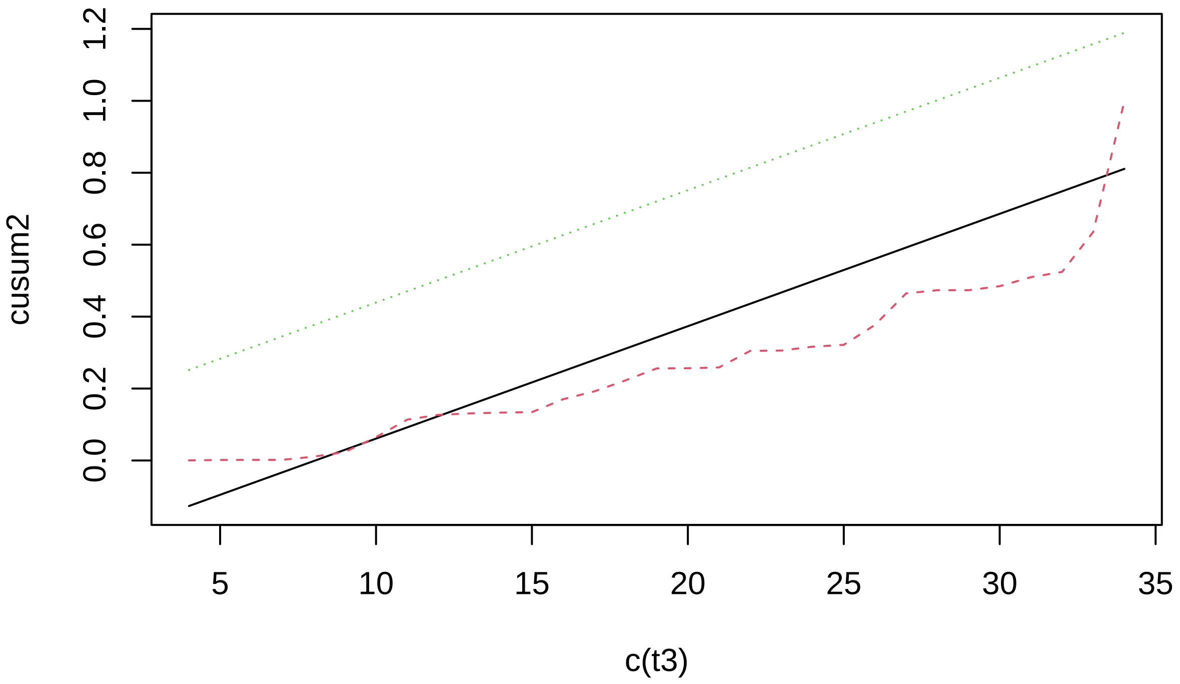
```
## Warning in cumsum(rr)/scr: Le recyclage d'un tableau (array) de longueur 1 dans un calcul arithmétiqu
tableau est obsolète.
##   Utilisez c() ou as.vector() à la place.
```

```
#
# Valeurs seuil de la distribution Cusum
#
c0 = 0.18915

kp2=K+1
c0 = 0.18915 # valeur critique de c0

t2 <- ts(kp2:n)
t3=t2-1
smin <-((t2-k)/(n-k))-c0
smax <- ((t2-k)/(n-k))+c0
#
```

```
vec2 <- c(smin, cumrr, smax)
cusum2 <- matrix(vec2, ncol = 3);
matplot(c(t3), cusum2, type ="l")
```



```
#sctest(y ~ x, type = "Chow", point =)
# pour R, la rupture est test?e non pas sur 1979-1980 mais sur 1979.
#Sur Eviews, la rupture est test?e sur 1980/
for(i in 9:30) {

print(sctest(y ~ x, type = "Chow", point = i) )
}
```

```
##
##  Chow test
##
## data:  y ~ x
## F = 4.6272, p-value = 0.005642
##
##
##  Chow test
##
## data:  y ~ x
## F = 4.6428, p-value = 0.005547
##
##
##  Chow test
```

```
##
## data:  y ~ x
## F = 4.6513, p-value = 0.005497
##
##
##  Chow test
##
## data:  y ~ x
## F = 5.796, p-value = 0.00168
##
##
##  Chow test
##
## data:  y ~ x
## F = 8.5115, p-value = 0.0001406
##
##
##  Chow test
##
## data:  y ~ x
## F = 10.764, p-value = 2.389e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 11.173, p-value = 1.772e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 11.398, p-value = 1.507e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 10.585, p-value = 2.728e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 10.003, p-value = 4.242e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 9.932, p-value = 4.48e-05
##
##
##  Chow test
```

```
##
## data:  y ~ x
## F = 8.8977, p-value = 0.0001021
##
##
##  Chow test
##
## data:  y ~ x
## F = 8.902, p-value = 0.0001017
##
##
##  Chow test
##
## data:  y ~ x
## F = 9.1596, p-value = 8.247e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 8.2241, p-value = 0.0001793
##
##
##  Chow test
##
## data:  y ~ x
## F = 8.8558, p-value = 0.0001056
##
##
##  Chow test
##
## data:  y ~ x
## F = 9.8364, p-value = 4.825e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 11.003, p-value = 2.004e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 9.5052, p-value = 6.256e-05
##
##
##  Chow test
##
## data:  y ~ x
## F = 7.0616, p-value = 0.0005013
##
##
##  Chow test
```

```
##
## data:  y ~ x
## F = 6.9945, p-value = 0.0005331
##
##
##  Chow test
##
## data:  y ~ x
## F = 7.5019, p-value = 0.0003366
```

```
n=length(Transport_France2019$Qdieselcamion)

P1 <- replicate(35, 0)
P1[1:16] <-  Transport_France2019$Pdiesel[1:16]/Transport_France2019$CPI[1:16]

P2 <- replicate(35, 0)
P2[17:35] <-  Transport_France2019$Pdiesel[17:35]/Transport_France2019$CPI[17:35]

vec <- c(P1,P2, Transport_France2019$"PIB en volume (en milliards d'euros 2014)",Transport_France2019$Q
X <- matrix( vec, ncol=4)
Y=matrix(Transport_France2019$Qdieselcamion,n,1)
q=ncol(Y);
k=ncol(X);
K=k+1

 y=Y
 x=X

 nobs=cbind(1:n)

 OLS=lm(formula = y ~ x)

 summary(OLS)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2393.64  -264.17    46.23   364.36  1468.79
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.174e+02  1.670e+03   0.070  0.94445
## x1          -2.946e+05  1.410e+05  -2.089  0.04528 *
## x2          -3.651e+05  1.203e+05  -3.034  0.00495 **
## x3           4.871e+00  1.525e+00   3.195  0.00328 **
## x4           3.735e+01  5.607e+00   6.661 2.23e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 718.3 on 30 degrees of freedom
## Multiple R-squared:  0.9496, Adjusted R-squared:  0.9429
## F-statistic: 141.4 on 4 and 30 DF,  p-value: < 2.2e-16
```

```
 names(OLS)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
xc = cbind(1,x)
bhat = OLS$coefficients
yf = xc %*% bhat
res = y - yf
scr = t(res) %*% res


d1 = t(res) %*% res
d2 =  t(res[2:n]-res[1:n-1]) %*% (res[2:n]-res[1:n-1])
dw = d2/d1
print (dw)
```

```
##           [,1]
## [1,] 0.5232029
```