*ECOM90004 – Time Series Analysis and Forecasting – Assignment 2*

*Joshua Copeland (SID:1444772)*

*Tutorial: Wednesday 12pm*

*No other group members*

*Note: I have included my R code as an appendix to this report (as specified in the Assignment 2 instructions) but it is extremely difficult to read. To make reading easier, I have also attached the html output from the R Markdown used to prepare my analysis after the appendix.*

**Question 1**

Given limited space for reporting results, answers to part (a) to (c) have been presented together.

***Table 1: Charts required for Q1(a) to Q1(c).***

|  |  |  |
| --- | --- | --- |
| (a) |  |  |
| (b) |  |  |
| (c) |  |  |

***Table 2: Reported coefficients for Q1(a) to Q1(c)***

|  |  |  |  |
| --- | --- | --- | --- |
| Coefficient | Linear model (a) | 1 break model (b) | 2 break model (c) |
|  | 11058.93 | 9858.717 | 9789.124 |
|  | 681.632 | 738.412 | 745.876 |
|  | N/A | 2630.154 | 2837.692 |
|  | N/A | –140.256 | –181.425 |
|  | N/A | N/A | –3609.528 |
|  | N/A | N/A | 151.267 |

* 1. Enforcing continuity at a breakpoint means the regression line before and after meets at that breakpoint. By incorporating structural break dummies, you are adding an extra intercept and slope to the model, therefore the only way to avoid this jump (maintain continuity) is to restrict their coefficients such that the fitted values make them disappear exactly at the breakpoints. This allows the fitted lines to pass smoothly through the breakpoints by restricting an intercept jump and only permitting slope changes.

Algebraically, the condition for the one break model (i.e. 1973) can be shown by starting with the one break model:

Before the break, the time interaction terms equal zero, giving:

By equating these two equations and solving for zero you arrive at the restriction required. We can only do this by setting :

Collecting like terms and dropping the innovation (as we’re working with fitted values) gives us:

Collecting like terms further, moving everything to one side and equate to zero (to enforce continuity):

The same process can be applied to get the conditions for 2008, the only difference is that this time the LHS is the one-break model rather linear time model. Giving us:

* 1. Applying the continuity restriction means the fitted line can only change in slope, not in intercept. Therefore, we get a GFC style trend break by forcing the fitted line to pass smoothly through the break date. In the break models we use here, impacts the level whereas impacts the slope. In the example we used above, we are effectively neutralising the impact of on the level in either model, which generalises to the GFC trend break for retail sales modelled earlier in the semester:
  2. The estimates of the jumps are:

We can test if their implementation is supported by using a Wald test to see if their values are statistically different from zero. The p-values of and are 0.000 and 0.063 respectively. Therefore, at the 5% significant level, there is only support for a structural break in 1973.

**Question 2**

1. Table X summarises the preferred AR lag orders . They all agree on modelling as an AR(3) process and are hence modelled as such. Importantly, all models chosen have large Ljung-Box test p-values, which does not give us any evidence of autocorrelation in .

***Table X: Summary of choice***

|  |  |  |
| --- | --- | --- |
| Model | choice | Ljung-Box p-value |
| Time trend | 3 | 0.220 |
| 1 break | 3 | 0.450 |
| 2 break | 3 | 0.351 |

1. Table X summarise the Augmented Dicky-Fuller (ADF) test statistics produced by each model. For each test I am using Model 1 (i.e. type = “none”) given is visually centred around zero in each model (as per the charts prepared for Question 1). It also contains information on how these test statistics compare to their relevant critical values.

***Table X: Summary of Augmented Dicky-Fuller (ADF) tests***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Model* | *Test statistic* | *Critical values* | | |
| *10%* | *5%* | *1%* |
| Linear | -2.54 | -1.62 | -1.95 | -2.58 |
| 1 break | -3.45 | -1.62 | -1.95 | -2.58 |
| 2 break | -3.55 | -1.62 | -1.95 | -2.58 |

1. Table X summarises the ADF critical values produced by the simulation process for each model, all using 1000 repetitions.

***Table X: Simulated critical values for ADF statistics***

|  |  |
| --- | --- |
| *Model* | *Value* |
| Linear | -1.90 |
| 1 break | -1.90 |
| 2 break | -1.90 |

1. Table X summarises the p-values implied by the ADF simulation process.

***Table X: Simulated p-values for ADF statistics***

|  |  |  |
| --- | --- | --- |
| *Model* | *ADF test statistic* | *Simulated p-value* |
| Linear | -2.54 | 0.01 |
| 1 break | -3.45 | 0.00 |
| 2 break | -3.55 | 0.00 |

1. Across all the models used for this exercise, the ADF tests show quite strong evidence against a unit root in . However, this evidence is weakest for the linear model, which is unable to reject the null hypothesis of a unit root at the 1% significance level. Both structural break models are able to reject the null hypothesis at the 10%, 5% and 1% significance levels. Using finite-sample simulations makes this evidence slightly stronger by virtue of it generating a critical value closer to zero (-1.90) relative to the conventional, asymptotic case (-1.95).
2. These results show us that the basic linear model provides the weakest evidence of stationarity in . By incorporating structural breaks into this time series, and using finite-sample simulations, we can be more certain of it’s stationarity. This exercise highlights the importance of these two things (structural break and finite-sampling distributions) when drawing conclusions from nuit root tests.

**R code appendix**

# Question 1

data <- read\_csv("usrealgdppercapita.csv") %>%

select(date = observation\_date, value = RealGDPPerCap) %>%

mutate(date = dmy(date)) %>%

na.omit()

# Q1a)

data <- data %>%

mutate(time = row\_number()\*0.25)

linear\_model <- lm(value ~ time, data = data)

summary(linear\_model)

linear\_model\_data <- tibble(

date = data$date,

value = data$value,

fitted = fitted(linear\_model),

residuals = resid(linear\_model)

)

c1 <- ggplot(linear\_model\_data, aes(x = date, y = value)) +

geom\_line() +

geom\_line(aes(y = fitted), color = "red", linewidth = 1) +

labs(

title = "Chart 1: Linear \n model fitted values",

y = "value",

x = "date"

)

c2 <- ggplot(linear\_model\_data, aes(x = date, y = residuals)) +

geom\_line() +

geom\_hline(yintercept = 0, linetype = "dashed") +

labs(

title = "Chart 2: Linear \n model residuals",

y = "residuals",

x = "date"

) +

theme(plot.title = element\_text(hjust = 0.5))

ggsave("chart1.png", c1, width = 3, height = 2.5, units = "in", dpi = 300)

ggsave("chart2.png", c2, width = 3, height = 2.5, units = "in", dpi = 300)

# Q1b)

data <- data %>%

mutate(

du\_1 = if\_else(lubridate::year(date) > 1973, 1, 0),

dt\_1 = if\_else(lubridate::year(date) > 1973, time, 0)

)

one\_break\_model <- lm(value ~ time + du\_1 + dt\_1, data = data)

summary(one\_break\_model)

one\_break\_model\_data <- tibble(

date = data$date,

value = data$value,

fitted = fitted(one\_break\_model),

residuals = resid(one\_break\_model)

)

c3 <- ggplot(one\_break\_model\_data, aes(x = date, y = value)) +

geom\_line() +

geom\_line(aes(y = fitted), color = "red", linewidth = 1) +

labs(

title = "Chart 3: One break model \n fitted values",

y = "y value",

x = "date"

) +

theme(plot.title = element\_text(hjust = 0.5))

*# Chart 4*

c4 <- ggplot(one\_break\_model\_data, aes(x = date, y = residuals)) +

geom\_line() +

geom\_hline(yintercept = 0, linetype = "dashed") +

labs(

title = "Chart 4: One break model \n residuals",

y = "residuals",

x = "date"

) +

theme(plot.title = element\_text(hjust = 0.5))

ggsave("chart3.png", c3, width = 3, height = 2.5, units = "in", dpi = 300)

ggsave("chart4.png", c4, width = 3, height = 2.5, units = "in", dpi = 300)

# c)

data <- data %>%

mutate(

du\_2 = if\_else(lubridate::year(date) > 2008, 1, 0),

dt\_2 = if\_else(lubridate::year(date) > 2008, time, 0)

)

two\_break\_model <- lm(value ~ time + du\_1 + dt\_1 + du\_2 + dt\_2, data = data)

summary(two\_break\_model)

two\_break\_model\_data <- tibble(

date = data$date,

value = data$value,

fitted = fitted(two\_break\_model),

residuals = resid(two\_break\_model)

)

*# Chart 5*

c5 <- ggplot(two\_break\_model\_data, aes(x = date, y = value)) +

geom\_line() +

geom\_line(aes(y = fitted), color = "red", linewidth = 1) +

labs(

title = "Chart 5: Two break model \n fitted values",

y = "value",

x = "date"

) +

theme(plot.title = element\_text(hjust = 0.5))

*# Chart 6*

c6 <- ggplot(two\_break\_model\_data, aes(x = date, y = residuals)) +

geom\_line() +

geom\_hline(yintercept = 0, linetype = "dashed") +

labs(

title = "Chart 6: Two break model \n residuals",

y = "residuals",

x = "date"

) +

theme(plot.title = element\_text(hjust = 0.5))

ggsave("chart5.png", c5, width = 3, height = 2.5, units = "in", dpi = 300)

ggsave("chart6.png", c6, width = 3, height = 2.5, units = "in", dpi = 300)

# d) iii)

*# We see if Perron's imposition of continuity is supported at 1973 and 2008 by applying a Wald test to see if the implied jump in the fitted value at each breakpoint is significantly different from zero.*

b <- coef(two\_break\_model)

V <- vcov(two\_break\_model)

*# Jump at 1973*

J73\_hat <- b["du\_1"] + 1973 \* b["dt\_1"]

se\_J73 <- sqrt(c(1,1973) %\*% V[c("du\_1","dt\_1"), c("du\_1","dt\_1")] %\*% c(1,1973))

t\_J73 <- J73\_hat / se\_J73

p\_J73 <- 2 \* (1 - pnorm(abs(t\_J73))) *# two-sided p-value*

*# Jump at 2008*

J08\_hat <- b["du\_2"] + 2008 \* b["dt\_2"]

se\_J08 <- sqrt(c(1,2008) %\*% V[c("du\_2","dt\_2"), c("du\_2","dt\_2")] %\*% c(1,2008))

t\_J08 <- J08\_hat / se\_J08

p\_J08 <- 2 \* (1 - pnorm(abs(t\_J08)))

*# Print results*

print(p\_J73)

print(p\_J08)

# Question 2

# Q2a

select\_ar\_summary <- **function**(formula, data, p\_max = 8) {

y <- data$value

X <- model.matrix(formula, data = data)

n <- length(y)

*# Ljung–Box lag: min(8, n/5)*

lb\_lag <- min(8, floor(n/5))

*# fit AR(p) with exogenous regressors for p = 0..p\_max*

grid <- map\_dfr(0:p\_max, **function**(p) {

fit <- Arima(

y, order = c(p, 0, 0),

xreg = X,

include.mean = FALSE,

method = "ML"

)

tibble(

p = p,

AIC = fit$aic,

BIC = fit$bic,

AICc = fit$aicc,

fit = list(fit)

)

})

*# pick best lag by AICc*

best <- grid %>% slice\_min(AICc, with\_ties = FALSE)

best\_p <- best$p

best\_f <- best$fit[[1]]

*# Ljung–Box test*

lb <- Box.test(residuals(best\_f), type = "Ljung-Box",

lag = lb\_lag, fitdf = best\_p)

tibble(

model = deparse(formula),

best\_p = best\_p,

lb\_pval = unname(lb$p.value)

)

}

*# Example usage*

linear <- value ~ time

one\_break <- value ~ time + du\_1 + dt\_1

two\_break <- value ~ time + du\_1 + dt\_1 + du\_2 + dt\_2

ar\_summary <- bind\_rows(

select\_ar\_summary(linear, data),

select\_ar\_summary(one\_break, data),

select\_ar\_summary(two\_break, data)

) %>%

mutate(model = c("linear", "one break", "two breaks"))

print(ar\_summary)

# Q2b

*# 1. Collect residuals from the three models*

Z\_data <- tibble(

date = data$date,

linear = residuals(linear\_model),

one\_break = residuals(one\_break\_model),

two\_break = residuals(two\_break\_model)

) %>%

pivot\_longer(-date, names\_to = "model", values\_to = "residual")

*# 3. ADF test function with lag selection by AIC, max 5*

get\_adf <- **function**(z, model\_name, max\_lags = 5) {

test <- ur.df(z, type = "none", lags = max\_lags, selectlags = "AIC")

tau <- test@teststat[1]

cv <- test@cval[1, ]

tibble(

model = model\_name,

best\_lag\_AIC = test@lags, *# reports chosen lag order*

tau\_stat = tau,

cv\_1pct = cv["1pct"],

cv\_5pct = cv["5pct"],

cv\_10pct = cv["10pct"]

)

}

*# 4. Apply to all three models*

adf\_table <- bind\_rows(

get\_adf(residuals(linear\_model), "linear"),

get\_adf(residuals(one\_break\_model), "one break"),

get\_adf(residuals(two\_break\_model), "two breaks")

)

print(adf\_table)

#Q2c

*# Collect residuals*

Z <- list(

linear = residuals(linear\_model),

one\_break = residuals(one\_break\_model),

two\_break = residuals(two\_break\_model)

)

*# Function: simulate null distribution of ADF tau (type="none")*

simulate\_adf\_null <- **function**(n, reps = 1000, max\_lags = 5, seed = 123) {

set.seed(seed)

taus <- numeric(reps)

**for** (r **in** seq\_len(reps)) {

*# Generate unit root process under H0*

e <- rnorm(n)

z <- cumsum(e)

*# ADF test with lag selection by AIC (0..max\_lags)*

test <- ur.df(z, type = "none", lags = max\_lags, selectlags = "AIC")

taus[r] <- test@teststat[1]

}

taus

}

*# Simulation per model*

reps <- 1000

max\_lags <- 5

cv\_table <- tibble(

model = names(Z),

n = map\_int(Z, length),

tau\_actual = map\_dbl(Z, ~ {

test <- ur.df(.x, type = "none", lags = max\_lags, selectlags = "AIC")

test@teststat[1]

}),

cv5\_simulated = map2\_dbl(map\_int(Z, length), names(Z), ~ {

taus <- simulate\_adf\_null(.x, reps = reps, max\_lags = max\_lags)

quantile(taus, probs = 0.05, na.rm = TRUE)

})

) %>%

mutate(reject\_5pct = tau\_actual < cv5\_simulated)

print(cv\_table)

# Q2d

*# Function: compute observed ADF tau for a given series*

adf\_tau\_actual <- **function**(series, max\_lags) {

test <- ur.df(series, type = "none", lags = max\_lags, selectlags = "AIC")

unname(test@teststat[1])

}

*# Function: simulate null distribution of ADF tau under unit root*

simulate\_adf\_null <- **function**(n, reps, max\_lags, seed = 123) {

set.seed(seed + n)

taus <- numeric(reps)

**for** (r **in** seq\_len(reps)) {

z <- cumsum(rnorm(n)) *# random walk under H0*

test <- ur.df(z, type = "none", lags = max\_lags, selectlags = "AIC")

taus[r] <- unname(test@teststat[1])

}

taus

}

*# Function: empirical left-tail p-value*

emp\_pval <- **function**(tau\_obs, taus\_null) {

(sum(taus\_null <= tau\_obs) + 1) / (length(taus\_null) + 1)

}

*# Compute observed tau and simulated p-value for each model*

pval\_tbl <- tibble(

model = c("linear", "one break", "two breaks"),

resid = list(

residuals(linear\_model),

residuals(one\_break\_model),

residuals(two\_break\_model)

)

) %>%

mutate(

n = map\_int(resid, length),

tau\_obs = map\_dbl(resid, ~ adf\_tau\_actual(.x, max\_lags)),

taus\_sim = map(n, ~ simulate\_adf\_null(.x, reps = reps, max\_lags = max\_lags)),

p\_value = map2\_dbl(tau\_obs, taus\_sim, emp\_pval)

) %>%

select(model, n, tau\_obs, p\_value)

print(pval\_tbl)