Tuto week 4

Rodolphe

2026-01-28

Contents

Setup . . . 2

Exercice 1 **3**

Pulling data . . . 3

Exercice 2 **6**

Pulling data needed . . . 6

Using expanding window . . . 7

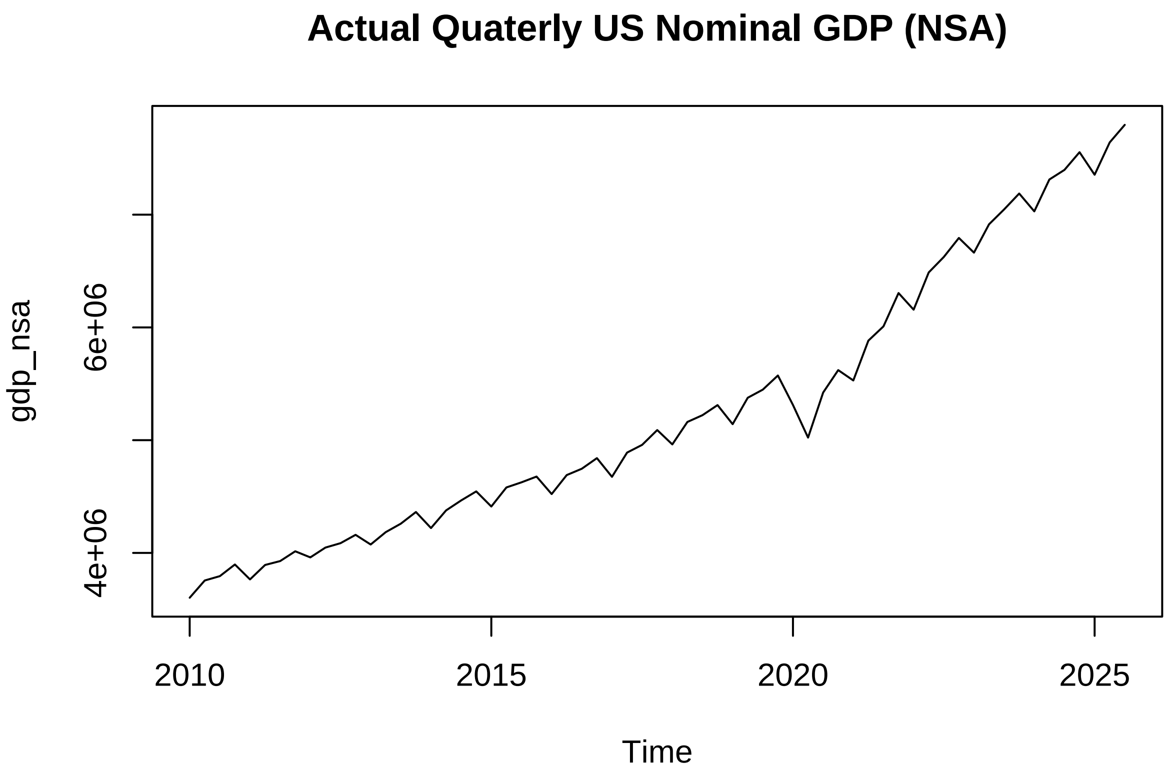
Setup

```
packages <- c(
  "quantmod",   # download time-series
  "tidyverse",  # data manipulation and visualization
  "stargazer",  # publication-ready tables
  "conflicted", # management of function name conflicts across packages
  "moments",    # statistical moments, like skewness and kurtosis
  "lubridate",  # manipulation of data
  "knitr",      # integrate R code with text (e.g. LaTeX, HTML, Markdown)
  "sn",         # simulation of skewed distributions
  "patchwork",  # composition of graphs
  "latex2exp",  # latex
  "fredr",
  "forecast",
  "dplyr",
  "tidyquant",
  "timetk",
  "lmtest",
  "sandwich"
)

# --- from the ones needed extract the ones not installed
to_install <- packages[!packages %in% installed.packages()[, "Package"]]

# --- install the packages
if (length(to_install) > 0) {
  install.packages(to_install)
}

# --- load the packages in our session
invisible(lapply(packages, library, character.only = TRUE))

# --- defining `lag()` from the dplyr package as the preference
conflict_prefer("lag", "dplyr")
conflict_prefer("filter", "dplyr")
conflicts_prefer(PerformanceAnalytics::legend)
```

Exercice 1

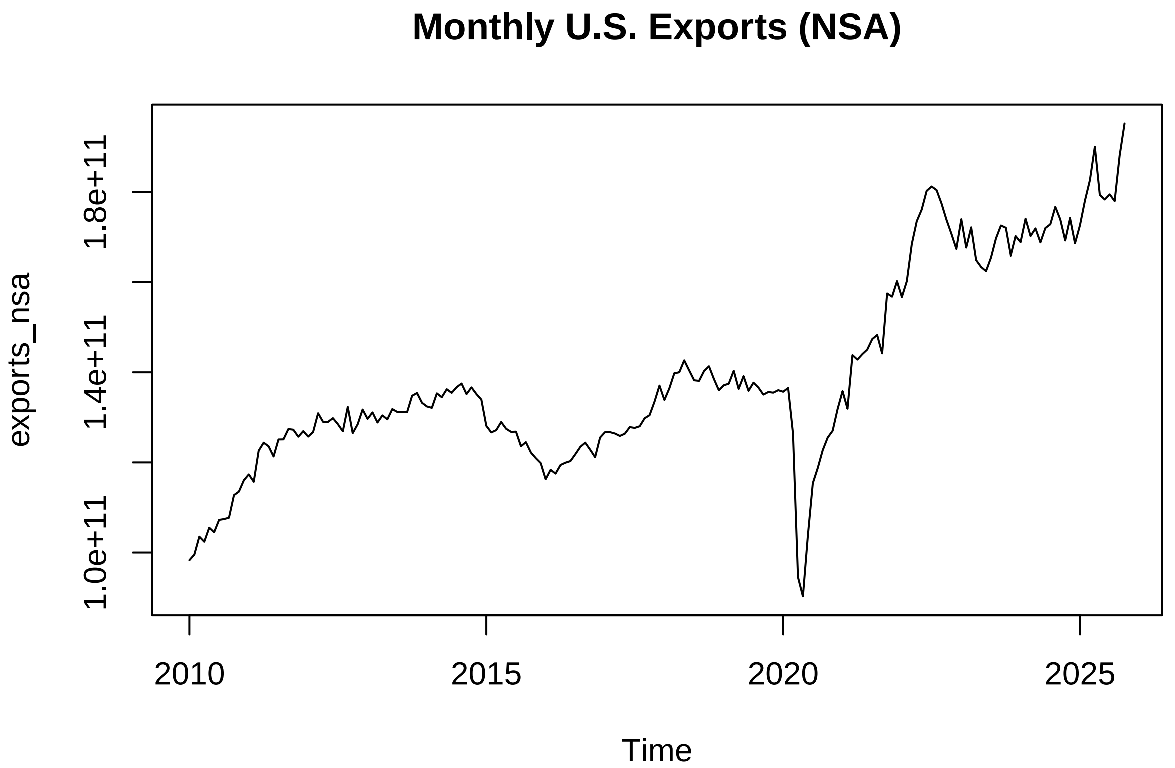
Pulling data

```r
# Install and load
#install.packages("fredr")
#library(fredr)

# Set your API key
fredr_set_key("f36608b6a8dbafaf4b377a110e49a4f3")

# Fetch Nominal GDP - Not Seasonally Adjusted (NSA)
# Series ID 'NA000334Q' is US Nominal GDP, Quarterly, NSA
gdp_data <- fredr(
  series_id = "NA000334Q",
  observation_start = as.Date("2010-01-01")
);

# Convert to a 'ts' (Time Series) object for your exercise
gdp_nsa <- ts(gdp_data$value, frequency = 4, start = c(2010, 1));

plot(gdp_nsa, main="Actual Quaterly US Nominal GDP (NSA)")
```




```r
fredr_set_key("f36608b6a8dbafaf4b377a110e49a4f3")
exports_data <- fredr(
  series_id = "XTEXVA01USM664S",
  observation_start = as.Date("2010-01-01")
)

# Convert to time series object
```

```r
exports_nsa <- ts(exports_data$value, frequency = 12, start = c(2010, 1))
plot(exports_nsa, main="Monthly U.S. Exports (NSA)")
```



Try to extract seasonality

Get **t** as list of timeseries object dummies is a binary matrix 11 columns (Jan to Nov). 1 if it correspond to the month 0 if not. We don't want December so that we can estimate the intercept without having an perfect equation $Jan + Feb + .. + Nov + Dec = 1$. Indeed, the sum of the month is always equal to 1 and the intercept equal 1 as well, it is therefore impossible to compute the effect of the intercept ==> mathematically, the matrix is singular and therefore cannot be inverted.

```
time_trend <- time(exports_nsa)
dummies <- seasonaldummy(exports_nsa)
```

Then the model is estimated. And we compute the seasonal effect from Jan to Nov. model is constructed such that: - 1. Intercept - 2. t : the trend the model found - 3. Jan - 4. Fev … - 13. Nov

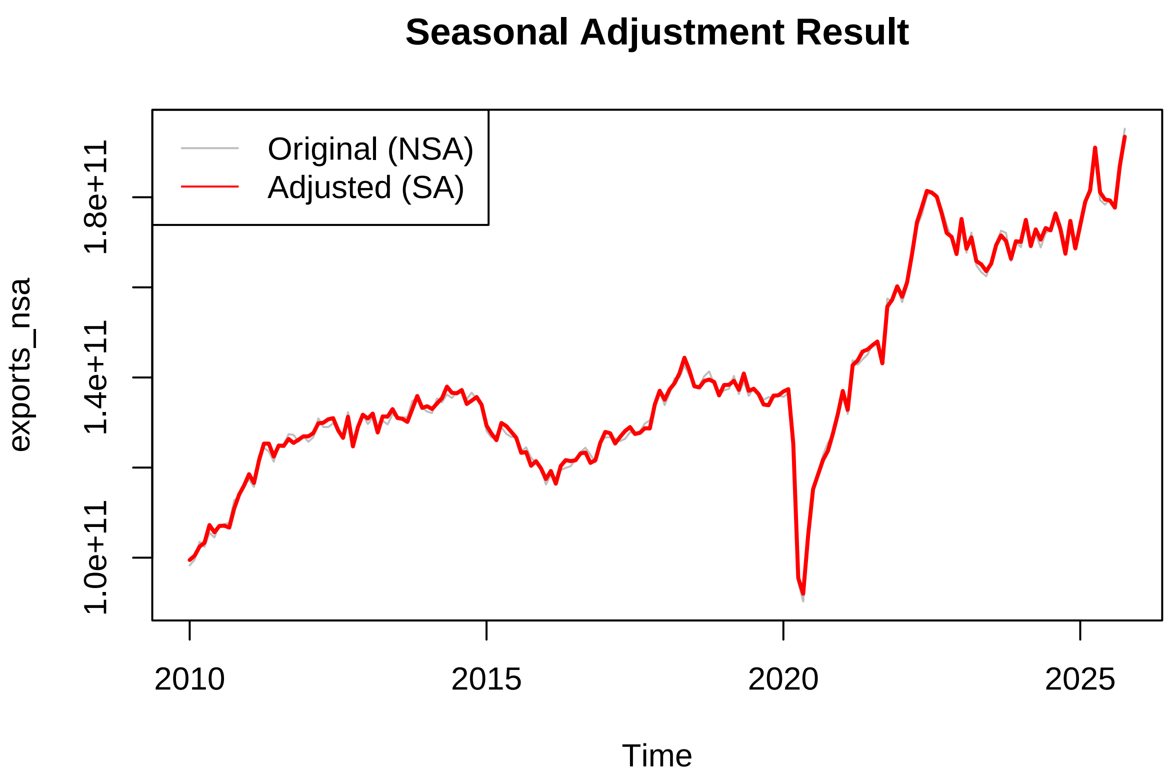
As mentionned earlier, the model should not take december in account

```
model <- lm(exports_nsa ~ time_trend + dummies)
seasonal_effects <- dummies %*% coef(model)[3:13]
```

Here we remove the seasonal component from the serie

```
exports_sa <- exports_nsa - seasonal_effects
```

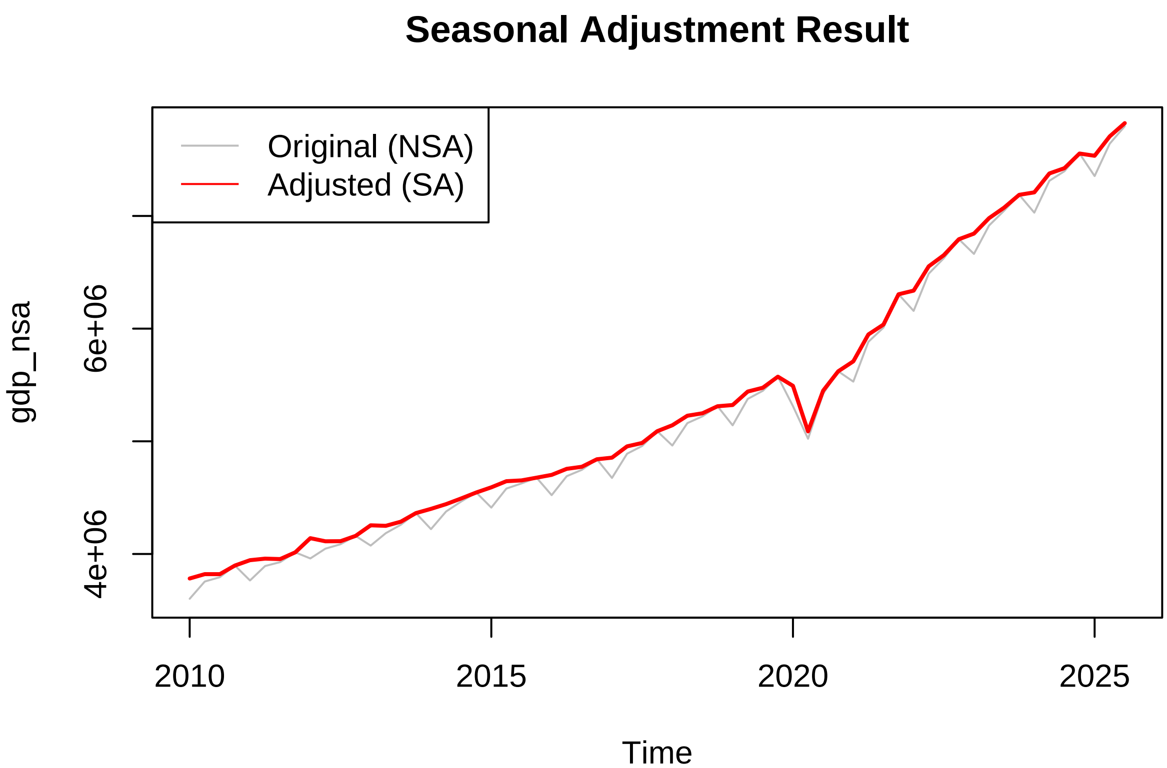
```
plot(exports_nsa, col="gray", main="Seasonal Adjustment Result")
lines(exports_sa, col="red", lwd=2)
legend("topleft", legend=c("Original (NSA)", "Adjusted (SA)"), col=c("gray", "red"), lty=1)
```




```
# 1. Create variables
time_trend2 <- time(gdp_nsa)
month_dummies2 <- seasonaldummy(gdp_nsa)

# 2. Fit the Regression Model
# This captures the trend and the monthly 'shocks'
```

```r
fit2 <- lm(gdp_nsa ~ time_trend2 + month_dummies2)

# 3. Extract the Seasonally Adjusted (SA) Series
# We remove the effect of the dummies, but keep the trend and intercept
seasonal_component2 <- month_dummies2 %*% coef(fit2)[3:5]
gdp_sa <- gdp_nsa - seasonal_component2
```

```r
plot(gdp_nsa, col="gray", main="Seasonal Adjustment Result")
lines(gdp_sa, col="red", lwd=2)
legend("topleft", legend=c("Original (NSA)", "Adjusted (SA)"), col=c("gray", "red"), lty=1)
```



On GDP, we can see that is can be seasonally adjusted, the red plot is flattened

Exercice 2

Pulling data needed

We'll use ajdusted column so that it takes dividends into account (there is no drop bc of divs and stock splits)

```r
data_raw <- tq_get("SPY",
                   get = "stock.prices",
                   from = "1960-01-01")

# 2. Pull Dividends specifically
div_raw <- tq_get("SPY",
                  get = "dividends",
                  from = "1960-01-01")
```

Contruct monthly data

```
monthly_prices <- data_raw %>%
  tq_transmute(select = adjusted, mutate_fun = to.monthly, indexAt = "firstof")

monthly_divs <- div_raw %>%
  tq_transmute(select = value, mutate_fun = to.monthly, indexAt = "firstof")
```

join the two tables based on time index. Is it a norm to use **12 months** to avoid seasonality in div payments?

```
#df <- left_join(monthly_prices, monthly_divs, by = "date") %>%
#  mutate(value = ifelse(is.na(value), 0, value),
#         l_price = log(adjusted),
#         ann_div = slidify_vec(value, .f = sum, .period = 12, .align = "right"),
#         l_div_price = log(ann_div / adjusted))
df <- left_join(monthly_prices, monthly_divs, by = "date") %>%
  mutate(
    value = ifelse(is.na(value), 0, value),
    ann_div = slidify_vec(value, .f = sum, .period = 12, .align = "right"),
    l_div_price = log(ann_div / adjusted), #dpt

    ret_next = lead(log(adjusted + value)) - log(adjusted) #r_t+1
  ) %>%
  filter(!is.na(l_div_price), !is.na(ret_next), !is.infinite(l_div_price))

head(df)
```

```
## # A tibble: 6 x 6
##   date       adjusted value ann_div l_div_price ret_next
##   <date>        <dbl> <dbl>   <dbl>       <dbl>    <dbl>
## 1 1993-12-01     26.4 0.317   0.53        -3.91   0.0343
## 2 1994-01-01     27.3 0       0.53        -3.94  -0.0296
## 3 1994-02-01     26.5 0       0.53        -3.91  -0.0322
## 4 1994-03-01     25.4 0.271   0.588       -3.76   0.0111
## 5 1994-04-01     25.7 0       0.588       -3.78   0.0158
## 6 1994-05-01     26.1 0       0.588       -3.79  -0.0232
```

Using expanding window

```
T <- nrow(df)
m <- floor(T * 0.5)

f_benchmark <- numeric(T - m)
f_model     <- numeric(T - m)
actual_ret  <- numeric(T - m)
```

```
for (i in 1:(T - m)) {
  current_t <- m + i - 1
  train_data <- df[1:current_t, ]

  f_benchmark[i] <- mean(train_data$ret_next, na.rm = TRUE) #mean is forecast bc E(\epsilon) = 0

  model_fit <- lm(ret_next ~ l_div_price, data = train_data) #unrestricted: r_{t+1} = a + b*dp_t

  last_dp <- df$l_div_price[current_t]
  f_model[i] <- coef(model_fit)[1] + coef(model_fit)[2] * last_dp
```

```r
  actual_ret[i] <- df$ret_next[current_t]
}
```

Finally we compute $R^2_{oos} = 1 - \frac{MSFE_{model}}{MSFE_{benchmark}}$

```r
e_bench <- actual_ret - f_benchmark
e_model <- actual_ret - f_model

msfe_bench <- mean(e_bench^2)
msfe_model <- mean(e_model^2)

# Calculate R2_OOS
r2_oos <- 1 - (msfe_model / msfe_bench)
cat("Out-of-Sample R-squared:", round(r2_oos, 5), "\n")
```

```
## Out-of-Sample R-squared: -0.00319
```

the model works slightly worst than random walk

```r
f_t <- e_bench^2 - (e_model^2 - (f_benchmark - f_model)^2)
cw_model <- lm(f_t ~ 1)
cw_results <- coeftest(cw_model, vcov = NeweyWest(cw_model))

print(cw_results)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.4536e-05 2.5731e-05  0.5649   0.5728
```

No clue how to interprete though