Session 8

Intro to Time Series

Anthony Tay

This Version: 04 Oct 2025

Time Series in R

We will use the following packages

```
library(tidyverse)  # for data management
library(patchwork)  # for plotting
library(fpp3)       # a suite of packages for forecasting
library(ggfortify)  # for plotting
library(readxl)     # used to read in excel files
library(mFilter)    # require for HP filter
```

Time Series in R

We will use the following datasets:

```
ts01 <- read_excel("data\\ts_01.xlsx")
ts01 %>% head(3)
```

```
# A tibble: 3 x 7
  DATE                ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG POULTRY_US
  <dttm>                    <dbl>   <dbl> <dbl>  <dbl>     <dbl>      <dbl>
1 1983-01-01 00:00:00        667.  232164  14.3   97.8        NA     340252
2 1983-02-01 00:00:00        587.  212591  11.4   97.9        NA     313680
3 1983-03-01 00:00:00        727.  242272  14.5   97.9        NA     368250
```

```
ts01 %>% tail(3)
```

```
# A tibble: 3 x 7
  DATE                ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG POULTRY_US
  <dttm>                    <dbl>   <dbl> <dbl>  <dbl>     <dbl>      <dbl>
1 2017-10-01 00:00:00       4504. 1402299  118.   247.     3565.     783000
2 2017-11-01 00:00:00       4283. 1397330  115.   247.     3582.     728307
3 2017-12-01 00:00:00       4376. 1569616  120.   247.     3682.     704485
```

Time Series in R

```
ts02 <- read_csv("data\\ts_02.csv")
ts02 %>% head(4)
```

```
# A tibble: 4 x 3
  Period      Y    Y1
  <chr>   <dbl> <dbl>
1 1980Q1   2.67  2.67
2 1980Q2   3.41  3.41
3 1980Q3   3.49  3.49
4 1980Q4   3.84  3.84
```

```
ts02 %>% tail(4)
```

```
# A tibble: 4 x 3
  Period      Y    Y1
  <chr>   <dbl> <dbl>
1 2011Q1   3.49  1.81
2 2011Q2   4.15  4.42
3 2011Q3   5.09  8.18
4 2011Q4   4.47  5.69
```

Both datasets currently stored as `tibble` objects

In R, there are several data structures for working with time series, including

- `ts` objects
- tsibble (from fpp3)

Some time series-related functions in R require time series data to be stored as `ts` objects, others require `tsibble` objects

Time Series in R (Converting tibble into tsibble)

```
ts01 <- ts01 %>% mutate(DATE=yearmonth(DATE)) %>% as_tsibble(index=DATE)
glimpse(ts01)
```

```
Rows: 420
Columns: 7
$ DATE        <mth> 1983 Jan, 1983 Feb, 1983 Mar, 1983 Apr, 1983 May, 1983 Jun, 1983 Jul, 1983 Aug, 1983
$ ELEC_GEN_SG <dbl> 667.3, 586.9, 727.4, 719.0, 727.1, 728.4, 741.1, 764.2, 730.7, 771.0, 730.4, 732.4, 7
$ TOUR_SG     <dbl> 232164, 212591, 242272, 226610, 236720, 227843, 249891, 263236, 230076, 240012, 23722
$ IP_SG       <dbl> 14.34, 11.37, 14.50, 12.86, 13.01, 12.62, 13.56, 14.89, 14.74, 15.52, 13.80, 15.60, 1
$ CPI_US      <dbl> 97.8, 97.9, 97.9, 98.6, 99.2, 99.5, 99.9, 100.2, 100.7, 101.0, 101.2, 101.3, 101.9, 1
$ DOMEX5_SG   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
$ POULTRY_US  <dbl> 340252, 313680, 368250, 345740, 364275, 373732, 332128, 380313, 349947, 341530, 3097
```

```
ts01 %>% select(DATE, DOMEX5_SG) %>% filter_index("1996M7" ~ "1998M6") %>% glimpse()
```

```
Rows: 24
Columns: 2
$ DATE      <mth> 1996 Jul, 1996 Aug, 1996 Sept, 1996 Oct, 1996 Nov, 1996 Dec, 1997 Jan, 1997 Feb, 1997 M
$ DOMEX5_SG <dbl> NA, NA, NA, NA, NA, NA, 3842.243, 2883.241, 4147.861, 4121.983, 3893.149, 3879.160, 390
```

Time Series in R (Converting tibble into ts)

```
ts02 <- ts02 %>% select(-Period) %>% ts(start = c(1980,1), freq=4)
```

```
ts02[, "Y"] %>% window(end=c(1992,2)) %>%
  print()
```

```
         Qtr1     Qtr2     Qtr3     Qtr4
1980 2.665478 3.406533 3.486567 3.842664
1981 3.627703 3.354770 3.829209 3.015743
1982 2.840764 2.745151 2.294659 1.941930
1983 1.843535 1.542656 1.532376 1.945526
1984 1.781760 2.240837 3.160621 2.515336
1985 3.297786 3.508998 3.898615 3.453972
1986 4.091660 4.626595 4.893875 3.836605
1987 3.418058 3.369342 3.622167 3.993465
1988 3.449985 4.280609 3.174542 2.900369
1989 2.556112 2.461438 2.328120 1.711213
1990 1.963203 2.515729 2.164962 2.881831
1991 2.564570 2.962895 3.628741 4.738045
1992 4.600449 4.117222
```

```
ts02 %>% window(start=c(1990,3), end=c(1991,4)) %>%
  print()
```

```
               Y       Y1
1990 Q3 2.164962 2.164962
1990 Q4 2.881831 2.881831
1991 Q1 2.564570 2.564570
1991 Q2 2.962895 2.962895
1991 Q3 3.628741 3.628741
1991 Q4 4.738045 4.738045
```

Common Data Transformations

Taking lags

```r
ts02a <- cbind(Y = ts02[,"Y"],
               Y_1 = stats::lag(ts02[,"Y"], -1),    ## This is `stats` package's lag function
               Y_2 = stats::lag(ts02[,"Y"], -2),
               Y_3 = stats::lag(ts02[,"Y"], -3),
               Y_4 = stats::lag(ts02[,"Y"], -4))
```

```r
ts02a %>% window(end=c(1981,4)) %>% round(3)
```

```
            Y   Y_1   Y_2   Y_3   Y_4
1980 Q1 2.665    NA    NA    NA    NA
1980 Q2 3.407 2.665    NA    NA    NA
1980 Q3 3.487 3.407 2.665    NA    NA
1980 Q4 3.843 3.487 3.407 2.665    NA
1981 Q1 3.628 3.843 3.487 3.407 2.665
1981 Q2 3.355 3.628 3.843 3.487 3.407
1981 Q3 3.829 3.355 3.628 3.843 3.487
1981 Q4 3.016 3.829 3.355 3.628 3.843
```

```r
ts02a %>% window(start=c(2011,1)) %>% round(3)
```

```
            Y   Y_1   Y_2   Y_3   Y_4
2011 Q1 3.495 3.837 3.937 4.520 4.892
2011 Q2 4.149 3.495 3.837 3.937 4.520
2011 Q3 5.087 4.149 3.495 3.837 3.937
2011 Q4 4.466 5.087 4.149 3.495 3.837
2012 Q1    NA 4.466 5.087 4.149 3.495
2012 Q2    NA    NA 4.466 5.087 4.149
2012 Q3    NA    NA    NA 4.466 5.087
2012 Q4    NA    NA    NA    NA 4.466
```

Common Data Transformations

Taking logs, taking differences

```
ts01 <- ts01 %>%
  mutate(LN_IP_SG = log(IP_SG),
         LN_IP_SG_1 = lag(LN_IP_SG, 1),  ## This is tidyverse's lag function
         D_LN_IP_SG = LN_IP_SG - LN_IP_SG_1)
```

```
ts01 %>%
  select(
    DATE, IP_SG, LN_IP_SG, LN_IP_SG_1, D_LN_IP_SG
  ) %>% head(5)
```

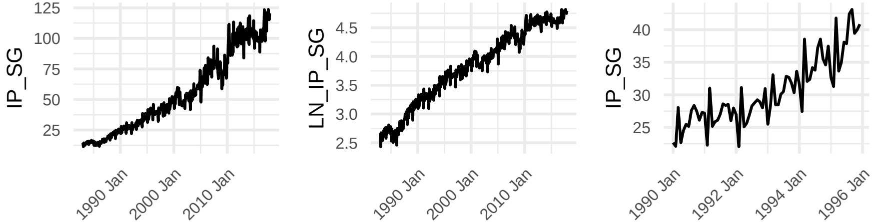
```
# A tsibble: 5 x 5 [1M]
      DATE IP_SG LN_IP_SG LN_IP_SG_1 D_LN_IP_SG
     <mth> <dbl>    <dbl>      <dbl>      <dbl>
1 1983 Jan  14.3     2.66      NA         NA
2 1983 Feb  11.4     2.43       2.66      -0.232
3 1983 Mar  14.5     2.67       2.43       0.243
4 1983 Apr  12.9     2.55       2.67      -0.120
5 1983 May  13.0     2.57       2.55       0.0116
```

```
ts01 %>%
  select(
    DATE, IP_SG, LN_IP_SG, LN_IP_SG_1, D_LN_IP_SG
  ) %>% tail(5)
```

```
# A tsibble: 5 x 5 [1M]
       DATE IP_SG LN_IP_SG LN_IP_SG_1 D_LN_IP_SG
      <mth> <dbl>    <dbl>      <dbl>      <dbl>
1  2017 Aug  119.     4.78       4.79   -0.00970
2 2017 Sept  124.     4.82       4.78    0.0380
3  2017 Oct  118.     4.77       4.82   -0.0483
4  2017 Nov  115.     4.75       4.77   -0.0226
5  2017 Dec  120.     4.79       4.75    0.0451
```

Characteristics of Economic Time Series Data

```
theme0 <- theme_minimal() +
  theme(text=element_text(size=8), axis.text.x = element_text(angle=45, hjust=1))
p1 <- ts01 %>% autoplot(IP_SG) + xlab("") + theme0
p2 <- ts01 %>% autoplot(LN_IP_SG) + xlab("") + theme0
p3 <- ts01 %>% filter_index("1990M1" ~ "1995M12") %>% autoplot(IP_SG) + xlab("") + theme0
p1 | p2 | p3
```

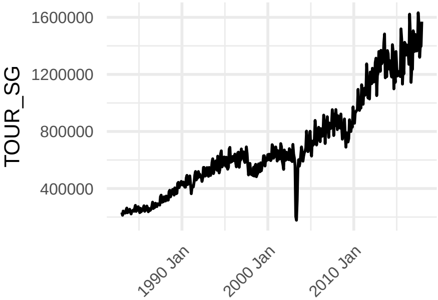


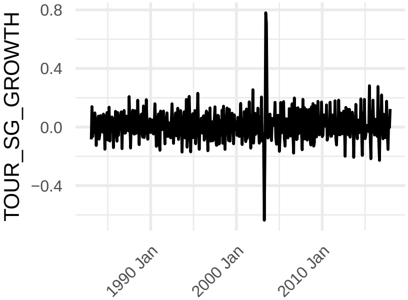
Trend, Cycles, Seasonality and other features (e.g., increasing variance)

Characteristics of Economic Time Series Data

Occasional “special events”

```
ts01 <- ts01 %>% mutate(TOUR_SG_GROWTH = (TOUR_SG - lag(TOUR_SG))/lag(TOUR_SG))
p1 <- ts01 %>% autoplot(TOUR_SG) + xlab("") + theme0
p2 <- ts01 %>% autoplot(TOUR_SG_GROWTH) + xlab("") + theme0
p1 | p2
```





Characteristics of Economic Time Series

Aside: $\ln Y_t - \ln Y_{t-1}$ can be interpreted as

- approximate per period growth rate

$$\ln Y_{t+1} - \ln Y_t \approx \frac{Y_{t+1} - Y_t}{Y_t}$$

- continuous growth rate: if

$$Y_t = Y_0 e^{rt} \text{ so that } \frac{dY_t/dt}{Y_t} = \frac{rY_0 e^{rt}}{Y_0 e^{rt}} = r$$

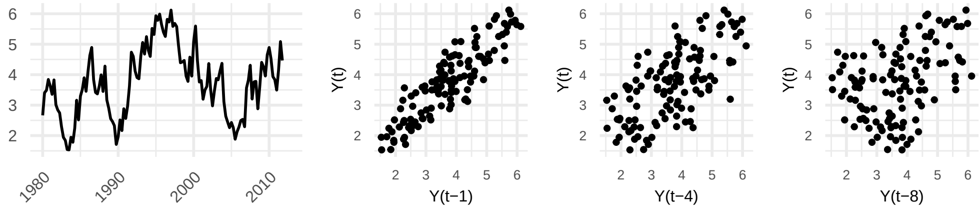
then

$$\ln Y_{t+1} - \ln Y_t = \ln Y_0 + r(t+1) - \ln Y_0 + rt = r\,.$$

Characteristics of Economic Time Series

Intertemporal correlations

```
theme1 <- theme_minimal() + theme(text=element_text(size=6), aspect.ratio=1)
p1 <- autoplot(ts02a[,"Y"]) + theme(aspect.ratio = 0.7) + xlab("") + theme0
p2 <- ts02a %>% ggplot(aes(x=Y_1, y=Y)) + geom_point(size=0.6) + ylab("Y(t)") + xlab("Y(t-1)") + theme1
p3 <- ts02a %>% ggplot(aes(x=Y_4, y=Y)) + geom_point(size=0.6) + ylab("Y(t)") + xlab("Y(t-4)") + theme1
p4 <- ts02a %>% ggplot(aes(x=lag(Y,8), y=Y)) + geom_point(size=0.6) + ylab("Y(t)") +
      xlab("Y(t-8)") + theme1
p1 | p2 | p3 | p4
```



Characteristics of Economic Time Series

Measuring intertemporal correlation

Sample Autocovariance Function:

$$\widehat{\gamma}_k = \frac{1}{T}\sum_{t=k+1}^{T}(Y_t - \overline{Y})(Y_{t-k} - \overline{Y})\,,\; k = 0, 1, 2, \ldots$$

- Compare with sample covariance of $\{X_i, Y_i\}_{i=1}^{n}$

$$Smpl.Cov.(X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})$$

- $\widehat{\gamma}_0$ is the sample variance (biased version)

Characteristics of Economic Time Series

Recall sample correlation:

$$Smpl.Corr.(X_i, Y_i) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

Corresponding formula for autocorrelation at lag k might be something like

$$Smpl.Corr.(Y_t, Y_{t-k}) = \frac{\sum_{t=k+1}^{T}(Y_t - \overline{Y})(Y_{t-k} - \overline{Y})}{\sqrt{\sum_{t=k+1}^{T}(Y_t - \overline{Y})^2}\sqrt{\sum_{t=k+1}^{T}(Y_{t-k} - \overline{Y})^2}}$$

But for large T and small k, denominator terms are close to each other (lots of overlapping terms), therefore...

Characteristics of Economic Time Series

Use the following definition

Sample Autocorrelation Function:

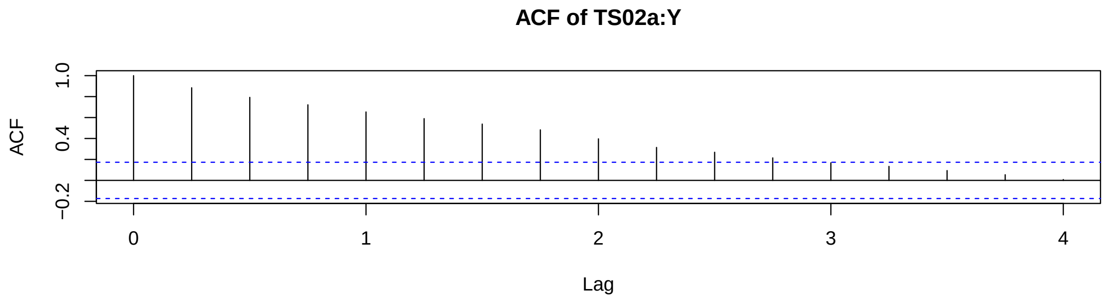
$$\hat{\rho}_k = \frac{\sum_{t=k+1}^{T}(Y_t - \overline{Y})(Y_{t-k} - \overline{Y})}{\sum_{t=1}^{T}(Y_t - \overline{Y})^2}$$

$$= \frac{\hat{\gamma}_k}{\hat{\gamma}_0}\ ,\ k = 0, 1, 2, 3, \ldots$$

Characteristics of Economic Time Series

For series Y in ts02a

```
ts02a[, "Y"] %>% na.omit() %>% acf(16, main="ACF of TS02a:Y")
```

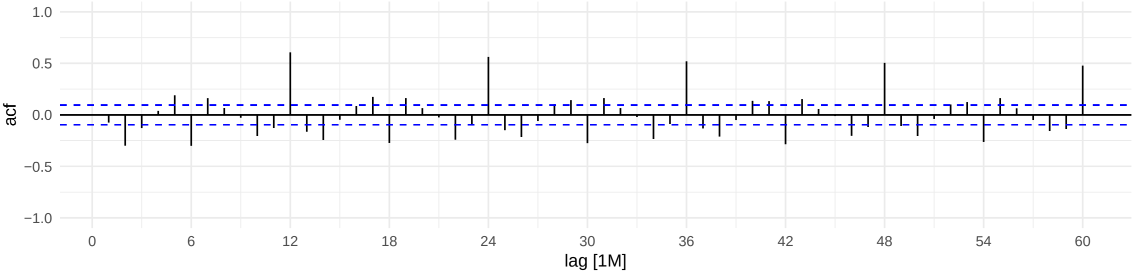


- "Lags" given in fraction of year (quarterly data, so lags given in 0.25)
- decaying ACF is typical of cyclical data

Characteristics of Economic Time Series

For growth in TOUR_SG series in ts01

```
ts01 %>% ACF(TOUR_SG_GROWTH, lag_max=60) %>% autoplot() + theme_minimal() + ylim(-1,1)
```

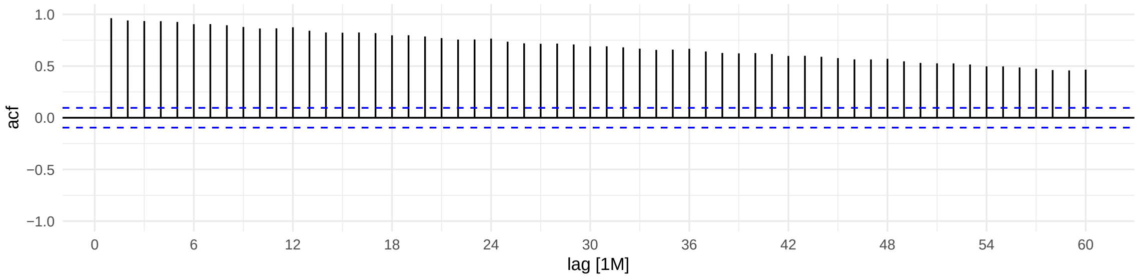


Seasonality show up as spikes at “seasonal lags” (12 for monthly data) in ACF

Characteristics of Economic Time Series

For LN_IP_SG series in ts01

```
ts01 %>% ACF(TOUR_SG, lag_max=60) %>% autoplot() + theme_minimal() + ylim(-1,1)
```



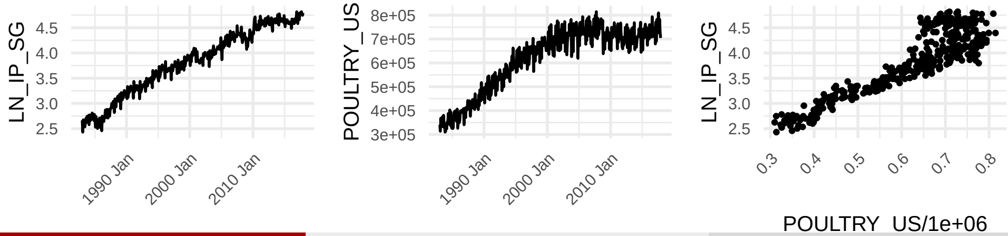
- Typical for trending series
- Trend dominates seasonality and cycles in the ACF

Characteristics of Economic Time Series

The presence of trend, seasonality, cycles can affect the behavior of regressions, e.g.,

$$\log IP\_SG_t = \beta_0 + \beta_1 POULTRY\_US_t + \epsilon_t$$

```
theme2 <- theme_minimal() + theme(text=element_text(size=8), axis.text.x = element_text(angle=45, hjust=1
p1 <- ts01 %>% autoplot(LN_IP_SG) + theme_minimal() + xlab("") + theme_bw() + theme2
p2 <- ts01 %>% autoplot(POULTRY_US) + theme_minimal() + xlab("") + theme_bw() + theme2
p3 <- ts01 %>% ggplot(aes(x=POULTRY_US/1000000, y=LN_IP_SG)) + geom_point(size=0.5) + theme_bw() + theme(
p1 | p2 | p3
```



Characteristics of Economic Time Series

```
fit_chicken <- ts01 %>%
  model(
    m1 = TSLM(LN_IP_SG ~ I(POULTRY_US/1000000)),
    m2 = TSLM(LN_IP_SG ~ I(POULTRY_US/1000000) + trend() + I(trend()^2/1000) + season())
  )
fit_chicken %>% select(m1) %>% coefficients()
```

```
# A tibble: 2 x 6
  .model term                  estimate std.error statistic   p.value
  <chr>  <chr>                    <dbl>     <dbl>     <dbl>     <dbl>
1 m1     (Intercept)               1.06    0.0616      17.3 7.64e- 51
2 m1     I(POULTRY_US/1e+06)       4.45    0.0975      45.6 2.01e-164
```

Model `m1` gives a silly result

Model `m2` takes into account trend and seasonality

Characteristics of Economic Time Series

```
fit_chicken %>% select(m2) %>% coefficients()
```

```
# A tibble: 15 x 6
   .model term                 estimate std.error statistic   p.value
   <chr>  <chr>                   <dbl>     <dbl>     <dbl>     <dbl>
 1 m2     (Intercept)          2.48      0.0508     48.9    6.11e-172
 2 m2     I(POULTRY_US/1e+06)  0.0224    0.158       0.142  8.87e-  1
 3 m2     trend()              0.00771   0.000446   17.3    1.53e- 50
 4 m2     I(trend()^2/1000)   -0.00574   0.000718   -7.99   1.38e- 14
 5 m2     season()year2       -0.100     0.0230     -4.36   1.62e-  5
 6 m2     season()year3        0.113     0.0216      5.23   2.72e-  7
 7 m2     season()year4       -0.000401  0.0217     -0.0185 9.85e-  1
 8 m2     season()year5       -0.00324   0.0217     -0.149  8.82e-  1
 9 m2     season()year6        0.0427    0.0217      1.97   4.93e-  2
10 m2     season()year7        0.0345    0.0216      1.60   1.11e-  1
11 m2     season()year8        0.0445    0.0222      2.00   4.59e-  2
12 m2     season()year9        0.0831    0.0216      3.84   1.41e-  4
13 m2     season()year10       0.0591    0.0219      2.70   7.22e-  3
14 m2     season()year11       0.00804   0.0229      0.351  7.26e-  1
15 m2     season()year12       0.0825    0.0220      3.74   2.08e-  4
```

Elementary Time Series Models

Discuss some elementary time series models for describing these characteristics

- Trends
  - Deterministic trend, stochastic trend
- Seasonality
  - Seasonal dummies, “stochastic seasonality”
- Cycles
  - Covariance-Stationary Autoregression of Order 1

Effect on OLS estimation of linear regression model, and appropriate remedial action depends on nature of features (next week)

Elementary Time Series Models

Trend

- "deterministic" or "stochastic"

Deterministic Trend Models

$$Y_t = f(t) + \epsilon_t \, , \, t = 1, 2, 3, \ldots$$

- Deterministic Linear Trend: $Y_t = \beta_0 + \beta_1 t + \epsilon_t \, , \;\; t = 1, 2, \ldots$
- Deterministic Quadratic Trend: $Y_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t \, , \;\; t = 1, 2, \ldots$
- Many other possibilities
- As long as $f(t)$ is linear-in-parameters, can estimate with OLS

Deterministic Trend

```
fit_lnipsg <- ts01 %>%
  model(ip1 = TSLM(LN_IP_SG ~ trend() + I(trend()^2/1000)))
fit_lnipsg %>% tidy() %>% mutate(across(where(is.numeric), ~ round(.x, 3))) %>% as.data.frame() %>%
  print(row.names=FALSE)
fit_lnipsg %>% glance() %>% select(r_squared) %>% round(4) %>% as.data.frame() %>%
  print(row.names=FALSE)
```

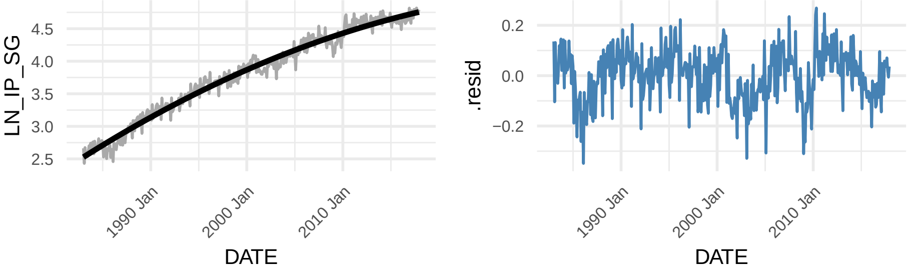
```
 .model              term estimate std.error statistic p.value
    ip1       (Intercept)    2.519     0.015   164.709       0
    ip1           trend()    0.008     0.000    46.332       0
    ip1 I(trend()^2/1000)   -0.006     0.000   -15.114       0
 r_squared
    0.9752
```

```
fitted_ip1 <- fit_lnipsg %>% select(ip1) %>% augment()
fitted_ip1 %>% colnames()
```

```
[1] ".model"   "DATE"     "LN_IP_SG" ".fitted"  ".resid"   ".innov"
```

Deterministic Trend

```r
p1 <- fitted_ip1 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = LN_IP_SG), color="darkgrey") +
  geom_line(aes(y = .fitted), color="black", linewidth=1) + theme0
p2 <- fitted_ip1 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = .resid), color="steelblue") + theme0
p1 | p2
```



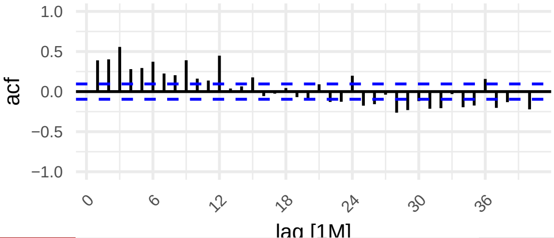
Deterministic Trend

Think of residuals from this regression as

- "detrended" LN_IP_SG
- Manufacturing output gap

Contains the seasonal, cyclical and other (if any) components

```
fitted_ip1 %>% ACF(.resid, lag_max=40) %>% autoplot() + theme2 + ylim(-1,1)
```



Deterministic Trend (Nonparametric Approach)

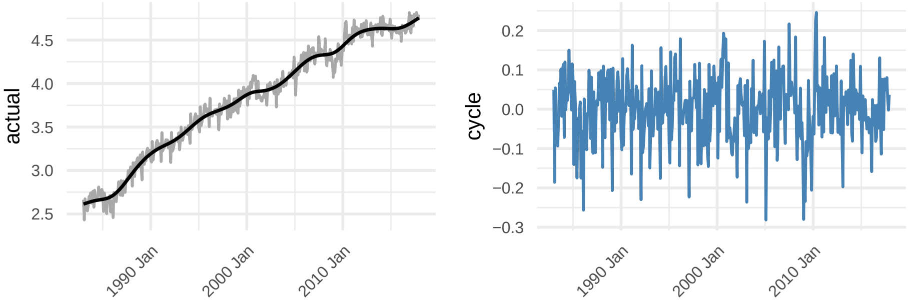
A Nonparametric Approach (HP Filter)

$$\hat{\tau}_t^{hp} = \text{argmin}_{\hat{\tau}_t} \left(\sum_{t=1}^{T} (y_t - \hat{\tau}_t)^2 + \lambda \sum_{t=2}^{T-1} \left[(\hat{\tau}_{t+1} - \hat{\tau}_t) - (\hat{\tau}_t - \hat{\tau}_{t-1}) \right]^2 \right)$$

```
ts01.ts <- ts01 %>% as.ts()
lnipsg_hp <- hpfilter(ts01.ts[,"LN_IP_SG"], type="lambda", freq=14400)
lnipsg_hp <- as_tsibble(ts.union("actual" = lnipsg_hp$x,
                                 "cycle" = lnipsg_hp$cycle,
                                 "trend" = lnipsg_hp$trend), pivot_longer=F)
p1 <- autoplot(lnipsg_hp, actual, color="darkgrey") +
      autolayer(lnipsg_hp, trend, size=0.6, color="black") +
      theme(legend.position = "bottom") + theme0 + xlab("")
p2 <- autoplot(lnipsg_hp, cycle, color="steelblue") + theme2 + xlab("")
```

Deterministic Trend (Nonparametric Approach)

```
p1 | p2
```



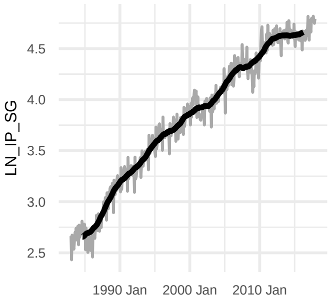
Deterministic Trend (Nonparametric Approach)

Moving Average:

$$\hat{\tau}_t^{ma} = \frac{1}{2k+1} \sum_{j=-k}^{k} Y_{t+j}, t = k+1, ..., T-k$$

```
lnipsg_ma <- ts01 %>% select(LN_IP_SG) %>%
  mutate("LN_IP_SG_MA"=as.numeric(NA))
k <- 20
T <- dim(lnipsg_ma)[1]
for (i in (k+1):(T-k)){
  lnipsg_ma[i,"LN_IP_SG_MA"] <- mean(
    lnipsg_ma$LN_IP_SG[(i-k):(k+i)]
    )
}
p1 <- autoplot(lnipsg_ma, LN_IP_SG, color="darkgray") +
  autolayer(lnipsg_ma, LN_IP_SG_MA,
            linewidth=1, color="black") +
  theme1 + xlab("")
```

```
p1
```



Stochastic Trend

"Stochastic Trend": Models Y_t as

$$Y_t - Y_{t-1} = \alpha + \epsilon_t \quad \text{i.e.,} \quad Y_t = \alpha + Y_{t-1} + \epsilon_t \quad \text{for all} \quad t$$

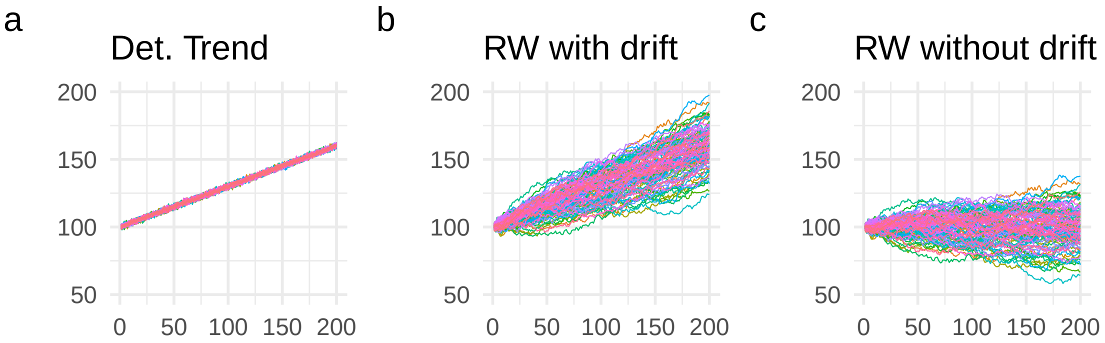
"Random Walk" (with drift if $\alpha \neq 0$)

Essential difference between "stochastic trend" and "deterministic trend"

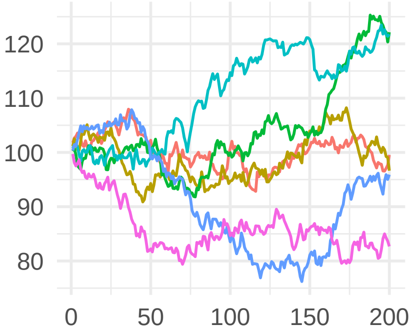
$$\begin{aligned}
Y_1 &= \alpha + Y_0 + \epsilon_1 & Var(Y_1 \mid Y_0) &= \sigma^2 \\
Y_2 &= \alpha + Y_1 + \epsilon_2 = Y_0 + 2\alpha + \epsilon_1 + \epsilon_2 & Var(Y_2 \mid Y_0) &= 2\sigma^2 \\
Y_3 &= \alpha + Y_2 + \epsilon_3 = Y_0 + 3\alpha + \epsilon_1 + \epsilon_2 + \epsilon_3 & Var(Y_3 \mid Y_0) &= 3\sigma^2 \\
&\vdots & &\vdots \\
Y_t &= \alpha + Y_{t-1} + \epsilon_t = Y_0 + \alpha t + \epsilon_1 + \epsilon_2 + \epsilon_3 + \cdots + \epsilon_t & Var(Y_t \mid Y_0) &= t\sigma^2
\end{aligned}$$

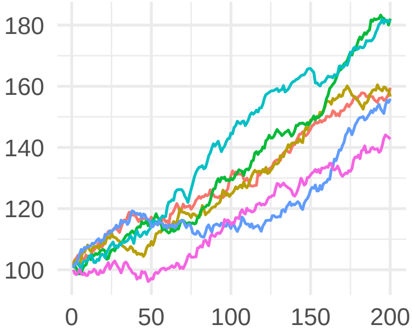
Stochastic Trend

Simulated det. trends and random walks, with unit error variance



Stochastic Trend





Stochastic Trend

```
fit_lnipsg <- ts01 %>%
  model(
    ip1 = TSLM(LN_IP_SG ~ trend() + I(trend()^2/1000)),
    ip2 = ARIMA(LN_IP_SG ~ 1 + pdq(0,1,0) + PDQ(0,0,0))
  )
fit_lnipsg %>% tidy()
```

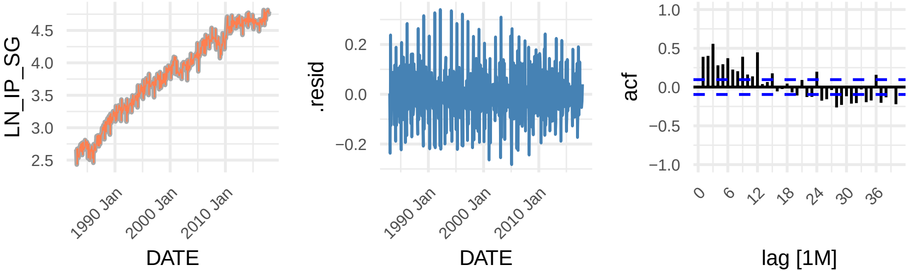
```
# A tibble: 4 x 6
  .model term                estimate std.error statistic   p.value
  <chr>  <chr>                  <dbl>     <dbl>     <dbl>     <dbl>
1 ip1    (Intercept)          2.52     0.0153     165.    0
2 ip1    trend()              0.00777  0.000168    46.3   1.52e-166
3 ip1    I(trend()^2/1000) -0.00583  0.000386   -15.1   1.81e- 41
4 ip2    constant             0.00508  0.00559      0.909 3.64e-  1
```

```
fitted_ip2 <- fit_lnipsg %>% select(ip2) %>% augment()
cat("ip2 r-squared: ",
  round(1 - sum(fitted_ip2$.resid^2) / sum((fitted_ip2$LN_IP_SG - mean(fitted_ip2$LN_IP_SG))^2), 4))
```

```
ip2 r-squared:  0.9698
```

Stochastic Trend

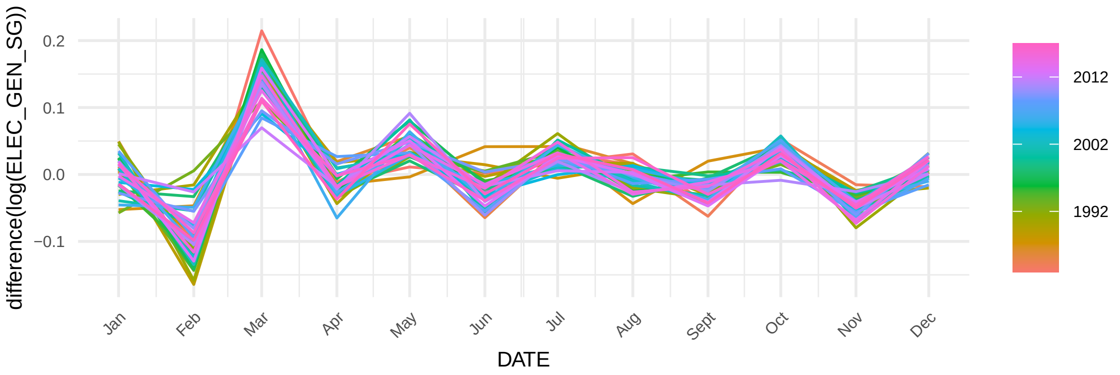
```
p1 <- fitted_ip2 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = LN_IP_SG), color="darkgrey", linewidth=0.8) +
  geom_line(aes(y = .fitted), color="coral", linewidth=0.2) + theme0
p2 <- fitted_ip2 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = .resid), color="steelblue") + theme0
p3 <- fitted_ip1 %>% ACF(.resid, lag_max=40) %>% autoplot() + theme2 + ylim(-1,1)
p1 | p2 | p3
```



Seasonality

Patterns with regular period, arising from ‘mechanical’ reasons

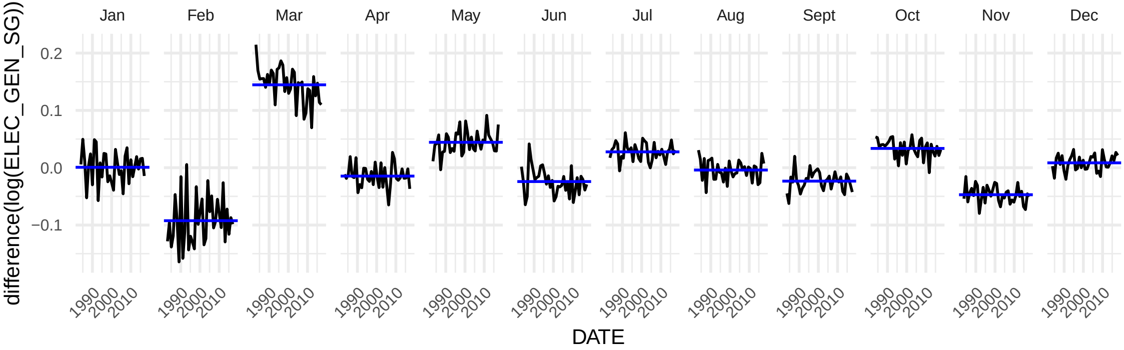
```
ts01 %>% gg_season(difference(log(ELEC_GEN_SG))) + theme2
```



Seasonality

Patterns with regular period, arising from 'mechanical' reasons

```
ts01 %>% gg_subseries(difference(log(ELEC_GEN_SG))) + theme2
```



Modelling Seasonality with Seasonal Dummies

Date	$d_{1,t}$	$d_{2,t}$	$d_{3,t}$	$d_{4,t}$	$d_{5,t}$	$d_{6,t}$	$d_{7,t}$	$d_{8,t}$	$d_{9,t}$	$d_{10,t}$	$d_{11,t}$	$d_{12,t}$
Jan 1980	1	0	0	0	0	0	0	0	0	0	0	0
Feb 1980	0	1	0	0	0	0	0	0	0	0	0	0
Mar 1980	0	0	1	0	0	0	0	0	0	0	0	0
Apr 1980	0	0	0	1	0	0	0	0	0	0	0	0
May 1980	0	0	0	0	1	0	0	0	0	0	0	0
Jun 1980	0	0	0	0	0	1	0	0	0	0	0	0
Jul 1980	0	0	0	0	0	0	1	0	0	0	0	0
Aug 1980	0	0	0	0	0	0	0	1	0	0	0	0
Sep 1980	0	0	0	0	0	0	0	0	1	0	0	0
Oct 1980	0	0	0	0	0	0	0	0	0	1	0	0
Nov 1980	0	0	0	0	0	0	0	0	0	0	1	0
Dec 1980	0	0	0	0	0	0	0	0	0	0	0	1
Jan 1981	1	0	0	0	0	0	0	0	0	0	0	0
Feb 1981	0	1	0	0	0	0	0	0	0	0	0	0
Mar 1981	0	0	1	0	0	0	0	0	0	0	0	0
Apr 1981	0	0	0	1	0	0	0	0	0	0	0	0
May 1981	0	0	0	0	1	0	0	0	0	0	0	0
Jun 1981	0	0	0	0	0	1	0	0	0	0	0	0

Modelling Seasonality with Seasonal Dummies

Three equivalent specifications

$$Y_t = \beta_1 d_{1,t} + \beta_2 d_{2,t} + \cdots + \beta_{12} d_{12,t} + \epsilon_t$$

$$Y_t = \alpha_0 + \alpha_2 d_{2,t} + \cdots + \alpha_{12} d_{12,t} + \epsilon_t$$

$$Y_t = \delta_0 + \delta_2 (d_{2,t} - \tfrac{1}{12}) + \delta_3 (d_{3,t} - \tfrac{1}{12}) + \cdots + \delta_{12} (d_{12,t} - \tfrac{1}{12}) + \epsilon_t$$

Can use in combination with other regressors, e.g.,

- Seasonal Dummies with Quadratic Trend

$$Y_t = \alpha_0 + \alpha_2 d_{2,t} + \cdots + \alpha_{12} d_{12,t} + \beta_1 t + \beta_2 t^2 + \epsilon_t$$

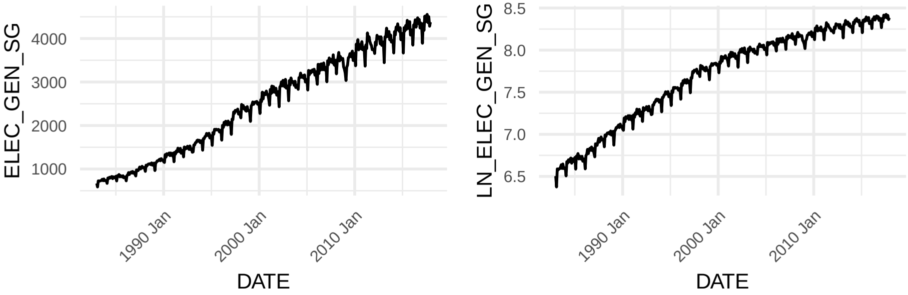
- Seasonal Dummies with Stochastic Trend

$$Y_t = \alpha_0 + Y_{t-1} + \alpha_2 d_{2,t} + \cdots + \alpha_{12} d_{12,t} + \epsilon_t$$

Modelling Seasonality with Seasonal Dummies

Application to ELEC_GEN_SG

```
ts01 <- ts01 %>% mutate(LN_ELEC_GEN_SG = log(ELEC_GEN_SG))
p1 <- ts01 %>% ggplot(aes(x=DATE)) + geom_line(aes(y=ELEC_GEN_SG)) + theme0
p2 <- ts01 %>% ggplot(aes(x=DATE)) + geom_line(aes(y=LN_ELEC_GEN_SG)) + theme0
p1 | p2
```



Modelling Seasonality with Seasonal Dummies

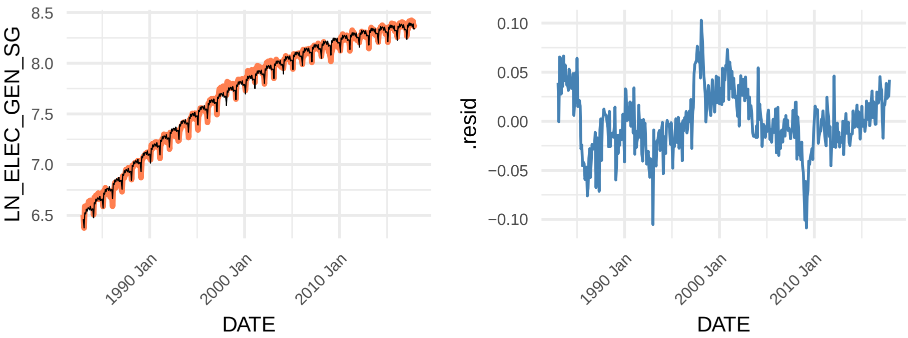
Applying Cubic Det. Trend with Seasonal Dummies to LN_ELEC_GEN_SG

```
fit_elecsg1 <- ts01 %>%
  model(
    elec1a = TSLM(LN_ELEC_GEN_SG ~ trend() + I(trend()^2/1000) + season()),
    elec1b = TSLM(LN_ELEC_GEN_SG ~ 1 + lag(LN_ELEC_GEN_SG) + season())
    )
fit_elecsg2 <- ts01 %>%
  model(
    elec2 = TSLM(log(ELEC_GEN_SG) ~ 1 + lag(log(ELEC_GEN_SG)) + season())
    )

fitted_elec1a <- fit_elecsg1 %>% select(elec1a) %>% augment()
fitted_elec1b <- fit_elecsg1 %>% select(elec1b) %>% augment()
fitted_elec2 <- fit_elecsg2 %>% select(elec2) %>% augment()
```

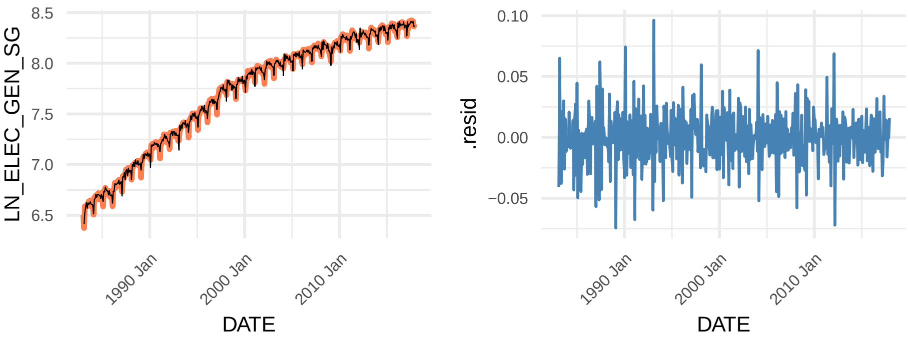
Modelling Seasonality with Seasonal Dummies

```
p1a <- fitted_elec1a %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = LN_ELEC_GEN_SG), color="coral", linewidth=1) +
  geom_line(aes(y = .fitted), color="black", linewidth=0.2) + theme0
p2a <- fitted_elec1a %>% ggplot(aes(x=DATE)) + geom_line(aes(y = .resid), color="steelblue") + theme0
p1a | p2a
```



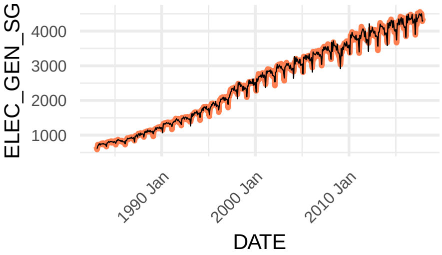
Modelling Seasonality with Seasonal Dummies

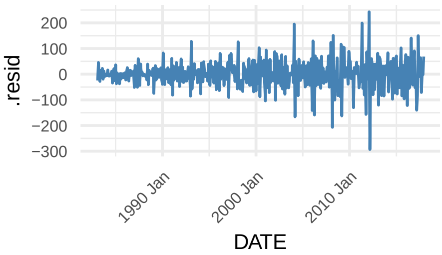
```
p1b <- fitted_elec1b %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = LN_ELEC_GEN_SG), color="coral", linewidth=1) +
  geom_line(aes(y = .fitted), color="black", linewidth=0.2) + theme0
p2b <- fitted_elec1b %>% ggplot(aes(x=DATE)) + geom_line(aes(y = .resid), color="steelblue") + theme0
p1b | p2b
```



Modelling Seasonality with Seasonal Dummies

```
p3 <- fitted_elec2 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = ELEC_GEN_SG), color="coral", linewidth=1) +
  geom_line(aes(y = .fitted), color="black", linewidth=0.2) + theme0
p4 <- fitted_elec2 %>% ggplot(aes(x=DATE)) + geom_line(aes(y = .resid), color="steelblue") + theme0
p3 | p4
```





With `log(ELEC_GEN_SG)`, fitted value are converted back to `ELEC_GEN_SG`

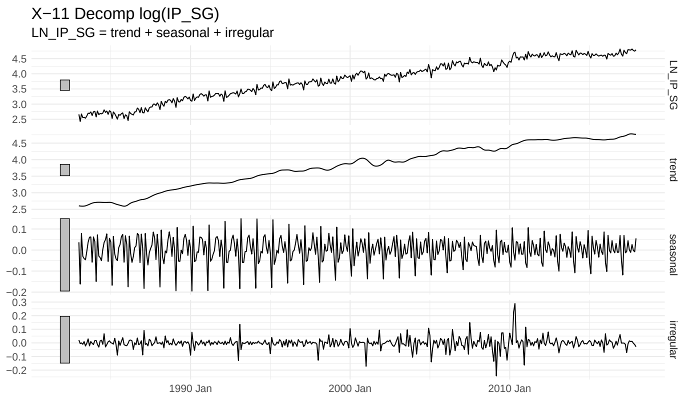
Modelling Seasonality with Seasonal Dummies

- Seasonal dummy approach assumes "very regular" seasonal patterns — may or may not be appropriate
- Other methods (e.g., seasonal ARIMA) — not covered in this course
- Seasonally Adjusted Data
  - Official statistics agencies often provide seasonally-adjusted data
  - Typical Method
    - Estimate "trend-cycle" nonparametrically (e.g., moving average methods)
    - Estimate seasonal component nonparametrically (e.g., "seasonal" m.a. methods)
    - Remove seasonal component from original series

Seasonally-Adjusted Series

```r
ipsg_dcmp <- ts01 %>%
  model(
    x11=X_13ARIMA_SEATS(LN_IP_SG ~ x11())
  ) %>%
  components()
p_decomp <- autoplot(ipsg_dcmp) +
  theme1 +
  labs(title="X-11 Decomp log(IP_SG)") +
  xlab("") +
  theme(text=element_text(size=14))
```

```r
p_decomp
```

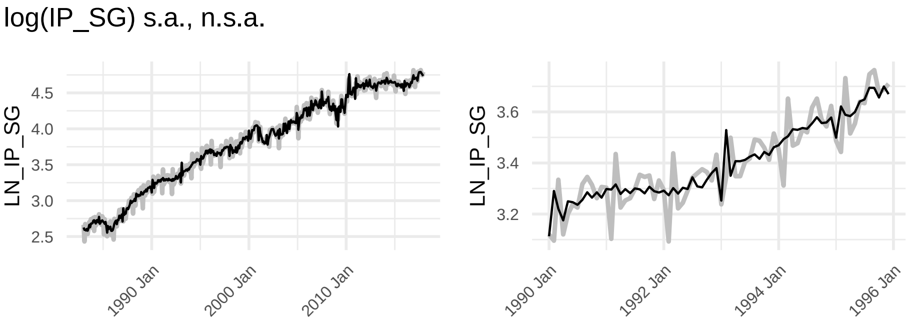


Seasonally-Adjusted Series

```
p2 <- ipsg_dcmp %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = LN_IP_SG), color="grey", size=0.8) +
  geom_line(aes(y = season_adjust), size=0.4) + theme2 + xlab("")
p3 <- ipsg_dcmp %>%
  filter_index("1990M1" ~"1995M12") %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = LN_IP_SG), size=0.8, color="grey") +
  geom_line(aes(y = season_adjust), size=0.4) + theme2 + xlab("")
```

Seasonally-Adjusted Series

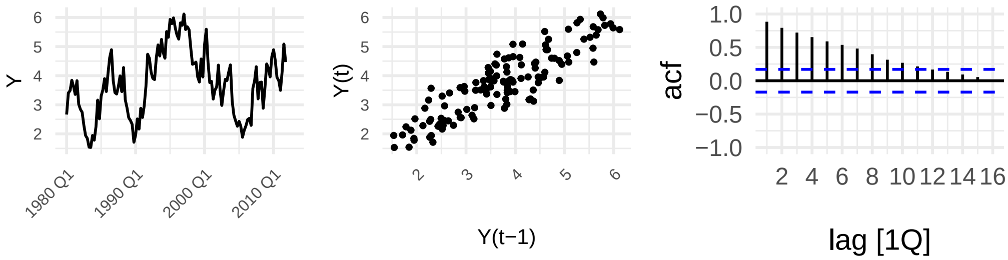
```
(p2 | p3) +
  plot_annotation(
    title="log(IP_SG) s.a., n.s.a.",
    theme = theme(title = element_text(size=8))
    )
```



Modelling Cycles

Recall series Y from ts02

```
ts02a.tsb <- as_tsibble(ts02a, pivot_longer = FALSE)
p1 <- ts02a.tsb %>% autoplot(Y) + theme(aspect.ratio = 0.7) + xlab("") + theme0
p2 <- ts02a.tsb %>% ggplot(aes(x=Y_1, y=Y)) + geom_point(size=0.6) +
  ylab("Y(t)") + xlab("Y(t-1)") + theme0
p3 <- ts02a.tsb %>% ACF(Y, lag_max=16) %>% autoplot() + theme_minimal() + ylim(-1,1)
p1 | p2 | p3
```



Modelling Cycles

Scatterplot suggests something like:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t$$

may be suitable for modeling cycles

"Autoregression of Order 1" or "AR(1)" — Member of ARIMA class of models

- ARIMA models not covered in detail here
- We limit ourselves to AR(1) in this course

Modelling Cycles

We have already seen an AR(1)

- Random Walk: $Y_t = \beta_0 + Y_{t-1} + \epsilon_t$, i.e., $\beta_1 = 1$
- Used to describe "stochastic trend"

For cycles, we use the "covariance-stationary AR(1)"

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t \text{ with } |\beta_1| < 1\,,\ \epsilon_t \overset{iid}{\sim} (0, \sigma^2)$$

Covariance Stationary Processes

A **Covariance Stationary** process Y_t is one where

- $E(Y_t)$ is the same finite constant for all t
- $Var(Y_t)$ is the same finite constant for all t
- $Cov(Y_t, Y_{t-k})$ is, for any k, the same finite constant for all t
  - May be different for different k
  - but for any given k, same for all t, i.e.,

Covariance-Stationary Processes are "stable" processes

- Expect steady fluctuations around a constant (no trend, no changes in variance)
- Autocorrelation patterns not changing over time

Covariance Stationary Processes

Example A white noise process is covariance stationary:

$$E(Y_t) = 0,\ Var(Y_t) = \sigma^2,\ Cov(Y_t, Y_s) = 0 \text{ for all } t, s, t \neq s$$

Example A (pure) deterministic trend process is not covariance stationary, since

$$Y_t = \beta_0 + \beta_1 t + \epsilon_t\,,\ \epsilon_t \sim WN(0, \sigma^2) \implies E(Y_t) = \beta_0 + \beta_1 t \text{ depends on } t$$

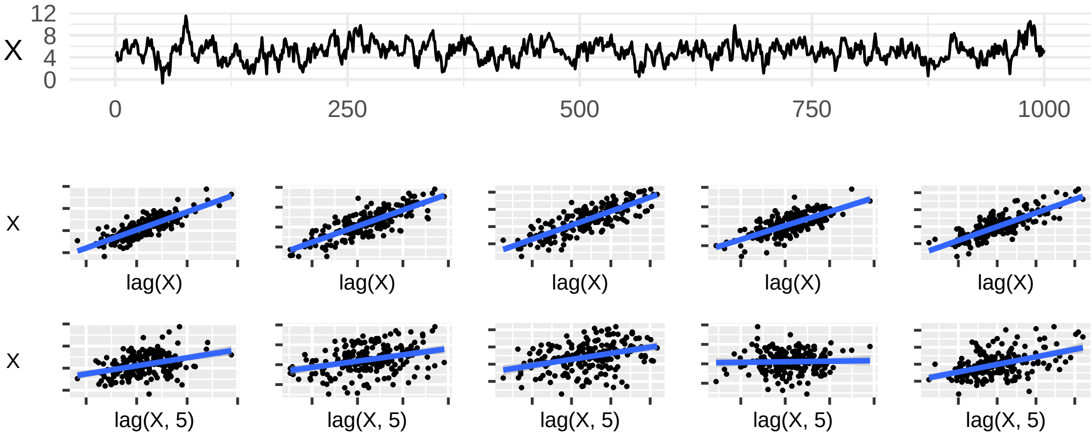
Example Random walk processes are not covariance-stationary

Example Covariance-stationary AR(1)

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t \text{ with } |\beta_1| < 1\,,\ \epsilon_t \overset{iid}{\sim} (0, \sigma^2)$$

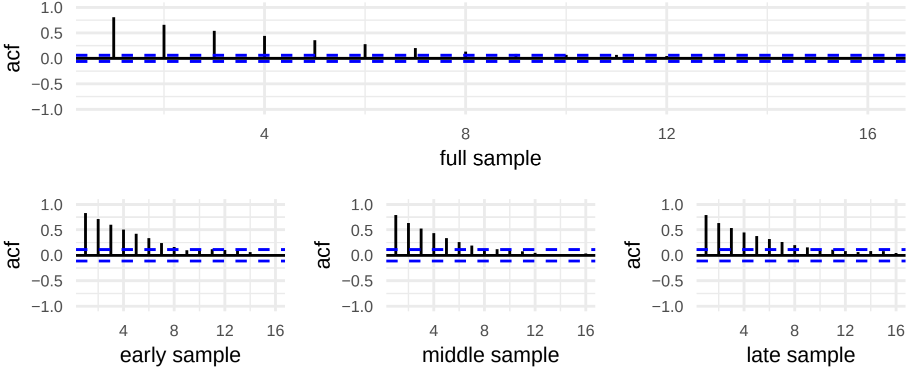
Proof of covariance-stationarity omitted

Covariance-Stationary AR(1)



Covariance-Stationary AR(1)

ACF of X, various subsamples



ACF $\to 0$ as lag tends to infinity: **"weak-dependence"**

Combining Elements

A few points regarding the covariance-stationary AR(1)

- It can be shown (exercise) that for the covariance-stationary AR(1)

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \epsilon_t \text{ with } |\beta_1| < 1$$

  we have $E(Y_t) = \dfrac{\beta_0}{1-\beta_1}$

- $\beta_0 = 0 \iff E(Y_t) = 0$
- We can write the AR(1) as constant c plus zero-mean AR(1) error

$$Y_t = c + u_t\,,\; u_t = \rho u_{t-1} + \epsilon_t$$

Combining Elements

Often prefer to put AR(1) into error (easier to interpret)

e.g. Linear det. trend + zero-mean AR(1) error

$$Y_t = \alpha + \delta t + u_t \, , \;\; u_t = \rho u_{t-1} + \epsilon_t \, , \; |\rho| < 1 \, , \; \epsilon \overset{iid}{\sim} (0, \sigma^2)$$

e.g. Quad. det. trend + seasonal dummies + zero-mean AR(1) error

$$Y_t = \alpha + (\text{seasonal dummies}) + \delta_1 t + \delta_2 t^2 + u_t \, , \;\; u_t = \rho u_{t-1} + \epsilon_t \, , \; |\rho| < 1 \, , \; \epsilon \overset{iid}{\sim} (0, \sigma^2)$$

Combining Elements

ip3 below is quad. det. trend + seasonal dummies + zero-mean AR(1) error

```
fit_lnipsg <- ts01 %>%
  model(
    ip1 = TSLM(LN_IP_SG ~ trend() + I(trend()^2/1000)),
    ip2 = ARIMA(LN_IP_SG ~ season() + trend() + I(trend()^2/1000)),
    ip3 = ARIMA(LN_IP_SG ~ season() + trend() + I(trend()^2/1000) + pdq(1,0,0) + PDQ(0,0,0))
    )
fitted_ip3 <- fit_lnipsg %>% select(ip3) %>% augment()
acf_ip3 <- fitted_ip1 %>% ACF(.resid, lag_max=40) %>% autoplot() + theme2 + ylim(-1,1)
p3a <- fitted_ip3 %>% ggplot(aes(x=DATE)) + geom_line(aes(y = LN_IP_SG), color="darkgrey") +
  geom_line(aes(y = .fitted), color="black", linewidth=0.3) + theme0
p3b <- fitted_ip3 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = .resid), color="steelblue") + theme0
p3c <- fitted_ip3 %>% ACF(.resid, lag_max=40) %>% autoplot() + theme2 + ylim(-1,1)
```

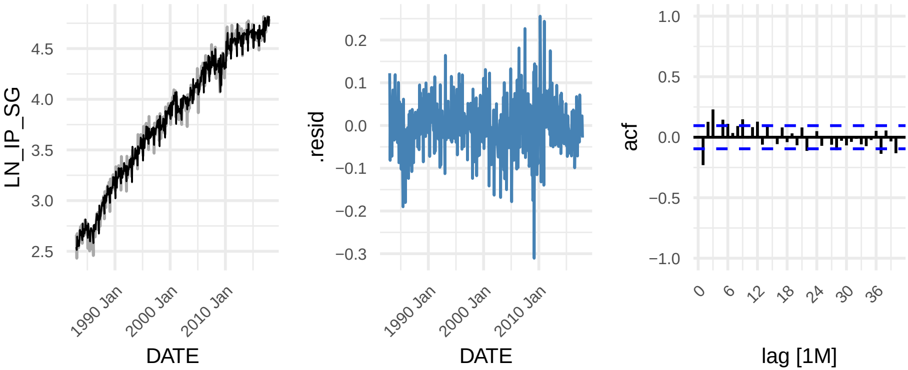
Combining Elements

```
fit_lnipsg %>% select(ip3) %>% tidy() %>% mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
  as.data.frame() %>% print(row.names=FALSE)
```

```
 .model               term estimate std.error statistic p.value
    ip3                ar1    0.643     0.037    17.169   0.000
    ip3      season()year2   -0.101     0.013    -7.995   0.000
    ip3      season()year3    0.113     0.016     7.010   0.000
    ip3      season()year4    0.000     0.018    -0.016   0.987
    ip3      season()year5   -0.002     0.019    -0.128   0.898
    ip3      season()year6    0.044     0.020     2.222   0.027
    ip3      season()year7    0.035     0.020     1.790   0.074
    ip3      season()year8    0.046     0.020     2.353   0.019
    ip3      season()year9    0.084     0.019     4.408   0.000
    ip3     season()year10    0.061     0.018     3.385   0.001
    ip3     season()year11    0.009     0.016     0.571   0.568
    ip3     season()year12    0.085     0.013     6.668   0.000
    ip3            trend()    0.008     0.000    25.222   0.000
    ip3 I(trend()^2/1000)   -0.006     0.001    -8.246   0.000
    ip3          intercept    2.496     0.030    84.077   0.000
```

Combining Elements

```
p3a | p3b | p3c
```



Combining Elements

Remarks:

- If model is captures all "dynamics" in the data, residual ACF should show mostly insignificant
- May require more than AR(1) errors
  - We are limiting ourselves to AR(1) errors in this course
- Technically, estimation with AR(1) errors uses "maximum likelihood estimation", not OLS
- deterministic trend or stochastic trend?

Combining Elements

Combining stochastic trend and cycle is a bit more complicated

Let $\Delta Y_t = Y_t - Y_{t-1}$

- To combine stochastic trend and cov-stat AR(1), we model Y_t as

$$\Delta Y_t = \beta_0 + \beta_1 \Delta Y_{t-1} + \epsilon\,,\ |\beta_1| < 1\,,\ \epsilon_t \overset{iid}{\sim} (0, \sigma^2)$$

  This is **not** RW + AR(1)

- To combine stochastic trend, cov-stat AR(1) and seasonal dummies

$$Z_t = Y_t - \beta_0 - \delta_2 d_{2,t} - \cdots - \delta_{12} d_{12,t}$$
$$\Delta Z_t = \beta_1 \Delta Z_{t-1} + \epsilon\,,\ |\beta_1| < 1\,,\ \epsilon_t \overset{iid}{\sim} (0, \sigma^2)$$

Combining Elements

```
fit_lnipsg <- ts01 %>%
  model(
    ip1 = TSLM(LN_IP_SG ~ trend() + I(trend()^2/1000)),
    ip2 = ARIMA(LN_IP_SG ~ season() + trend() + I(trend()^2/1000)),
    ip3 = ARIMA(LN_IP_SG ~ season() + trend() + I(trend()^2/1000) + pdq(1,0,0) + PDQ(0,0,0)),
    ip4 = ARIMA(LN_IP_SG ~ 1 + pdq(1,1,0) + PDQ(0,0,0)),
    ip5 = ARIMA(LN_IP_SG ~ 1 + season() + pdq(1,1,0) + PDQ(0,0,0)),
    )
```

```
fit_lnipsg %>% select(ip4) %>% tidy() %>% mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
  as.data.frame() %>% print(row.names=FALSE)
```

```
 .model     term estimate std.error statistic p.value
    ip4      ar1   -0.510     0.042   -12.072   0.000
    ip4 constant    0.008     0.005     1.642   0.101
```

Combining Elements

```
fit_lnipsg %>% select(ip5) %>% tidy() %>% mutate(across(where(is.numeric), ~ round(.x, 3))) %>%
  as.data.frame() %>% print(row.names=FALSE)
```

```
 .model            term estimate std.error statistic p.value
    ip5             ar1   -0.468     0.043   -10.813   0.000
    ip5   season()year2   -0.101     0.012    -8.101   0.000
    ip5   season()year3    0.114     0.012     9.233   0.000
    ip5   season()year4    0.000     0.014     0.019   0.985
    ip5   season()year5   -0.002     0.015    -0.109   0.913
    ip5   season()year6    0.045     0.016     2.860   0.004
    ip5   season()year7    0.037     0.016     2.339   0.020
    ip5   season()year8    0.047     0.016     3.038   0.003
    ip5   season()year9    0.085     0.015     5.718   0.000
    ip5  season()year10    0.062     0.014     4.289   0.000
    ip5  season()year11    0.010     0.012     0.821   0.412
    ip5  season()year12    0.085     0.013     6.718   0.000
    ip5       intercept    0.005     0.002     2.267   0.024
```

Combining Elements

```
fitted_ip5 <- fit_lnipsg %>% select(ip5) %>% augment()
p5a <- fitted_ip5 %>% ggplot(aes(x=DATE)) + geom_line(aes(y = LN_IP_SG), color="darkgrey") +
  geom_line(aes(y = .fitted), color="black", linewidth=0.3) + theme0
p5b <- fitted_ip5 %>% ggplot(aes(x=DATE)) +
  geom_line(aes(y = .resid), color="steelblue") + theme0
p5c <- fitted_ip5 %>% ACF(.resid, lag_max=40) %>% autoplot() + theme2 + ylim(-1,1)
p5a | p5b | p5c
```