

Script R - Parte 2 - Thiago Menezes (ajuste para estilo do python)

1) Importação do conjunto de dados “dataset_st” no RStudio.

```
library(readxl)

dataset_st <- read_excel("dataset_st.xlsx")
head(dataset_st)
```

2) Conversão dos dados para em um objeto tsibble:

```
install.packages("dplyr")
install.packages("tsibble")
install.packages("fabletools")
install.packages("ggplot2")
```

```
library(dplyr)
library(tsibble)
library(fabletools)
library(ggplot2)
```

```
dataset_st <- dataset_st |>
  mutate(mes = yearmonth(mes)) |>
  as_tsibble(index = mes)
```

```
dataset_st
```

3) Geração das séries temporais por grupos:

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

```
# Paleta "tab10" do Matplotlib
tab10 <- c(
  "#1f77b4", "#ff7f0e", "#2ca02c", "#d62728",
  "#9467bd", "#8c564b", "#e377c2", "#7f7f7f",
  "#bcbd22", "#17becf"
)
```

```
theme_python <- function(base_size = 14) {
  theme_bw(base_size = base_size) +
  theme(
    panel.background = element_rect(fill = "white"),
    panel.grid.major = element_line(colour = "grey85", linewidth = 0.4),
    panel.grid.minor = element_line(colour = "grey92", linewidth = 0.2),
    panel.border   = element_rect(colour = "grey80", fill = NA),
    plot.title    = element_text(hjust = 0.5),
```

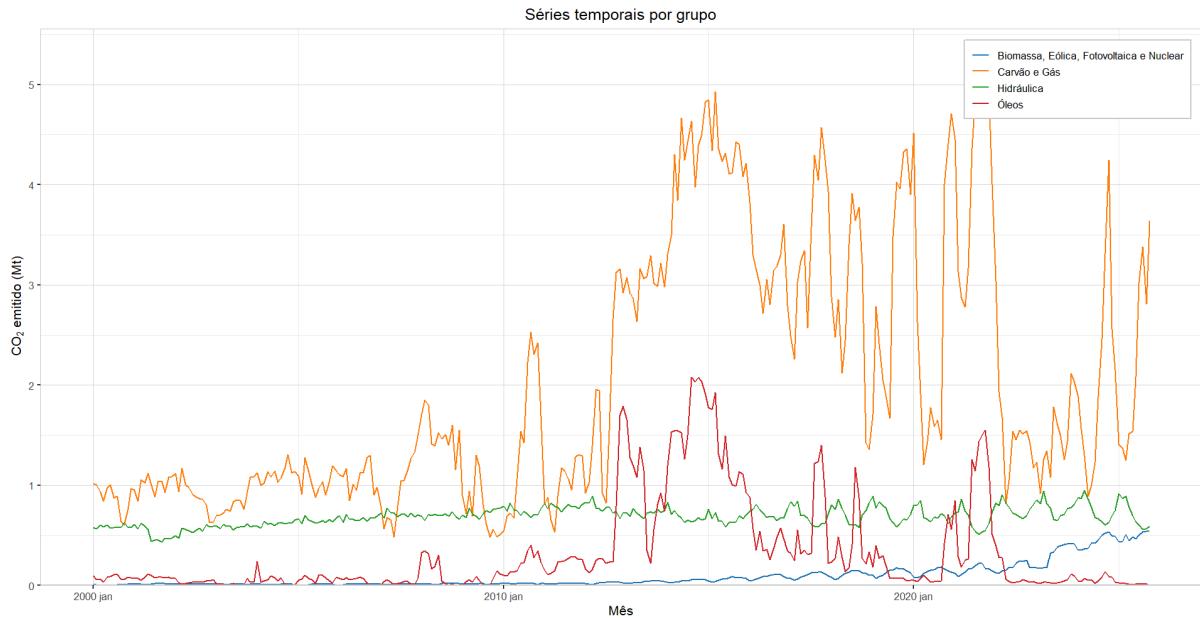
```

        legend.title    = element_blank()
    )
}

# Tema global
theme_set(theme_python())


dataset_st %>%
  select(mes, grupo_befn, grupo_cg, grupo_hidraulica, grupo_o) %>%
  pivot_longer(-mes, names_to = "grupo", values_to = "valor") %>%
  mutate(
    grupo = recode(
      grupo,
      grupo_befn     = "Biomassa, Eólica, Fotovoltaica e Nuclear",
      grupo_cg       = "Carvão e Gás",
      grupo_hidraulica = "Hidráulica",
      grupo_o        = "Óleos"
    )
  ) %>%
  ggplot(aes(x = mes, y = valor, colour = grupo)) +
  geom_line(linewidth = 0.9) +
  scale_colour_manual(values = tab10[1:4]) +
  labs(
    y = expression("CO"[2] * " emitido (Mt)"),
    x = "Mês",
    title = "Séries temporais por grupo"
  ) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.02))) +
  theme(
    legend.position    = c(0.99, 0.98), # dentro do gráfico, canto sup. dir.
    legend.justification = c(1, 1),
    legend.background  = element_rect(fill = "white", colour = "grey80"),
    legend.key.width   = unit(1.5, "lines")
  )

```



4) Séries temporais por grupos cortadas:

```

library(dplyr)
library(tidyr)
library(ggplot2)
library(lubridate)
library(tsibble)

# --- manter apenas TREINO (até 2023 Dez) ---
train_only <- dataset_st %>%
  filter(mes <= yearmonth("2023 Dec"))

# --- aplicar cortes ANTES de plotar ---
train_only_cut <- train_only %>%
  mutate(
    grupo_befn      = if_else(mes < yearmonth("2012 Jan"), NA_real_, grupo_befn),
    grupo_cg       = if_else(mes < yearmonth("2008 Jan"), NA_real_, grupo_cg),
    grupo_hidraulica = if_else(mes < yearmonth("2008 Jan"), NA_real_, grupo_hidraulica),
    grupo_o        = if_else(mes < yearmonth("2008 Jan"), NA_real_, grupo_o)
  )

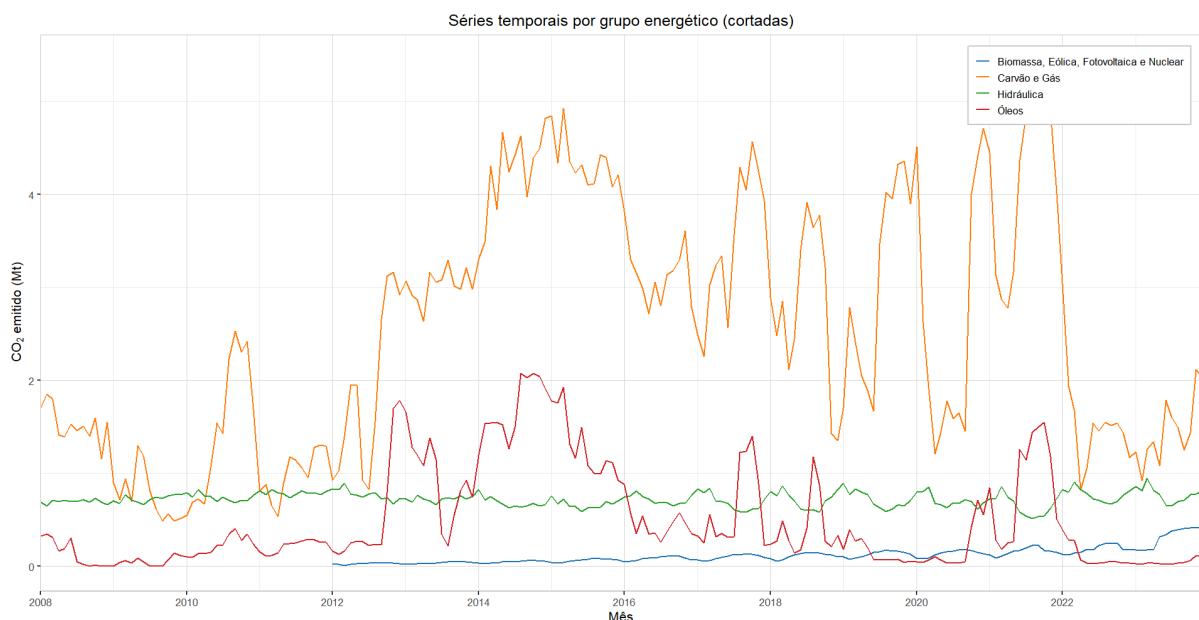
# --- GRÁFICO (somente treinamento, séries cortadas) ---
train_only_cut %>%
  filter(mes >= yearmonth("2005 Jan")) %>% # começa no ano desejado
  select(mes, grupo_befn, grupo_cg, grupo_hidraulica, grupo_o) %>%
  pivot_longer(-mes, names_to = "grupo", values_to = "valor") %>%
  mutate(
    grupo = recode(
      grupo,
      grupo_befn      = "Biomassa, Eólica, Fotovoltaica e Nuclear",
      grupo_cg        = "Carvão e Gás",
      grupo_hidraulica = "Hidráulica",
      grupo_o         = "Óleos"
    )
  )

```

```

grupo_cg      = "Carvão e Gás",
grupo_hidraulica = "Hidráulica",
grupo_o       = "Óleos"
),
date = as.Date(mes)
) %>%
ggplot(aes(x = date, y = valor, colour = grupo)) +
geom_line(linewidth = 0.9, na.rm = TRUE) +
scale_colour_manual(values = tab10[1:4]) +
labs(
  title = "Séries temporais por grupo energético (cortadas)",
  x = "Mês",
  y = expression("CO"[2] * " emitido (Mt)")
) +
scale_x_date(
  limits = c(as.Date(yeарmonth("2008 Jan")), max(as.Date(train_only_cut$mes))),
  expand = c(0, 0),
  date_breaks = "2 years",
  date_labels = "%Y"
) +
theme(
  legend.position = c(0.99, 0.98), # canto superior direito, dentro
  legend.justification = c(1, 1),
  legend.background = element_rect(fill = "white", colour = "grey80"),
  legend.title = element_blank()
)

```



5) Divisão dos dados em treinamento e teste:

```

# --- Treino: 2008–2023 ---
hidraulica_treino_2008_2023 <- dataset_st |>
  select(mes, grupo_hidraulica) |>
  filter_index("2008 Jan" ~ "2023 Dec")

# --- Teste: 2024–2025 ---
hidraulica_teste_2024_25 <- dataset_st |>
  select(mes, grupo_hidraulica) |>
  filter_index("2024 Jan" ~ "2025 Oct")

oleos_treino_2008_2023 <- dataset_st |>
  select(mes, grupo_o) |>
  filter_index("2008 Jan" ~ "2023 Dec")

oleos_teste_2024_25 <- dataset_st |>
  select(mes, grupo_o) |>
  filter_index("2024 Jan" ~ "2025 Oct")

cg_treino_2008_2023 <- dataset_st |>
  select(mes, grupo_cg) |>
  filter_index("2008 Jan" ~ "2023 Dec")

cg_teste_2024_25 <- dataset_st |>
  select(mes, grupo_cg) |>
  filter_index("2024 Jan" ~ "2025 Oct")

befn_treino_2012_2023 <- dataset_st |>
  select(mes, grupo_befn) |>
  filter_index("2012 Jan" ~ "2023 Dec")

befn_teste_2024_25 <- dataset_st |>
  select(mes, grupo_befn) |>
  filter_index("2024 Jan" ~ "2025 Oct")

```

6) Transformações Box-cox:

```

library(ggplot2)
library(fabletools)
library(feasts)
library(latex2exp)
library(patchwork)

# -----
# Hidráulica
# -----
lambda_hid <- hidraulica_treino_2008_2023 |>

```

```

features(grupo_hidraulica, features = guerrero) |>
pull(lambda_guerrero)

p_hid <- hidraulica_treino_2008_2023 |>
autoplot(box_cox(grupo_hidraulica, lambda_hid)) +
labs(y = "",
title = latex2exp::TeX(paste0(
  "Hidráulica — $\\lambda$ = ", round(lambda_hid,2)))) +
theme_minimal(14)

# -----
# Óleos
# -----
lambda_ole <- oleos_treino_2008_2023 |>
features(grupo_o, features = guerrero) |>
pull(lambda_guerrero)

p_ole <- oleos_treino_2008_2023 |>
autoplot(box_cox(grupo_o, lambda_ole)) +
labs(y = "",
title = latex2exp::TeX(paste0(
  "Óleos — $\\lambda$ = ", round(lambda_ole,2)))) +
theme_minimal(14)

# -----
# Carvão & Gás
# -----
lambda_cg <- cg_treino_2008_2023 |>
features(grupo_cg, features = guerrero) |>
pull(lambda_guerrero)

p_cg <- cg_treino_2008_2023 |>
autoplot(box_cox(grupo_cg, lambda_cg)) +
labs(y = "",
title = latex2exp::TeX(paste0(
  "Carvão e Gás — $\\lambda$ = ", round(lambda_cg,2)))) +
theme_minimal(14)

# -----
# Renováveis (BEFN)
# -----
lambda_befn <- befn_treino_2012_2023 |>
features(grupo_befn, features = guerrero) |>
pull(lambda_guerrero)

```

```

p_befn <- befn_treino_2012_2023 |>
  autoplot(box_cox(grupo_befn, lambda_befn)) +
  labs(y = "",
    title = latex2exp::TeX(paste0(
      "Biomassa, Eólica, Fotovoltaica e Nuclear — $\\lambda$ = ", round(lambda_befn,2))))
+
  theme_minimal(14)

# -----
# 2 x 2
# -----


(p_hid | p_ole) /
(p_cg | p_befn)

p_hid <- hidraulica_treino_2008_2023 |>
  autoplot(box_cox(grupo_hidraulica, lambda_hid)) +
  scale_colour_manual(values = tab10[1]) +
  labs(
    y = "",
    title = latex2exp::TeX(
      paste0("Hidráulica — $\\lambda$ = ", round(lambda_hid,2))
    )
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none"
  )

p_ole <- oleos_treino_2008_2023 |>
  autoplot(box_cox(grupo_o, lambda_ole)) +
  scale_colour_manual(values = tab10[2]) +
  labs(
    y = "",
    title = latex2exp::TeX(
      paste0("Óleos — $\\lambda$ = ", round(lambda_ole,2))
    )
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none"
  )

p_cg <- cg_treino_2008_2023 |>
  autoplot(box_cox(grupo_cg, lambda_cg)) +
  scale_colour_manual(values = tab10[3]) +

```

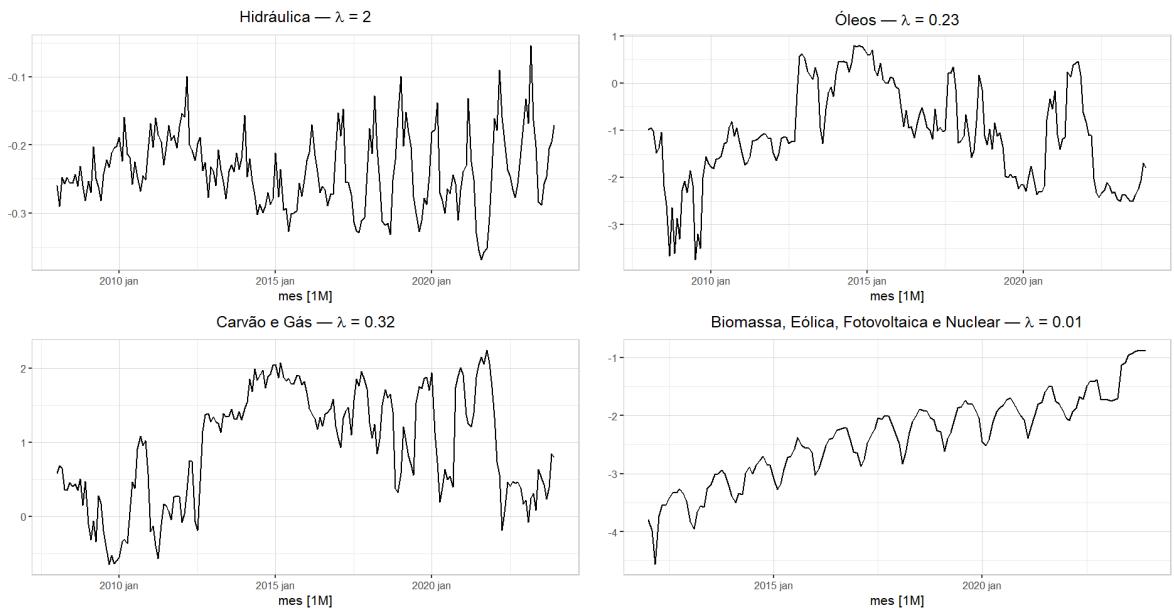
```

labs(
  y = "",
  title = latex2exp::TeX(
    paste0("Carvão e Gás —  $\lambda$  = ", round(lambda_cg,2))
  )
) +
theme(
  plot.title = element_text(hjust = 0.5),
  legend.position = "none"
)

p_befn <- befn_treino_2012_2023 |>
  autoplot(box_cox(grupo_befn, lambda_befn)) +
  scale_colour_manual(values = tab10[4]) +
  labs(
    y = "",
    title = latex2exp::TeX(
      paste0("Biomassa, Eólica, Fotovoltaica e Nuclear —  $\lambda$  = ", round(lambda_befn,2))
    )
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.position = "none"
  )

(p_hid | p_ole) /
(p_cg | p_befn)

```



7) Ajuste dos modelos ARIMA (automático) para cada grupo:

```
# HIDRÁULICA  
hidraulica_bc <- hidraulica_treino_2008_2023 %>%  
  mutate(co2_bc = box_cox(grupo_hidraulica, lambda_hid))
```

```
# ÓLEOS  
oleos_bc <- oleos_treino_2008_2023 %>%  
  mutate(co2_bc = box_cox(grupo_o, lambda_ole))
```

```
# CARVÃO & GÁS  
cg_bc <- cg_treino_2008_2023 %>%  
  mutate(co2_bc = box_cox(grupo_cg, lambda_cg))
```

```
# BEFN (Renováveis)  
befn_bc <- befn_treino_2012_2023 %>%  
  mutate(co2_bc = box_cox(grupo_befn, lambda_befn))
```

```
library(fable)  
library(fabletools)
```

```
# HIDRÁULICA  
fit_hid_arima <- hidraulica_bc %>%  
  model(  
    arima = ARIMA(co2_bc, stepwise = FALSE, approx = FALSE)  
)
```

```
# ÓLEOS  
fit_ole_arima <- oleos_bc %>%  
  model(  
    arima = ARIMA(co2_bc, stepwise = FALSE, approx = FALSE)  
)
```

```
# CARVÃO & GÁS  
fit_cg_arima <- cg_bc %>%  
  model(  
    arima = ARIMA(co2_bc, stepwise = FALSE, approx = FALSE)  
)
```

```
# RENOVÁVEIS + NUCLEAR (BEFN)  
fit_befn_arima <- befn_bc %>%  
  model(  
    arima = ARIMA(co2_bc, stepwise = FALSE, approx = FALSE)  
)
```

```

report(fit_hid_arima)
report(fit_ole_arima)
report(fit_cg_arima)
report(fit_befn_arima)

> report(fit_hid_arima)
Series: co2_bc
Model: ARIMA(2,0,0) (2,1,1) [12]

Coefficients:
      ar1      ar2      sar1      sar2      sma1
      0.9866   -0.1879   -0.0819   -0.2388   -0.6082
s.e.  0.0747    0.0762    0.1134    0.0902    0.1018

sigma^2 estimated as 0.0005725: log likelihood=413.37
AIC=-814.73  AICc=-814.25  BIC=-795.58
> report(fit_ole_arima)
Series: co2_bc
Model: ARIMA(0,1,4) (2,0,0) [12]

Coefficients:
      ma1      ma2      ma3      ma4      sar1      sar2
      -0.021   0.0503   -0.2379   -0.1198   0.0836   -0.1573
s.e.  0.073   0.0711   0.0828   0.0756   0.0788   0.0831

sigma^2 estimated as 0.1702: log likelihood=-99.3
AIC=212.6  AICc=213.21  BIC=235.36
> report(fit_cg_arima)
Series: co2_bc
Model: ARIMA(0,1,4) (2,0,0) [12]

Coefficients:
      ma1      ma2      ma3      ma4      sar1      sar2
      0.0971   -0.0947   -0.1529   -0.3150   0.0457   0.1613
s.e.  0.0710   0.0717   0.0759   0.0827   0.0723   0.0759

sigma^2 estimated as 0.08146: log likelihood=-29.07
AIC=72.15  AICc=72.76  BIC=94.91
> report(fit_befn_arima)
Series: co2_bc
Model: ARIMA(3,0,1) (0,1,1) [12] w/ drift

Coefficients:
      ar1      ar2      ar3      ma1      sma1  constant
      0.8309   -0.1311   0.2660   -0.4601   -0.8708    0.0077
s.e.  0.1913    0.1307   0.1322    0.1838    0.1624    0.0011

sigma^2 estimated as 0.01429: log likelihood=88
AIC=-162  AICc=-161.1  BIC=-141.82

```

8) Ajuste do e ETS com Box–Cox para cada grupo:

```

library(fable)

# HIDRÁULICA
fit_hid_ets <- hidraulica_bc %>%
  model(
    ets = ETS(co2_bc)
  )

# ÓLEOS
fit_ole_ets <- oleos_bc %>%
  model(
    ets = ETS(co2_bc)
  )

# CARVÃO & GÁS
fit_cg_ets <- cg_bc %>%
  model(
    ets = ETS(co2_bc)
  )

# BEFN
fit_befn_ets <- befn_bc %>%
  model(
    ets = ETS(co2_bc)
  )

fit_hid_ets
fit_ole_ets
fit_cg_ets
fit_befn_ets

report(fit_hid_ets)
report(fit_ole_ets)
report(fit_cg_ets)
report(fit_befn_ets)

> report(fit_hid_ets)
Series: co2_bc
Model: ETS(A,N,A)
  Smoothing parameters:
    alpha = 0.8386846
    gamma = 0.1578135

  Initial states:
    l[0]          s[0]          s[-1]         s[-2]         s[-3]         s[-4]
  s[-5]          s[-6]          s[-7]

```

```

-0.3380537  0.04870997  0.01194264  0.001345753 -0.03386021 -0.007175918
-0.03963994 -0.07013598 -0.02834456
      s[-8]      s[-9]      s[-10]     s[-11]
-0.009038573 0.05204129 0.007170066 0.06698545

sigma^2: 8e-04

      AIC      AICC      BIC
-348.3707 -345.6434 -299.5082
> report(fit_ole_ets)
Series: co2_bc
Model: ETS(A,N,N)
Smoothing parameters:
alpha = 0.9906681

Initial states:
l[0]
-0.9838421

sigma^2: 0.1844

      AIC      AICC      BIC
688.7740 688.9016 698.5464
> report(fit_cg_ets)
Series: co2_bc
Model: ETS(A,N,N)
Smoothing parameters:
alpha = 0.9998999

Initial states:
l[0]
0.581717

sigma^2: 0.0898

      AIC      AICC      BIC
550.7614 550.8891 560.5339
> report(fit_befn_ets)
Series: co2_bc
Model: ETS(A,A,A)
Smoothing parameters:
alpha = 0.276589
beta  = 0.03029959
gamma = 0.0001002197

Initial states:
l[0]          b[0]          s[0]          s[-1]         s[-2]         s[-3]         s[-4]
s[-5]          s[-6]          s[-7]
-3.683144  0.02525095 -0.05123652  0.08197678  0.2111572  0.2626876  0.282074
0.1891946  0.1274097  0.04237296
      s[-8]      s[-9]      s[-10]     s[-11]
-0.1369123 -0.3295867 -0.4228454 -0.256292

```

```

sigma^2: 0.0135

      AIC      AICc      BIC
113.0455 117.9027 163.5323

```

9) Ajuste dos modelos BENCHMARKS:

```

library(dplyr)
library(fable)
library(fabletools)
library(tsibble)

# =====
# Função correta para inversa Box–Cox
# =====
# Agora shift tem valor padrão = 0 → NÃO OCORRE MAIS O ERRO
inv_box_cox <- function(y_bc, lambda, shift = 0) {
  if (abs(lambda) < 1e-8) {
    exp(y_bc) - shift
  } else {
    (lambda * y_bc + 1)^(1 / lambda) - shift
  }
}

# =====
# Utilitária para aplicar inverse BC nas colunas previstas
# =====
inv_bc_across <- function(fc_tbl, lambda, cols = c(".mean", ".lower", ".upper")) {
  for (col in cols) {
    if (col %in% names(fc_tbl)) {
      fc_tbl[[col]] <- inv_box_cox(fc_tbl[[col]], lambda)
    }
  }
  fc_tbl
}

# =====
# HORIZONTES (número de períodos do teste)
# =====
h_hid <- nrow(hidraulica_teste_2024_25)
h_ole <- nrow(oleos_teste_2024_25)
h_cg <- nrow(CG_teste_2024_25)
h_befn <- nrow(befn_teste_2024_25)

# =====
# 1) Ajustar benchmarks (na série transformada co2_bc)
# =====

```

```

fit_hid_bench <- hidraulica_bc %>%
  model(
    naive = NAIVE(co2_bc),
    snaive = SNAIVE(co2_bc),
    mean = MEAN(co2_bc),
    drift = RW(co2_bc ~ drift())
  )

fit_ole_bench <- oleos_bc %>%
  model(
    naive = NAIVE(co2_bc),
    snaive = SNAIVE(co2_bc),
    mean = MEAN(co2_bc),
    drift = RW(co2_bc ~ drift())
  )

fit_cg_bench <- cg_bc %>%
  model(
    naive = NAIVE(co2_bc),
    snaive = SNAIVE(co2_bc),
    mean = MEAN(co2_bc),
    drift = RW(co2_bc ~ drift())
  )

fit_befn_bench <- befn_bc %>%
  model(
    naive = NAIVE(co2_bc),
    snaive = SNAIVE(co2_bc),
    mean = MEAN(co2_bc),
    drift = RW(co2_bc ~ drift())
  )

# =====
# 2) Forecasts (escala transformada), bias_adjust = TRUE
# =====
fc_hid_raw <- forecast(fit_hid_bench, h = h_hid, bias_adjust = TRUE)
fc_ole_raw <- forecast(fit_ole_bench, h = h_ole, bias_adjust = TRUE)
fc_cg_raw <- forecast(fit_cg_bench, h = h_cg, bias_adjust = TRUE)
fc_befn_raw <- forecast(fit_befn_bench, h = h_befn, bias_adjust = TRUE)

# =====
# 3) Inversão da Box-Cox (.mean / .lower / .upper)
# =====
fc_hid <- inv_bc_across(as_tibble(fc_hid_raw), lambda_hid)
fc_ole <- inv_bc_across(as_tibble(fc_ole_raw), lambda_ole)
fc_cg <- inv_bc_across(as_tibble(fc_cg_raw), lambda_cg)
fc_befn <- inv_bc_across(as_tibble(fc_befn_raw), lambda_befn)

```

```

# =====
# 4) Preparar dataframes finais (observado + previsão)
# =====
fc_hid_df <- fc_hid %>% select(mes, .model, pred = .mean)
fc_ole_df <- fc_ole %>% select(mes, .model, pred = .mean)
fc_cg_df <- fc_cg %>% select(mes, .model, pred = .mean)
fc_befn_df <- fc_befn %>% select(mes, .model, pred = .mean)

hid_join <- hidraulica_teste_2024_25 %>%
  as_tibble() %>%
  select(mes, obs = grupo_hidraulica) %>%
  left_join(fc_hid_df, by = "mes")

ole_join <- oleos_teste_2024_25 %>%
  as_tibble() %>%
  select(mes, obs = grupo_o) %>%
  left_join(fc_ole_df, by = "mes")

cg_join <- cg_teste_2024_25 %>%
  as_tibble() %>%
  select(mes, obs = grupo_cg) %>%
  left_join(fc_cg_df, by = "mes")

befn_join <- befn_teste_2024_25 %>%
  as_tibble() %>%
  select(mes, obs = grupo_befn) %>%
  left_join(fc_befn_df, by = "mes")

```

10) Previsões e geração dos gráficos

```

library(dplyr)
library(tsibble)
library(fable)
library(fabletools)
library(feasts)
library(ggplot2)
library(patchwork)

#-----
# 1) Função genérica para gerar forecast + intervalos
#-----
fc_models <- function(fit, h, lambda) {
  fit |>
    forecast(h = h, bias_adjust = TRUE) |>
    hilo(level = 95) |>
    unpack_hilo(`95%`) |>
    rename(

```

```

.lower = `95%_lower`,
.upper = `95%_upper`
) |>
as_tibble() |>
mutate(
  .mean = inv_box_cox(.mean, lambda),
  .lower = inv_box_cox(.lower, lambda),
  .upper = inv_box_cox(.upper, lambda)
)
}

#-----
# 2) Configuração dos grupos
#-----

groups <- list(
  list(grupo = "Hidráulica",
    lambda = lambda_hid,
    train = hidraulica_treino_2008_2023,
    test = hidraulica_teste_2024_25,
    fits = list(
      arima = fit_hid_arima,
      ets = fit_hid_ets,
      bench = fit_hid_bench
    )),
  list(grupo = "Óleos",
    lambda = lambda_ole,
    train = oleos_treino_2008_2023,
    test = oleos_teste_2024_25,
    fits = list(
      arima = fit_ole_arima,
      ets = fit_ole_ets,
      bench = fit_ole_bench
    )),
  list(grupo = "Carvão&Gás",
    lambda = lambda_cg,
    train = cg_treino_2008_2023,
    test = cg_teste_2024_25,
    fits = list(
      arima = fit_cg_arima,
      ets = fit_cg_ets,
      bench = fit_cg_bench
    )),
  list(grupo = "Renováveis + Nuclear",
    lambda = lambda_befn,
    train = befn_treino_2012_2023,

```

```

    test = befn_teste_2024_25,
    fits = list(
        arima = fit_befn_arima,
        ets = fit_befn_ets,
        bench = fit_befn_bench
    ))
}

#-----
# 3) Cores (tab10) igual ao Python
#-----
model_colors <- c(
    "arima" = "#1f77b4", # azul
    "ets" = "#ff7f0e", # laranja
    "naive" = "#2ca02c", # verde
    "snaive" = "#d62728", # vermelho
    "mean" = "#9467bd", # roxo
    "drift" = "#8c564b" # marrom
)

# Linhas: modelos principais = sólida, benchmarks = tracejada
model_linetype <- c(
    "arima" = "solid",
    "ets" = "solid",
    "naive" = "dashed",
    "snaive" = "dashed",
    "mean" = "dashed",
    "drift" = "dashed"
)

# helper simples pra pegar a coluna do grupo como 'obs'
get_grupo_obs <- function(df) {
    col <- grep("grupo", names(df), value = TRUE)
    if (length(col) == 0) {
        stop("Nenhuma coluna contendo 'grupo' encontrada.")
    }
    df[[col[1]]]
}

plots <- list()

#-----
# 4) Loop para gerar gráficos no estilo Python final
#-----
for (g in groups) {

    grupo <- g$grupo
    lambda <- g$lambda
}

```

```

train <- g$train
test <- g$test
fits <- g$fits
h <- nrow(test)

# Forecasts de cada modelo
fc_arima <- fc_models(fits$arima, h, lambda)
fc_ets <- fc_models(fits$ets, h, lambda)
fc_bench <- fc_models(fits$bench, h, lambda)

fc_all <- bind_rows(fc_arima, fc_ets, fc_bench) |>
  mutate(
    mes = as.Date(mes),
    .model = as.character(.model) # garante compatível com escalas manuais
  )

# Histórico (últimos 36 meses do treino)
hist_obs <- train |>
  mutate(
    mes = as.Date(mes),
    obs = get_grupo_obs(cur_data_all())
  ) |>
  dplyr::slice_tail(n = 36)

# Série real (teste)
real_obs <- test |>
  mutate(
    mes = as.Date(mes),
    obs = get_grupo_obs(cur_data_all())
  )

# -----
# Gráfico Estilo Python
# -----
p <- ggplot() +

  # Histórico em cinza
  geom_line(
    data = hist_obs,
    aes(x = mes, y = obs),
    colour = "grey75",
    linewidth = 1.2,
    alpha = 0.8
  ) +

  # Série REAL (preto grosso)
  geom_line(
    data = real_obs,

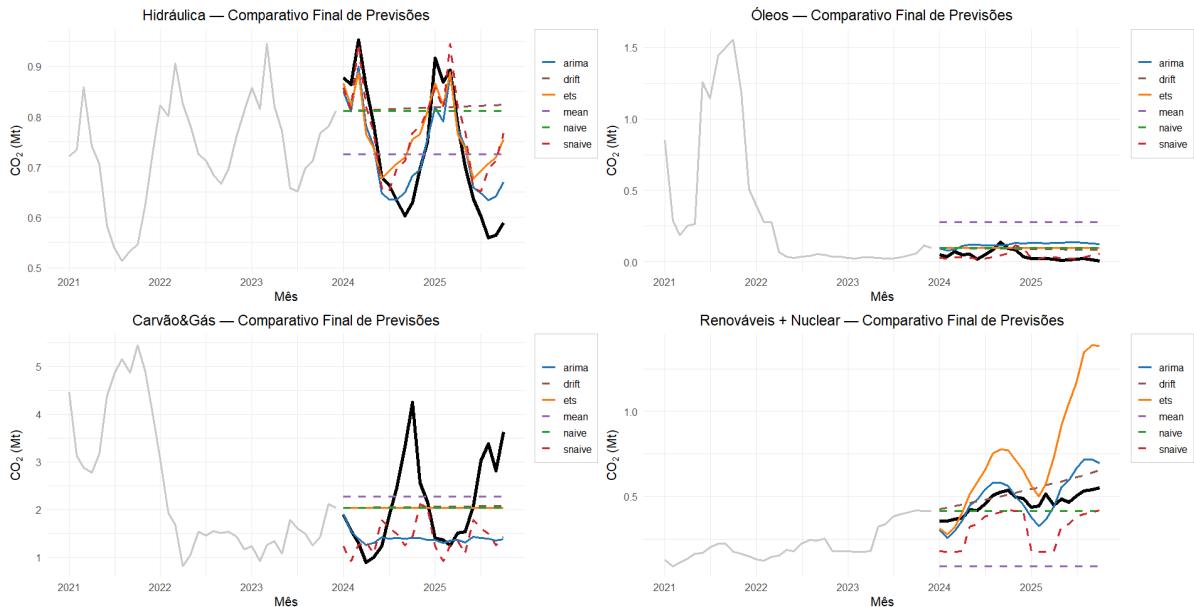
```

```

aes(x = mes, y = obs),
colour = "black",
linewidth = 1.6
) +
# Previsões (todas)
geom_line(
  data = fc_all,
  aes(x = mes, y = .mean, colour = .model, linetype = .model),
  linewidth = 1.1
) +
scale_colour_manual(values = model_colors) +
scale_linetype_manual(values = model_linetype) +
labs(
  title  = paste0(grupo, " — Comparativo Final de Previsões"),
  x      = "Mês",
  y      = expression("CO"[2] * " (Mt)" ),
  colour = "",
  linetype = ""
) +
theme_minimal(base_size = 13) +
theme(
  legend.position    = c(1.02, 1),
  legend.justification = c(0, 1),
  legend.background  = element_rect(fill = "white", colour = "grey80"),
  plot.margin        = margin(5.5, 80, 5.5, 5.5),
  plot.title         = element_text(hjust = 0.5)
)
# guarda usando o nome do grupo
plots[[grupo]] <- p
}

#-----
# 5) Exibir tudo em 2x2
#-----
(p1 <- plots[[1]] | plots[[2]])
(p2 <- plots[[3]] | plots[[4]])
p1 / p2

```



11) Identificação dos melhores modelos para cada série temporal por grupo:

```

calc_metrics <- function(df) {
  df %>%
    summarise(
      RMSE = sqrt(mean((obs - pred)^2, na.rm = TRUE)),
      MAE = mean(abs(obs - pred), na.rm = TRUE),
      MAPE = mean(abs((obs - pred) / obs), na.rm = TRUE) * 100
    )
}

metrics_list <- list()

for(g in groups) {

  grupo <- g$grupo
  lambda <- g$lambda
  train <- g$train
  test <- g$test
  fits <- g$fits
  h <- nrow(test)

  # forecasts já transformados para a escala original
  fc_arima <- fc_models(fits$arima, h, lambda)
  fc_ets <- fc_models(fits$ets, h, lambda)
  fc_bench <- fc_models(fits$bench, h, lambda)

  fc_all <- bind_rows(fc_arima, fc_ets, fc_bench) %>%
    mutate(mes = as.Date(mes))
}

```

```

# Observado (apenas teste)
obs_test <- test %>%
  mutate(
    mes = as.Date(mes),
    obs = dplyr::coalesce(
      across(contains("grupo")) |> unlist()
    )
  ) %>%
  select(mes, obs)

# juntar previsões x observados
joined <- fc_all %>%
  left_join(obs_test, by = "mes") %>%
  rename(pred = .mean)

# calcular métricas por modelo
metrics <- joined %>%
  group_by(.model) %>%
  calc_metrics() %>%
  mutate(grupo = grupo) %>%
  select(grupo, modelo = .model, RMSE, MAE, MAPE)

metrics_list[[grupo]] <- metrics
}

# resultado final
metrics_all <- bind_rows(metrics_list) %>%
  arrange(grupo, RMSE)

print(metrics_all)

> print(metrics_all)
# A tibble: 24 × 5
  grupo     modelo     RMSE     MAE     MAPE
  <chr>     <chr>     <dbl>    <dbl>    <dbl>
1 Carvão&Gás drift     0.917    0.764    41.3
2 Carvão&Gás ets      0.923    0.768    41.1
3 Carvão&Gás naive    0.923    0.768    41.1
4 Carvão&Gás mean     0.934    0.818    48.3
5 Carvão&Gás snaive   1.13     0.805    32.2
6 Carvão&Gás arima    1.16     0.808    30.4
7 Hidráulica arima    0.0521   0.0439   6.20
8 Hidráulica snaive   0.0803   0.0648   9.82
9 Hidráulica ets      0.0828   0.0678  10.3
10 Hidráulica mean    0.124    0.111    15.3

```

best_models <- metrics_all %>%

```

group_by(grupo) %>%
slice_min(RMSE, n = 1, with_ties = FALSE)

print(best_models)

> print(best_models)
# A tibble: 4 × 5
# Groups:   grupo [4]
  grupo             modelo    RMSE     MAE     MAPE
  <chr>            <chr>    <dbl>    <dbl>    <dbl>
1 Carvão&Gás      drift    0.917    0.764    41.3 
2 Hidráulica       arima    0.0521   0.0439   6.20  
3 Renováveis + Nuclear naive   0.0781   0.0682   14.2  
4 Óleos             snaive   0.0348   0.0265  108. 

```

12) Geração dos gráfico com a série temporal hidráulica com o melhor modelo ajustado (ARIMA):

```

library(dplyr)
library(tsibble)
library(fable)
library(fabletools)
library(feasts)
library(ggplot2)

# h = tamanho do conjunto de teste
h_hid <- nrow(hidraulica_teste_2024_25)

# -----
# 1) Forecast ARIMA vencedor + IC 95% (escala original)
# -----
fc_hid <- fit_hid_arima |>
  forecast(h = h_hid, bias_adjust = TRUE) |>
  hilo(95) |>
  unpack_hilo(`95%`) |>
  rename(
    .lower = `95%_lower`,
    .upper = `95%_upper`
  ) |>
  as_tibble() |>
  mutate(
    .mean = inv_box_cox(.mean, lambda_hid),
    .lower = inv_box_cox(.lower, lambda_hid),
    .upper = inv_box_cox(.upper, lambda_hid),
    mes   = as.Date(mes)
  )

```

```

# -----
# 2) Séries observadas (treino + teste)
# -----
obs_full <- dataset_st |>
  select(mes, obs = grupo_hidraulica) |>
  mutate(mes = as.Date(mes))

obs_test <- hidraulica_teste_2024_25 |>
  select(mes, obs = grupo_hidraulica) |>
  mutate(mes = as.Date(mes))

# -----
# FILTRO PARA PLOTAR APENAS A PARTIR DE 2024-01-01
# -----
obs_full <- obs_full %>% filter(mes >= as.Date("2024-01-01"))
obs_test <- obs_test %>% filter(mes >= as.Date("2024-01-01"))
fc_hid <- fc_hid %>% filter(mes >= as.Date("2024-01-01"))

# -----
# 3) Gráfico Estilo Python (igual ao notebook)
# -----
p_hid <- ggplot() +

  # Histórico em cinza claro
  geom_line(
    data = obs_full,
    aes(x = mes, y = obs),
    colour = "grey75",
    linewidth = 1.2,
    alpha = 0.8
  ) +

  # Série REAL de teste (preto grosso)
  geom_line(
    data = obs_test,
    aes(x = mes, y = obs),
    colour = "black",
    linewidth = 1.6
  ) +

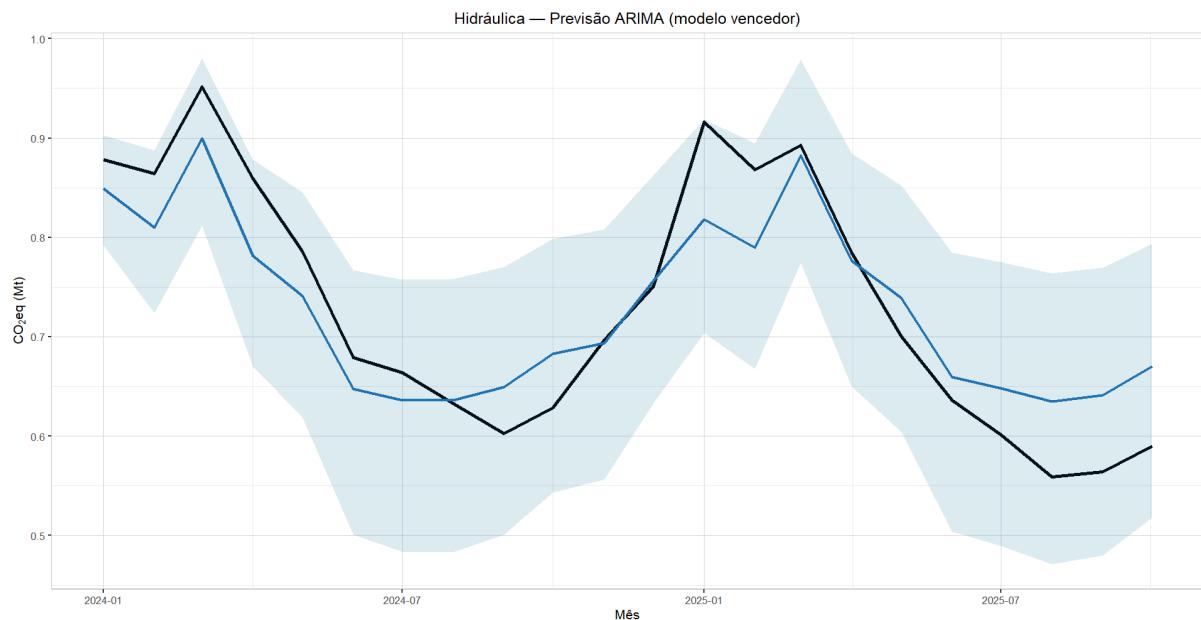
  # Faixa do IC (em azul claro = tab10 azul com alta transparência)
  geom_ribbon(
    data = fc_hid,
    aes(x = mes, ymin = .lower, ymax = .upper),
    fill = tab10[1],
    alpha = 0.15
  ) +

```

```

# Linha de previsão em azul (tab10)
geom_line(
  data = fc_hid,
  aes(x = mes, y = .mean),
  colour  = tab10[1],
  linewidth = 1.3
) +
  labs(
    title  = "Hidráulica — Previsão ARIMA (modelo vencedor)",
    x = "Mês",
    y = expression("CO"[2] * "eq (Mt)"),
    colour  = ""
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    legend.position    = "none",    # igual ao Python
    panel.grid.minor   = element_line(colour = "grey92", linewidth = 0.25),
    panel.grid.major    = element_line(colour = "grey85", linewidth = 0.40)
  )
print(p_hid)

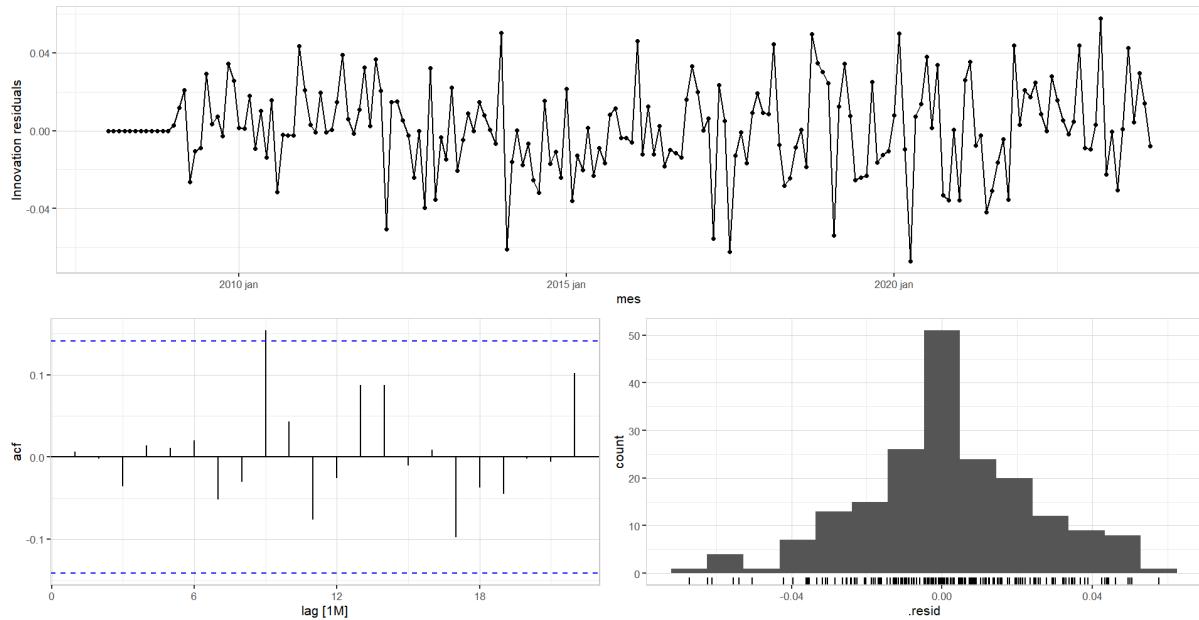
```



13) Diagnóstico dos resíduos e teste Ljung Box

```
fit_hid_arima |> gg_tsresiduals()
```

```
fit_hid_arima |>  
augment() |>  
features(.innov, ljung_box, lag = 24)
```



```
+ features(.innov, ljung_box, lag = 24)  
# A tibble: 1 × 3  
  .model  lb_stat lb_pvalue  
  <chr>    <dbl>     <dbl>  
1 arima     23.2      0.507
```