Exponential smoothing

Print-friendly PDF version

Four series, each from a different part of the Norwegian economy: how many people live here, what a US dollar costs in kroner, what it costs a bank to borrow overnight from Norges Bank, and how many nights people spent in Norwegian hotels. For each one, split into training and test data, fit a few candidate `ETS()` specifications, forecast, and check against the test set what actually helped.

The data

```
install.packages("remotes")
remotes::install_github("holleland/ban430data")
```

```
library(fpp3)
library(ban430data)
data(decomposition)
data(ets_lab)
```

Object	Series	Contents
`population_no`	1	Norway's national population at the start of each quarter, 1997 Q4 → today
`exchange_usdnok`	2	average monthly USD/NOK spot exchange rate, 2010 → today
`policy_rate_no`	3	Norges Bank's key policy rate, monthly, 2010 → today
`hotel_guestnights`	4	monthly guest nights at hotels and similar accommodation, all of Norway, 1986 → today

Cheat sheet: `ETS()` syntax

Term	Syntax
Simple exponential smoothing	`ETS(y ~ error("A") + trend("N") + season("N"))`
Holt's linear trend	`ETS(y ~ error("A") + trend("A") + season("N"))`
Damped trend	`ETS(y ~ error("A") + trend("Ad") + season("N"))`

Term	Syntax
Holt-Winters, additive season	`ETS(y ~ error("A") + trend("A") + season("A"))`
Holt-Winters, multiplicative season	`ETS(y ~ error("M") + trend("A") + season("M"))`
Let `ETS()` choose everything	`ETS(y)`
Fit	`model(name = ETS(...))`
Inspect a fitted model	`report(fit)` or `fit %>% select(name) %>% report()`
Forecast a known window	`forecast(fit, new_data = test)`
Accuracy vs. the actuals	`accuracy(fc, full_data)`

Series 1: Norwegian population

Quarterly, 1997 Q4 → today.

```
train_pop <- population_no %>% filter(quarter < yearquarter("2023 Q3"))
test_pop  <- population_no %>% filter(quarter >= yearquarter("2023 Q3"))
```

Fit a few candidate `ETS()` specifications on `train_pop`, forecast `test_pop`, and compare their accuracy.

Code hint

```
fit_pop <- train_pop %>%
  model(
    SES    = ETS(population ~ error("A") + trend("N")  + season("N")),
    Holt   = ETS(population ~ error("A") + trend("A")  + season("N")),
    Damped = ETS(population ~ error("A") + trend("Ad") + season("N")),
    Auto   = ETS(population)
  )
fit_pop %>% forecast(new_data = test_pop) %>% accuracy(population_no) %>%
  select(.model, RMSE, MASE) %>% arrange(RMSE)
```

Print `report()` for the automatically selected model (`fit_pop %>% select(Auto) %>% report()`) and read off which components it actually contains.

Q: Which components — trend, damping, season — actually earned their place on the test set, and which just added complexity without improving the forecast?

Series 2: the USD/NOK exchange rate

Monthly, 2010 → today.

```
train_fx <- exchange_usdnok %>% filter(month < yearmonth("2025-09"))
test_fx  <- exchange_usdnok %>% filter(month >= yearmonth("2025-09"))
```

Fit a few candidate `ETS()` specifications on `train_fx`, forecast `test_fx`, and compare their accuracy.

Code hint

```
fit_fx <- train_fx %>%
  model(
    SES    = ETS(rate ~ error("A") + trend("N")  + season("N")),
    Holt   = ETS(rate ~ error("A") + trend("A")  + season("N")),
    Auto   = ETS(rate),
    Naive  = NAIVE(rate)
  )
fit_fx %>% forecast(new_data = test_fx) %>% accuracy(exchange_usdnok) %>%
  select(.model, RMSE, MASE) %>% arrange(RMSE)
```

Print `report()` for the automatically selected model and for `Naive`'s closest competitor — what do their fitted parameters have in common?

Q: Did adding a trend component help here, hurt, or make no real difference? What does that tell you about which component actually matters for this series?

Series 3: Norges Bank's key policy rate

Monthly. Train on data only through the end of 2023, and forecast the two years after.

```
train_rate <- policy_rate_no %>% filter(month < yearmonth("2024-01"))
test_rate  <- policy_rate_no %>% filter(month >= yearmonth("2024-01"), month < yearmonth(
```

Fit a couple of candidate `ETS()` specifications on `train_rate` and forecast `test_rate`.

Code hint

```
fit_rate <- train_rate %>%
  model(
    Holt   = ETS(rate ~ error("A") + trend("A")  + season("N")),
    Damped = ETS(rate ~ error("A") + trend("Ad") + season("N"))
  )
fc_rate <- fit_rate %>% forecast(new_data = test_rate)
fc_rate %>% accuracy(policy_rate_no) %>% select(.model, RMSE, MASE)
```

Print `report()` for both models, then plot both forecasts against what actually happened:

```
fc_rate %>% autoplot(policy_rate_no %>% filter(month >= yearmonth("2022-01")), level = NU
```

Q: By how many percentage points do the two forecasts disagree two years out? What did the rate actually do? Which single component — trend or damping — is responsible for almost all of that gap?

Series 4: hotel guest nights

Monthly, 1986 → today. This is the one from the decomposition lecture.

```
hotel <- hotel_guestnights %>% rename(guests = `Guest nights`)

train_hotel <- hotel %>% filter(yearmonth < yearmonth("2024-07"))
test_hotel  <- hotel %>% filter(yearmonth >= yearmonth("2024-07"))
```

Fit a few candidate `ETS()` specifications on `train_hotel`, forecast `test_hotel`, and compare their accuracy.

Code hint

```
fit_hotel <- train_hotel %>%
  model(
    SES      = ETS(guests ~ error("A") + trend("N")  + season("N")),
    HW_add   = ETS(guests ~ error("A") + trend("A")  + season("A")),
    HW_mult  = ETS(guests ~ error("M") + trend("A")  + season("M")),
    Auto     = ETS(guests)
  )
fit_hotel %>% forecast(new_data = test_hotel) %>% accuracy(hotel) %>%
  select(.model, RMSE, MASE) %>% arrange(RMSE)
```

Print `report()` for the automatically selected model.

Q: Adding a season component clearly matters here — but does it matter *how* you add it? Look back at the full series plot: is there anything around early 2020 that might explain why an additive vs. multiplicative season makes such a difference?

Wrap-up discussion

- For each of the four series: which ETS components (trend, damping, season, and the choice between additive/multiplicative) actually mattered, and which ones didn't move the needle? Is there a series where none of the extra components beat simple exponential smoothing?
- In how many of your four series did `ETS()`'s own automatic pick actually win on the test set? Would you trust automatic selection for a model you had to actually deploy — or would you always want to check it against a held-out test set, and read its `report()`, first?
- The policy-rate series trains right up to the peak of a rapid change. Real deployed forecasting systems don't get to choose such a convenient cutoff — how would you *know*, in practice, that you were sitting at a turning point where an undamped trend was about to go badly wrong?
- Which of today's four series would you be comfortable handing a fully automatic `ETS()` pipeline for, with no human checking the forecast before it's used? Which would you not — and what would you want a human to check first?