Project: The Forecasting Tourism 2010 Competition

EM1415

Marco Solari, 875475

Table of contents

1 Setup and Data Loading

1.1 Setup

```
knitr::opts_chunk$set(
  echo = T,
  dev = "cairo_pdf"
)

libraries_list <- c(
  "tidyverse",
  "fpp3",
  "ggthemes"

)

lapply(
  X = libraries_list,
  FUN = require,
  character.only = TRUE
)
```

1.2 Loading Data

```
data_main <- readr::read_csv(
  "Data/tourism_data.csv",
  show_col_types = F
)
```

```
data_main %>%
  dim
```

```
[1]  43 518
```

```
data_main %>%
  is.na() %>%
  sum
```

```
[1] 11668
```

We are missing 52.38% of the observations.

1.3 Creating **tsibble**

```
tourism_full <- data_main %>%
  mutate(
    Year = 1965:2007
  ) %>%
  as_tsibble(
    index = Year
  )
```

`tmelt` (Table 1) contains the *melted* data frame, which allows us to apply the tidy forecasting workflow to all 518 time series at once. Its main variables are:

- `index`: `Year`, as in the original data frame.
- `key`: `Identifier`, a new categorical variable allowing us to transform the data frame to the tidy format; it consists in a set of *labels* that identify each time series.
- `value`: the Y_t value for each time series.

```
tmelt ← reshape2::melt(
  tourism_full,
  id = "Year",
  variable.name = "Identifier",
  value.name = "Value"
  ) %>%
  as_tsibble(
    index = "Year",
    key = "Identifier"
  )
```

```
tmelt %>%
  dim()
```

```
[1] 22274     3
```

Table 1: Excerpt of melted `tsibble` containing all time series.

Year	Identifier	Value
1998	Y518	1504
1999	Y518	1343
2000	Y518	1583
2001	Y518	1772
2002	Y518	1676
2003	Y518	1423
2004	Y518	1751
2005	Y518	1385
2006	Y518	1229
2007	Y518	1102

2 Assignment

2.1 Full Plot

In all the subsequent plots, a log_{10} transformation has been employed exclusively for representing the time series on the y-axis. This adjustment becomes necessary since the original data range[1] does not permit a clear and meaningful visualization of the series when plotted together.

[1] 10^9, shown in Table 2.

```
tmelt %>%
  reframe(
    "Range" =  range(
      Value,
      na.rm = T,
      finite = T
      )
    ) %>%
  mutate(
    "Y" = c(
```

```
      "min",
      "max"
      ),
    .before = "Range"
    )
```

Y	Range
min	5.810000e-02
max	5.200294e+07

Table 2: Range of Tourism Time Series

2.1.1 Everything, Everywhere, All At Once

Plot all the series (an advanced data visualization tool is recommended) - what type of components are visible? Are the series similar or different? Check for problems such as missing values and possible errors.

```
tmelt %>%
  ggplot(
    aes(
      x = Year,
      y = Value,
      colour = Identifier,
      group = Identifier
      )
    ) +
  geom_line(
    alpha = .8
  ) +
  scale_y_log10() +
  scale_color_viridis_d(
    option = "cividis"
  ) +
  labs(
    title = "Tourism Time Series: Everything All At Once",
    y = expression(log[10](Value))
  ) +
  theme(
    legend.position = "none",
    plot.margin = margin(
      1,
      1,
      3,
      1
    )
  )
```

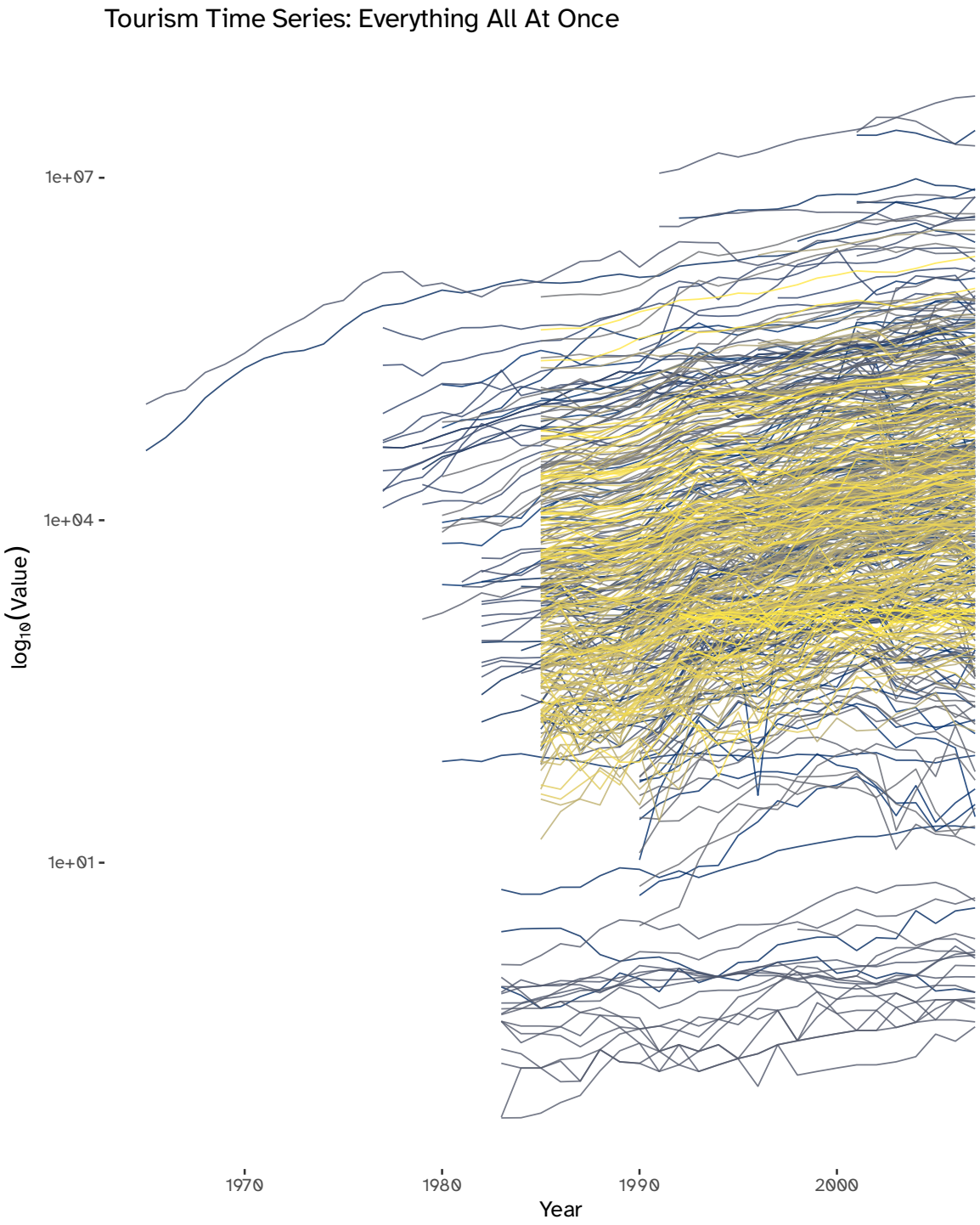

Figure 1: Printing a legend for 518 different series is not a viable option. However, color has been used only to differentiate the series and does not contain further information. Plotting the y-axis variable on the log scale was made necessary by the huge variation in the series values.

While plotting all 518 series simultaneously may hinder the clear identification of specific details, a distinct overall upward trend is discernible. Additionally, noteworthy outliers can be spotted, warranting further investigation. Furthermore, indications of cyclicality in certain series can be observed.

A check for NAs has already been made while loading data (Section 1.2) and it showed the presence of a large number of missing values, corresponding to 52.38% of all observations. This can be attributed to the difference in the initial timestamps of the series. We can categorize these series based on their respective starting years, indicating that an alternative visualization approach could be effectively implemented through this grouping method (Figure 2).

```
tmelt %>%
  summarise(
    "Available Observations" = sum(
      !is.na(Value)
      )
  )
```

Year	Available Observations
1965	2
1966	2
1967	2
1968	2
1969	2
1970	2
1971	2
1972	2
1973	2
1974	2
1975	2
1976	2
1977	13
1978	13
1979	18
1980	29
1981	31
1982	47
1983	66
1984	70
1985	336
1986	342
1987	342
1988	342
1989	342
1990	391
1991	406
1992	419
1993	419
1994	419
1995	419
1996	489
1997	494
1998	503
1999	503
2000	503
2001	518
2002	518
2003	518
2004	518
2005	518
2006	518
2007	518

Table 3: Missing observation by year: the presence of missing observations is related to the scarcity of long-run time series.

The set of complete time series starts in 2001.

2.1.2 Plotting Series By Starting Year

Arranging the series chronologically by their starting year not only aids in evaluating their variability but also amplifies clarity.

This grouping stresses the already noted upward trend, except for most series kickstarting in 2001 (top-left subplot of Figure 2). Another notable group of outliers can be seen in the subplot titled 18[2]: in this group, we can spot a cluster of series in which the upward trend is inverted.

[2] Time series starting in 1989.

```
tmelt %>%
  group_by(
    Identifier
    ) %>%
  mutate(
    series_length = 43 -  Value %>% is.na %>% sum
    ) %>%
  ungroup() %>%
  arrange(
    desc(
      series_length
      )
    ) %>%
  mutate(
    series_length = as_factor(series_length)
    ) %>%
  ggplot(
    aes(
      x = Year
      )
  ) +
  facet_wrap(
    ~series_length,
    nrow = 6,
    ncol = 3,
    scales = "free"
  ) +
  geom_line(
    aes(
      y = Value,
      color = Identifier
    )
  ) +
  labs(
    title = "Tourism Time Series By Starting Year",
    y = expression(log[10](Value))
  ) +
  scale_y_log10() +
  scale_color_viridis_d(
    option = "cividis"
  ) +
  theme(
    legend.position = "none"
  )
```

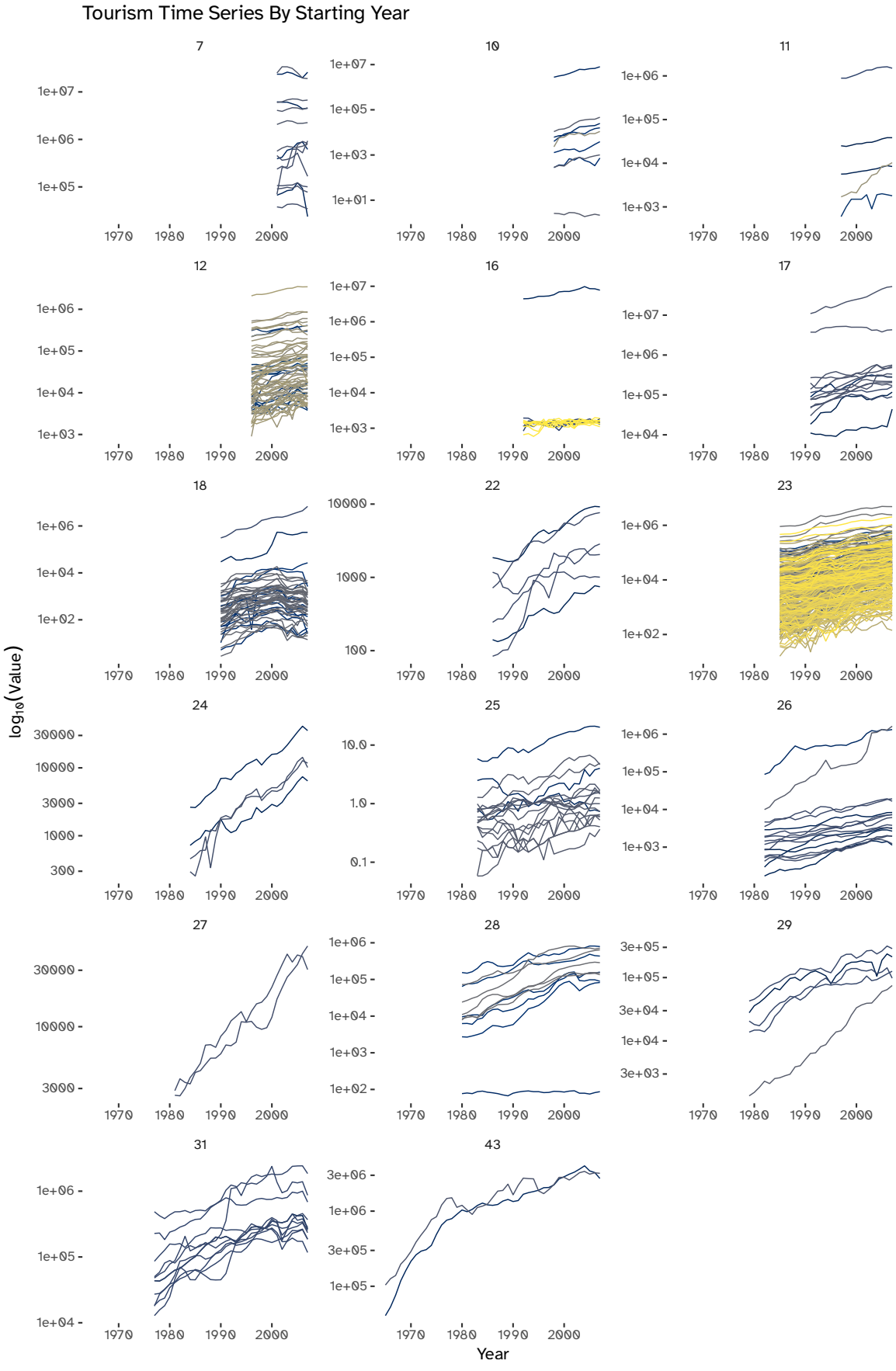

Figure 2: All series have been grouped by starting year and plotted to achieve more clarity. Each subtitle represents the number of periods for each subset. The same color mapping of Figure 1 has been used to differentiate the series.

2.2 Creating Validation Set

> *Partition the series into training and validation, so that the last 4 years are in the validation period for each series. What is the logic of such a partitioning? What is the disadvantage?*

```
train ← tmelt %>%
  filter(
    Year < 2004
    )
validation ← tmelt %>%
  filter(
    Year ⩾ 2004
    )
```

```
validation %>%
  head(8)
```

Year	Identifier	Value
2004	Y1	32613.50
2005	Y1	36053.17
2006	Y1	38472.75
2007	Y1	38420.89
2004	Y2	450569.00
2005	Y2	421513.00
2006	Y2	426166.00
2007	Y2	265729.00

The logic behind partitioning the series into a *training* and *validation* set is to *estimate the forecasting error*: we can train a model or apply a filter to the train set and use it to assess its performance with out-of-sample data. The main disadvantage of this approach is that we are not using all the information available to train our model; moreover, we are not computing *true forecasts*, therefore the accuracy measures from the residuals will be smaller.

2.3 Naïve Forecasts

> *Generate naïve forecasts for all series for the validation period. For each series, create forecasts with horizons of 1, 2, 3, and 4 years ahead (F_{t+1}, F_{t+2}, F_{t+3}, and F_{t+4}).*

We can produce the forecasts by applying Equation 1:

$$y_{T+h \mid T} = y_T \tag{1}$$

First of all, initializing a naïve model will allow us to use `R` to compute both point forecasts and prediction intervals:

```
naive_model ← train %>%
  na.omit() %>%
  model(
    NAIVE(
      Value
      )
    )
```

`naive_model` will contain a `mable` for all the series, to be used to compute both *training* and *validation* errors.

To obtain F_{t+1}, F_{t+2}, F_{t+3}, and F_{t+4}:

```
naive_fc ←
  naive_model %>%
  forecast(
    h = 4
  )
```

```
naive_fc %>%
  tail(20)
```

Identifier	.model	Year	Value	.mean
Y514	NAIVE(Value)	2004	N(1603, 47883)	1603
Y514	NAIVE(Value)	2005	N(1603, 95767)	1603
Y514	NAIVE(Value)	2006	N(1603, 143650)	1603
Y514	NAIVE(Value)	2007	N(1603, 191533)	1603
Y515	NAIVE(Value)	2004	N(1655, 138231)	1655
Y515	NAIVE(Value)	2005	N(1655, 276462)	1655
Y515	NAIVE(Value)	2006	N(1655, 414692)	1655
Y515	NAIVE(Value)	2007	N(1655, 552923)	1655
Y516	NAIVE(Value)	2004	N(1266, 63737)	1266
Y516	NAIVE(Value)	2005	N(1266, 127474)	1266
Y516	NAIVE(Value)	2006	N(1266, 191211)	1266
Y516	NAIVE(Value)	2007	N(1266, 254948)	1266
Y517	NAIVE(Value)	2004	N(1864, 205324)	1864
Y517	NAIVE(Value)	2005	N(1864, 410647)	1864
Y517	NAIVE(Value)	2006	N(1864, 615971)	1864
Y517	NAIVE(Value)	2007	N(1864, 821295)	1864
Y518	NAIVE(Value)	2004	N(1423, 19980)	1423
Y518	NAIVE(Value)	2005	N(1423, 39961)	1423
Y518	NAIVE(Value)	2006	N(1423, 59941)	1423
Y518	NAIVE(Value)	2007	N(1423, 79921)	1423

2.4 Choosing Measures

> *Which measures are suitable if we plan to combine the results for the 518 series? Consider MAE, Average error, MAPE and RMSE.*

When combining forecasting results for multiple time series it is crucial to account for the scale and potential variations across the series. The choice of measures can impact the overall assessment of forecasting accuracy.

The Mean Absolute Error (MAE) is a suitable measure to quantify the average absolute errors across all series without considering the direction of the errors. It provides a straightforward indication of the average magnitude of forecasting errors.

The Average Error[3] can complement the MAE by providing a simple measure of the overall bias in the forecasting. However, it does not consider the direction of errors and might not be suitable if positive and negative errors can cancel each other out.

The Mean Absolute Percentage Error (MAPE) is suitable for evaluating the forecasting accuracy in percentage terms, which can be particularly useful when dealing with many series of different scales. However, it is sensitive to series with small actual values.

Last but not least, the Root Mean Squared Error (RMSE) is suitable to penalize larger errors more heavily[4]. It provides a balance between considering both large and small errors; like MAE, it doesn't consider the direction of errors.

Having a very wide range of values, as seen in Table 2, the MAPE is the candidate for the most useful error measure among the ones listed[5], to ensure consistency when evaluating the forecasting error across different scaled series.

[3] Defined as the mean of all individual errors.

[4] Outliers might therefore skew its measurement.

[5] Although scaled errors are not considered and could address the issue of evaluating performance of series having a wide range, while not being sensitive to small values.

2.5 Computing MAPE

For each series, compute MAPE of the naive forecasts once for the training period and once for the validation period.

2.5.1 Training Period:

```
errors_training ← naive_model %>%
  accuracy()
```

This is the training MAPE for the first 10 series:

```
errors_training %>%
  select(
    Identifier,
    .type,
    MAPE
  ) %>%
  head(10)
```

Identifier	.type	MAPE
Y1	Training	4.104646
Y2	Training	10.392034
Y3	Training	12.815980
Y4	Training	15.022398
Y5	Training	6.771865
Y6	Training	6.441830
Y7	Training	5.526134
Y8	Training	3.936621
Y9	Training	10.516378
Y10	Training	10.321529

2.5.2 Validation Period:

```
errors_validation ←
  accuracy(
    naive_fc,
    validation
  )
```

This is the validation MAPE for the first 10 series:

```
errors_validation %>%
   select(
    Identifier,
    .type,
    MAPE
  ) %>%
  head(10)
```

Identifier	.type	MAPE
Y1	Test	16.58727
Y2	Test	22.12467

Identifier	.type	MAPE
Y3	Test	29.32627
Y4	Test	20.51794
Y5	Test	24.44290
Y6	Test	15.28149
Y7	Test	17.13218
Y8	Test	12.30457
Y9	Test	13.39204
Y10	Test	16.17211

2.5.3 Comparison Table

```
MAPE_comparison ← bind_rows(
errors_training[1:10, ] %>%
  select(MAPE) %>%
  round(., digits = 2) %>%
  t() %>%
  as_tibble() %>%
  mutate(
    Set = "Training",
    .before = V1
  ) %>%
  tail(),
errors_validation[1:10, ] %>%
  select(MAPE) %>%
  round(., digits = 2) %>%
  t() %>%
  as_tibble() %>%
  mutate(
    Set = "Validation"
  )
)

colnames(MAPE_comparison) ← c("Set", errors_training$Identifier %>%
 ↪  as.character() %>% unique %>% head(10))

MAPE_comparison
```

Set	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10
Training	4.10	10.39	12.82	15.02	6.77	6.44	5.53	3.94	10.52	10.32
Validation	16.59	22.12	29.33	20.52	24.44	15.28	17.13	12.30	13.39	16.17

Table 8: Naïve forecasts training and validation MAPEs for the first 10 time series.

2.6 Computing MASE

> *The performance measure used in the competition is Mean Absolute Scaled Error (MASE). Explain the advantage of MASE and compute the training and validation MASE for the naive forecasts.*

The Mean Absolute Scaled Error (MASE) serves as a robust performance metric thanks to its scale-independence. This quality makes it well-suited for the comparative assessment of forecast accuracy across diverse time series characterized by differing scales and magnitudes, such as in this dataset, accounting for the inherent scale differences among them by scaling the errors based on the training MAE from a simple forecast method.

It is an alternative to *percentage errors* such as the MAPE. For a non-seasonal time series, a useful way to define a scaled error uses naïve forecasts: because the numerator and denominator both involve values on the scale of the original data, scaled errors are independent of the scale of the data.

2.6.1 Training Period:

```
errors_training %>%
  select(
    Identifier,
    .type,
    MASE
  ) %>%
  head(10)
```

Identifier	.type	MASE
Y1	Training	1
Y2	Training	1
Y3	Training	1
Y4	Training	1
Y5	Training	1
Y6	Training	1
Y7	Training	1
Y8	Training	1
Y9	Training	1
Y10	Training	1

Table 9: Training MASE for the first 10 series.

Since MASE indicates the effectiveness of the forecasting algorithm for a naïve forecast, a value greater than one 1 indicates that the algorithm is performing poorly compared to the naïve forecast, and vice-versa: hence, since we have been computing the naïve MASE of in-sample data, it is equal to 1 for all time series in our training dataset.

2.6.2 Validation Period:

```
errors_validation ←
  accuracy(
    naive_fc,
    tmelt
  )
```

This is the validation MASE for the first 10 series:

```
errors_validation %>%
   select(
    Identifier,
    .type,
    MASE
  ) %>%
  head(10)
```

Identifier	.type	MASE
Y1	Test	5.474326

Table 10: Validation MASE for the first 10 series.

Identifier	.type	MASE
Y2	Test	4.476840
Y3	Test	3.740707
Y4	Test	2.310849
Y5	Test	10.320584
Y6	Test	4.757987
Y7	Test	4.738439
Y8	Test	4.025292
Y9	Test	2.812082
Y10	Test	3.957335

2.6.3 Comparison Table:

```
MASE_comparison ← bind_rows(
errors_training[1:10, ] %>%
  select(MASE) %>%
  round(., digits = 2) %>%
  t() %>%
  as_tibble() %>%
  mutate(
    Set = "Training",
    .before = V1
  ) %>%
  tail(),
errors_validation[1:10, ] %>%
  select(MASE) %>%
  round(., digits = 2) %>%
  t() %>%
  as_tibble() %>%
  mutate(
    Set = "Validation"
  )
)

colnames(MASE_comparison) ← c("Set", errors_training$Identifier %>%
 ↪  as.character() %>% unique %>% head(10))

MASE_comparison
```

Set	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10
Training	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Validation	5.47	4.48	3.74	2.31	10.32	4.76	4.74	4.03	2.81	3.96

Table 11: Naïve forecasts training and validation MASEs for the first 10 time series.

2.7 MAPE & MASE Pairs

Create a scatter plot of the MAPE pairs, with the training MAPE on the x-axis and the validation MAPE on the y-axis. Create a similar scatter plot for the MASE pairs. Now examine both plots. What do we learn? How does performance differ between the training and validation periods? How does performance range across series?

```
ggplot(
  data = MAPE_pairs,
  aes(
```

```
    x = Training_MAPE,
    y = Validation_MAPE,
    color = Series_Identifier
  ),
) + geom_point(
  alpha = .8
) +
  geom_rug() +
  geom_abline(
    slope = 1,
    color = "grey80",
    linetype = "dashed"
  ) +
  labs(
    title = "Training and Validation MAPE pairs, colored by series",
    x = "Training MAPE",
    y = "Validation MAPE"
  ) +
  scale_color_viridis_d(
    option = "cividis"
  ) +
  theme_updater
```

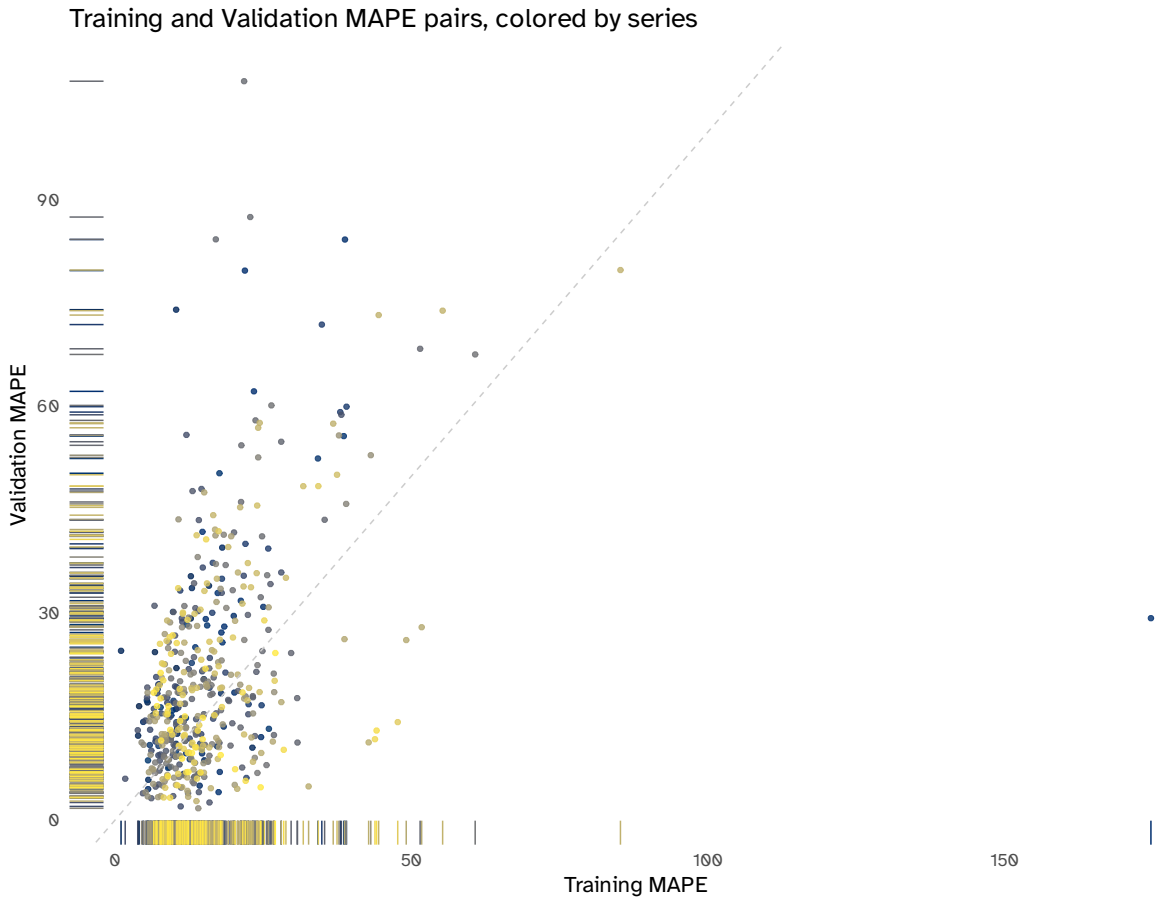

Figure 3: Scatterplot of training and validation MAPE pairs: on both axis, the distribution of values. The time series have been colored using the same mapping seen in Figure 1.

```
ggplot(
  data =
    MASE_pairs,
```

```
  aes(
    x = Training_MASE,
    y = Validation_MASE,
  color = Series_Identifier
  ),
) + geom_point(
) +
  geom_rug() +
  labs(
    title = "Training and Validation MASE pairs, colored by series",
    x = "Training MASE",
    y = "Validation MASE"
  ) +
  scale_color_viridis_d(
    option = "cividis"
  ) +
  theme_updater
```

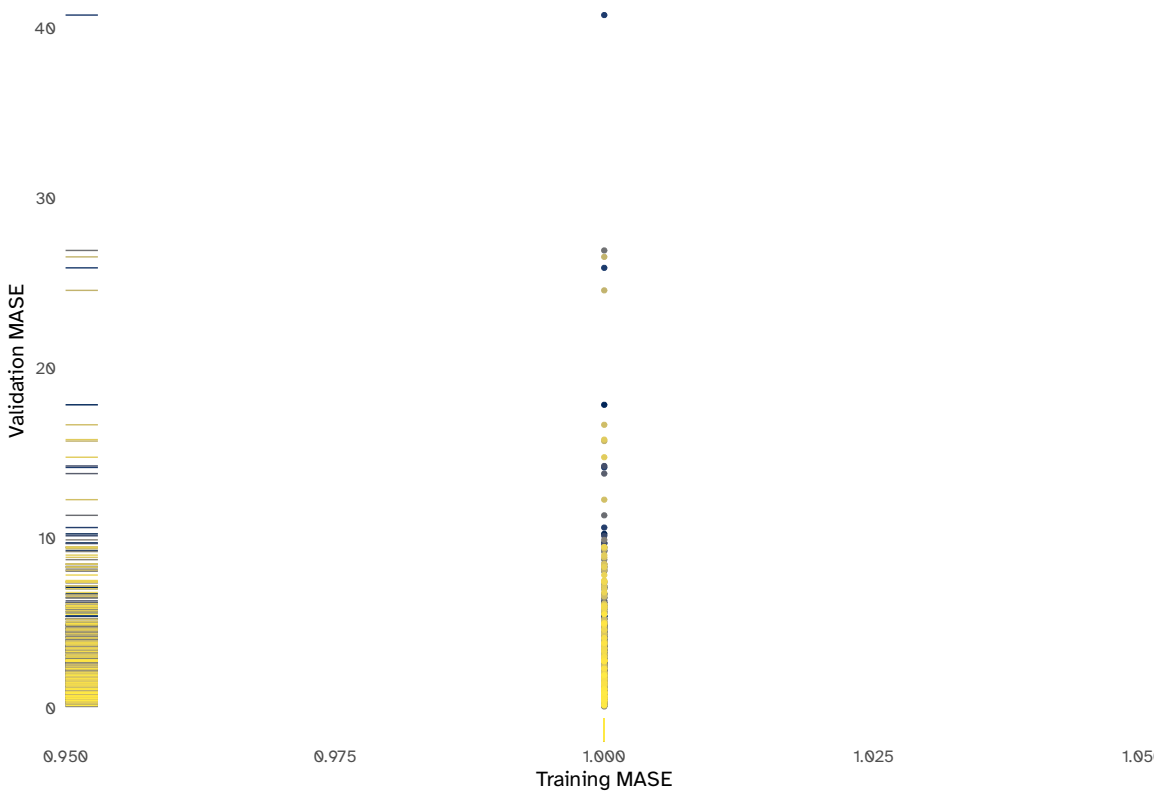

Figure 4: Scatterplot of training and validation MASE pairs: on both axis, the distribution of values. The time series have been colored using the same mapping seen in Figure 1.

We can add some jitter to better visualize the points:

```
ggplot(
  data =
    MASE_pairs,
  aes(
    x = Training_MASE,
    y = Validation_MASE,
  color = Series_Identifier
  ),
```

```
) + geom_jitter() +
  geom_rug() +
  labs(
    title = "Training and Validation MASE pairs, colored by series",
    x = "Training MASE",
    y = "Validation MASE"
  ) +
  scale_color_viridis_d(
    option = "cividis"
  ) +
  ggthemes::theme_tufte(
    base_size = 16,
    base_family = custom_typeface,
    ticks = F
  ) +
  theme(
    legend.position = "none",
    axis.text.x = element_blank()
  )
```

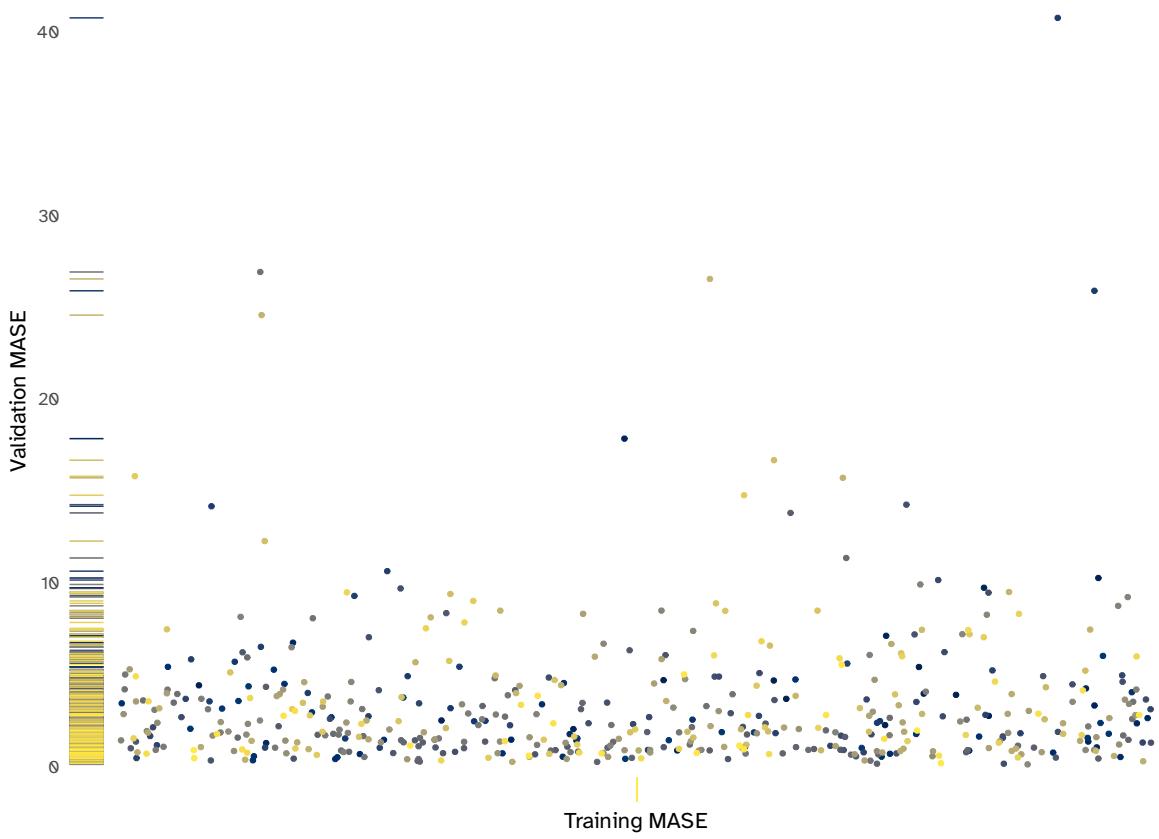

Figure 5: Jittered scatterplot of all MASE pairs, as in Figure 4, with jittering.

2.8 Ensemble Methods

The competition winner, Lee Baker, used an ensemble of three methods:

- *Naive forecasts multiplied by a constant trend*[6].
- Linear regression.
- Exponentially-weighted linear regression.

[6] Global/local trend: "globally tourism has grown at a rate of 6% annually."

a. *Write the exact formula used for generating the first method, in the form* $F_{t+k} = ...$, where $k = 1, 2, 3, 4$),

Starting from Equation 1 and introducing a constant trend multiplier, denoted as k[7], we can express the equation for naïve forecasts multiplied by this constant trend as follows:

[7] In the case of an annual growth rate of 6%, $k = 0.06$.

$$y_{T+h \mid T} = y_T \times (1 + k)^h \tag{2}$$

h, as usual, is the number of years in the future we want to forecast. This formulation represents the extension of naïve forecasts, incorporating a constant trend multiplier, thereby accounting for a consistent trend in the time series data.

b. *What is the rationale behind multiplying the naive forecasts by a constant?*[8]

[8] Hint: think empirical and domain knowledge.

The rationale would be to enhance the performance of naïve forecasts, a benchmark method, to apply them with real data that show a pronounced trended behaviour. While some other approaches can describe an average change over time[9], empirical and domain knowledge indicate that, in certain instances, the temporal evolution can be adequately represented by a constant: consequently, mixing a *random walk* behaviour with a constant trend aims at modeling the data more closely.

[9] E.g.: the *drift method*.

c. *What should be the dependent variable and the predictors in a linear regression model for this data? Explain.*

In the absence of any predictors in the data, a standard strategy involves formulating an $AR(p)$ model to capture dependencies of $Y_{T+h|T}$ on its past:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{3}$$

However, the time series are not stationary: this model would necessitate transformations to remove the trend, resulting in the loss of essential information crucial for accurate forecasting. Consequently, an alternative approach is to devise a simple linear model incorporating an independent trend variable:

$$y_t = \beta_0 + \beta_1 t + \epsilon_t \tag{4}$$

This latter methodology appears to align more consistently with the overall information available about the series and the models employed by the contest winner.

d. *Fit the linear regression model to the first five series and compute forecast errors for the validation period.*

```
ts_list ← c("Y1", "Y2", "Y3", "Y4", "Y5")

ts_trend_fit ← tmelt %>%
  filter(
    Identifier %in% ts_list) %>%
  model(
    TSLM(
      Value ~ trend()
    )
  )
```

```
ts_trend_fit %>%
  report() %>%
  select(
    Identifier,
```

```
    p_value,
    r_squared,
    AICc
  )
```

Identifier	p_value	r_squared	AICc
Y1	0.0000006	0.9438223	164.7734
Y2	0.0000000	0.9089079	658.0936
Y3	0.0000000	0.8381282	587.1175
Y4	0.0000018	0.7913794	331.4518
Y5	0.0054119	0.4128168	303.2326

Table 12: Selection of resulting selection criteria scores for the Time Series linear models, fitted for the first 5 series over the validation period.

```
ts_trend_fit %>%
  forecast(
    validation
  )  %>%
  accuracy(
    tmelt
    ) %>%
  select(
    Identifier,
    MAPE,
    MASE
  )
```

Identifier	MAPE	MASE
Y1	2.778091	0.8939776
Y2	22.531215	4.1514711
Y3	24.395110	2.7068572
Y4	8.212244	0.8510743
Y5	34.587229	10.5721022

Table 13: Selection of accuracy measures for the TSLM.

e. *Before choosing a linear regression, the winner described the following process:*

"I examined fitting a polynomial line to the data and using the line to predict future values. I tried using first through fifth-order polynomials to find that the lowest MASE was obtained using a first-order polynomial (simple regression line). This best-fit line was used to predict future values. I also kept the R^2 value of the fit for use in blending the results of the prediction."

What are two flaws in this approach?

A polynomial fitting is useful when the relationship between predictor and response is not linear: in such instances, the Ordinary Least Squares (OLS) estimator still can be used allowing the derivation of a fitting capable of accommodating non-linear relationships, while preserving linearity in the parameters. Adding polynomial terms at random leads to overfitting in most cases, leading to the excess variance that does not generalize well to out-of-sample data. Moreover, considering our context of a straightforward trended linear model, the incorporation of higher-order polynomial trends may compromise interpretability.

While the R^2 represents the portion of explained variability by the model, it is based on the *sum of squares*, which is inconsistent with the given error statistic, the MASE; as written in Section 2.4, it penalizes larger errors more heavily, while the MASE considers the average magnitude of forecasting errors.

Last but not least, assuming a simple univariate trended linear model, it is also equivalent to the squared correlation between the dependent variable and the predictor[10], so it is not a proper weight to combine forecasts.

[10] In a multivariate model, the R^2 it corresponds to the squared correlation between y_t and $\hat{y}_t$, so this point still stands.

f. *If we were to consider exponential smoothing, what particular type(s) of exponential smoothing are reasonable candidates?*

We could implement two different exponential smoothings that include a Trend component. Viable candidates are Holt's linear method (A, N) and Additive damped trend method (A_d, N): their performance should be tested against both the validation and test sets to choose the most appropriate.

g. *The winner concludes with possible improvements one being* an investigation into how to come up with a blending ensemble method that doesn't use much manual tweaking would also be of benefit. *Can you suggest methods or an approach that would lead to easier automation of the ensemble step?*

Automation of model selection and amalgamation could be achieved through the adoption of a "loss function"-centered approach, constructed upon the ensemble amalgamation of diverse methods. The allocation of weights to each method's forecast could be fine-tuned by assessing their performance within the parameter space on the validation set, employing a weighted least squares method.

To find these hyper-parameter estimates numerically, then, we could use different approaches, ranging from a linear combination of weights to a more complex deep neural network, capable of receiving a matrix containing forecasts from various ensemble methods as input, which would enable the learning of optimal weights through backpropagation, facilitated by a properly calibrated activation function. In both cases, learning the model would be equivalent to setting the proper mix of forecasts.

In other words, a machine learning based approach might prove helpful in choosing the best combination of forecast, with the aim of minimising the MASE of the validation set; nevertheless, caution should be used as it would introduce another potential source of overfitting.

However:

> "*While there has been considerable research on using weighted averages, or some other more complicated combination approach, using a simple average has proven hard to beat.*"[11].

This simple average could work as a benchmark to evaluate the amalgamation method used to build the ensemble forecast.

[11] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.

h. *The competition focused on minimizing the average MAPE of the next four values across all 518 series. How does this goal differ from goals encountered in practice when considering tourism demand? Which steps in the forecasting process would likely be different in a real-life tourism forecasting scenario?*

The main difference in a real-world scenario is the complete absence of *prediction intervals* and any interest in quantifying *uncertainty*.

As cleverly stated by the contest winner:

> "*If the test set includes data from 2008–2009, I'm speculating that depressed tourism numbers as a result of the global economic recession could have caused a significant difference in the trends.*"

This stresses the importance of understanding the underlying assumptions and the power of using *models* instead of *filters*, or in combinations with them.

Gaming a specific statistic is not a viable tool for general or specific purpose forecasting in a real-world scenario: on the one hand because we might be interested in a different question than the one that a minimized MAPE is answering, or in forecasting for longer time windows; on the other hand, relying solely on the point forecast offers a restricted perspective, neglecting the broader landscape of characteristics such as volatility that may be of interest. For instance, the necessity of computing prediction intervals requires a tailored approach. The amalgamation method applied for the ensemble forecast needs to be taken into account, coupled with the specifics of the computational tools enlisted in implementing the solution[12]. Merely optimizing a singular statistic fails to account for the multi-faceted demands of comprehensive forecasting.

A last remark is that in a real-world scenario, a forecaster is dealing with the unavailability of such a constant feedback loop, as the contest winner had, of submitting his results and observing a test MAPE on the Kaggle leaderboard. It is often the case that the chances to submit forecasts or act on them face time constraints and the absolute absence of this kind of feedback loop, and this implies the need for more robust planning and instruments, built upon the impossibility of observing the out-of-sample data and any related statistic.

[12] One of the solutions proposed earlier, a neural network, would probably pose the same overfitting issues and often outputs overconfident intervals.

To summarise, forecasting in a real-world context is not only about minimizing errors, but also about understanding the uncertainty in the phenomenon, to better assess the forecasting performance consistently and over time, and not on a specific data “snapshot” taken in a specific window.