Unit 08: Sybill Trelawney and Time Series Forecasting

Applied AI with R

Slides by Ferdinand Ferber and Wolfgang Trutschnig

Paris Lodron Universität Salzburg

2026-06-06

Table of contents I

1. Time Series Decomposition
2. Stationarity and I(d)
3. AR(p) Models
4. MA(q) Models
5. ARIMA(p,d,q) Models

Sybill Trelawney and Time Series Forecasting

AI generated image for the prompt "Sybill Trelawney in her Hogwards office looking into a crystal ball to see the future."

Introduction

- Task: We are given a time series of a certain quantity (e.g., daily number of passengers for an airline, ATM data, etc.) and want to predict the future
- Simple and pragmatic idea: Decompose the time series into three parts:
  - Trend: The overall evolution of the time series; e.g. for the passenger data we expect the number of airline passengers to increase
  - Seasonality: periodic fluctuation at regular intervals, e.g. more passengers in summer than in autumn
  - Remainder: Everything else, after accounting for the trend and seasonalities (there can be more than one)
- Then we can forecast the future by continuing the trend and correcting for seasonalities. The remainder can be further modeled.

Introduction

- For handling the stochastic remainder, we will use ARIMA models
- They consists of three components:
  - **AR** stands for autoregressive. The idea is that the future value can be predicted as some linear combination of the past values.
  - **MA** stands for moving average. The idea is that random shocks influence the time series and that this influence can be modeled as a moving average of the shocks.
  - **I** stands for integrated. The theory behind the AR and MA models requires the time series to be *stationary*, i.e., its statistical properties do not change over time. Often, differencing a non-stationary time series may produce a stationary one. The reverse transformation is called integration.

Section 1

Time Series Decomposition

Time Series Decomposition

- The afore-mentioned decomposition of a time series has the following form:

$$Y_t = T_t + S_t + R_t$$

- Thereby Y_t is the original value at time t , T_t is the trend (at time t), S_t is the seasonality (at time t) and R_t the residual (at time t)
- The time index ranges in an ordered set - think of t denoting years, months, days, minutes, seconds, or any other relevant unit.

Time Series Decomposition

- The process of decomposing a time series is simple:
- First, the trend T_t is estimated
- Then, the trend is subtracted from the original time series
- On the resulting detrended time series $Y_t - T_t$ the seasonality S_t (or various seasonalities) is/are estimated
- Finally, from the original time series the trend and seasonal components are subtracted
- This yields the residual time series $R_t = Y_t - T_t - S_t$ which can be further analysed (if necessary)

The Dataset

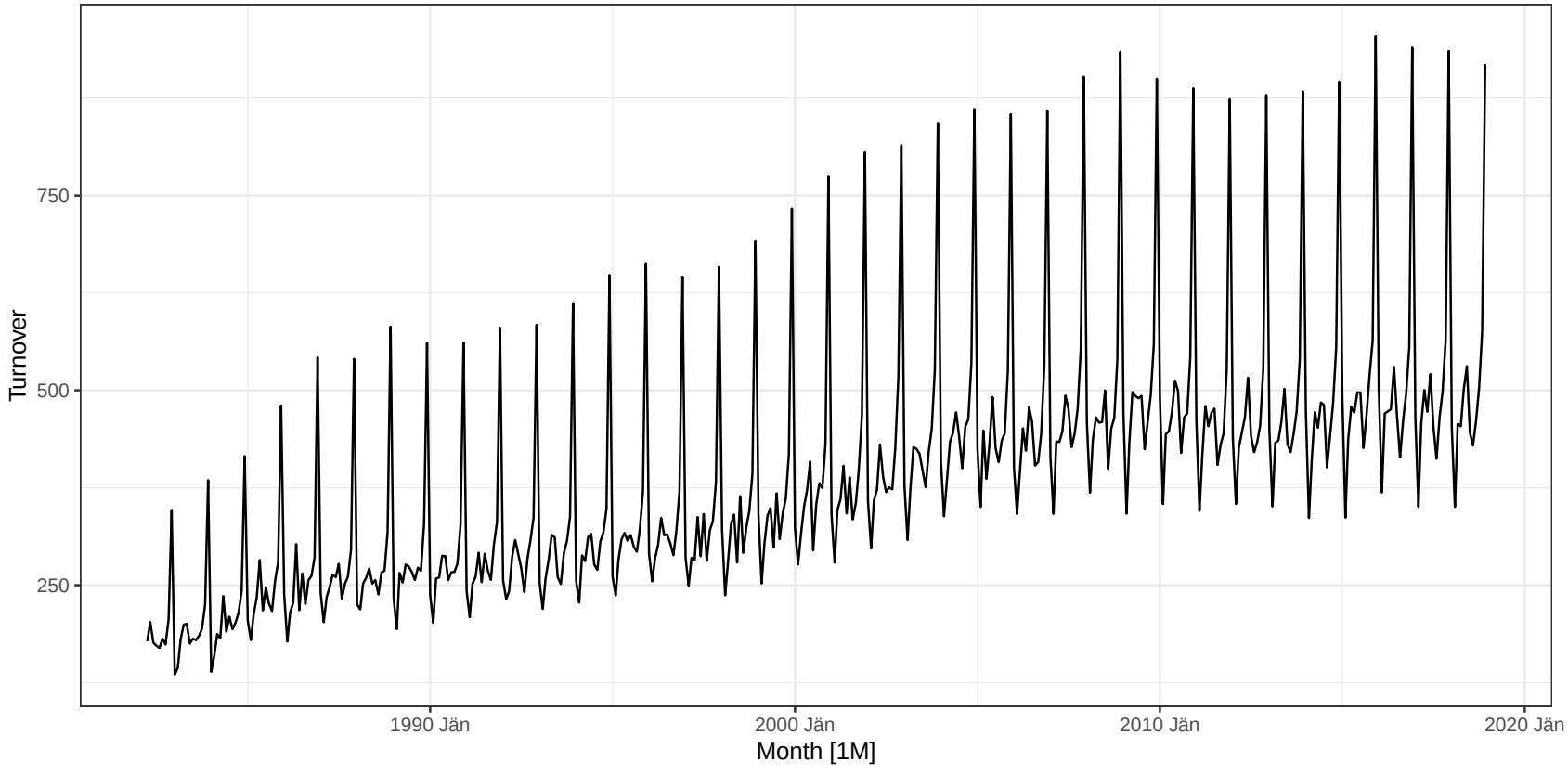
- In this unit we will use a time series on retail data of a department store in New South Wales, Australia
- This data is included in the {tsibbledata} package

```
tsdata <- tsibbledata::aus_retail |>
  filter(`Series ID` == "A3349790V") |>
  select(Month, Turnover)
```

```
# A tsibble: 5 x 2 [1M]
     Month Turnover
     <mth>    <dbl>
1 1982 Apr     178.
2 1982 Mai     203.
3 1982 Jun     176.
4 1982 Jul     173.
5 1982 Aug     170.
```

The Dataset

- This time series can be plotted using `autoplot()`:



Estimating the Trend

- There are several methods to estimate the trend of a time series.
- We will use a classical approach (well-known since Covid19):
- Work with a simple moving average (with a window size equal to the seasonality):

Estimated trend of a time series

$$T_t := \frac{1}{2k+1} \sum_{j=-k}^{k} Y_{t+j}$$

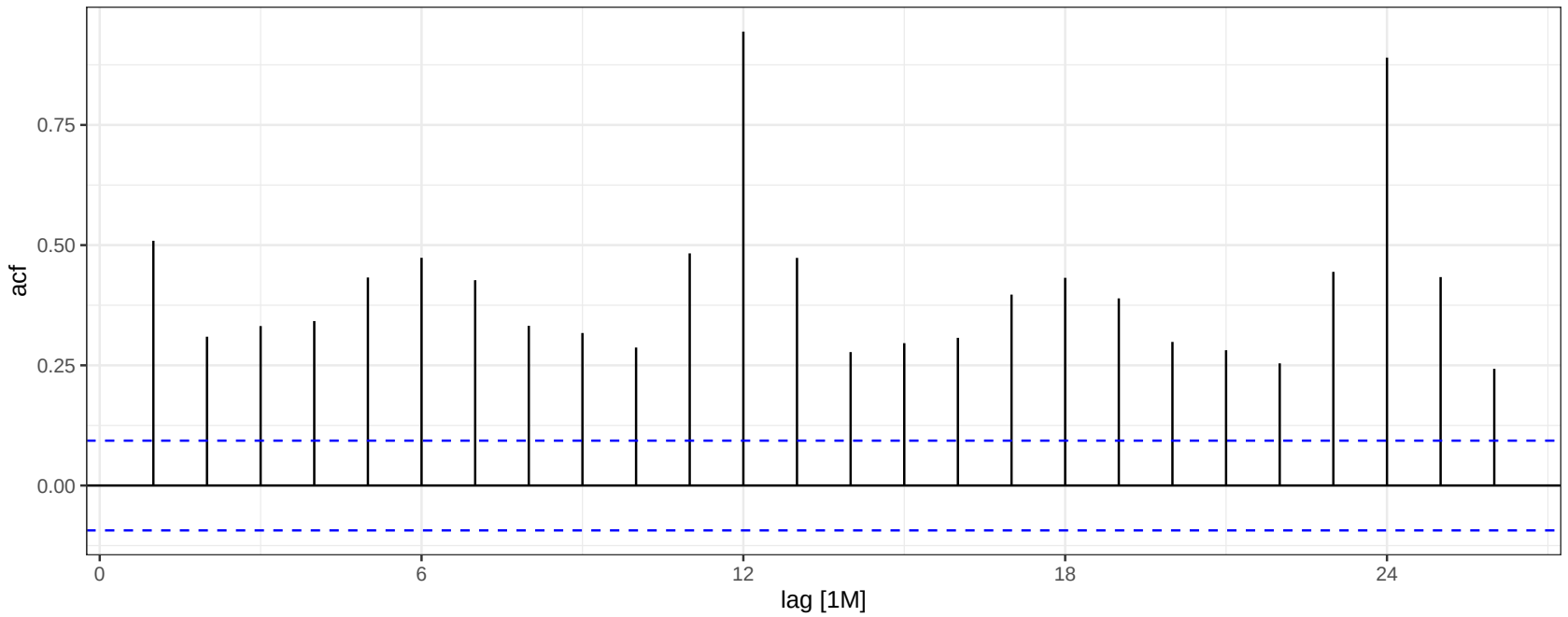
- $2k+1$ can (and frequently is), but does not need to be the estimated seasonality

Estimating the Trend

- Suppose $2k+1$ is the seasonality paramter (1 year, 1 week, etc.).
- We need an estimate for the seasonality to choose the right window size - natural way to get it:
  - Look at the plot of the time series and manually estimate it
  - Use an autocorrelation plot to estimate it
- The autocorrelation function ("ACF") computes the correlation of a time series with a delayed/lagged copy of itself as a function of the lag
- E.g., the correlation of the sample
  - $(Y_0, Y_7), (Y_1, Y_8), (Y_2, Y_9), \ldots$ yields the ACF for lag=7
  - $(Y_0, Y_{14}), (Y_1, Y_{15}), (Y_2, Y_{16}), \ldots$ yields the ACF for lag=14

Estimating the Trend

```
tsdata |> ACF(Turnover) |> autoplot() + theme_bw()
```



- We see two peaks at 12 months and 24 months, so there is good reason to assume the seasonality is 1 year

Exercise

- Discuss the following code snippet

```
tsdata |>
  mutate(1:12 |>
           map(\(n) lead(Turnover, n)) |>
           bind_cols()) |>
  rowwise() |>
  mutate(trend = mean(c_across(starts_with("..."))))
```

- Why don't we supply a variable name in the `mutate()` call?
- What are `rowwise()` and `c_across()` doing?
- Run the code snippet, plot `Month` against `trend` and `Month` against `Turnover-trend`

Identifying Seasonality

- Call $D_t := Y_t - T_t$ the detrended time series
- Suppose the seasonality is s, and we have n seasons (e.g. n years if the seasonality s is 12 months)
- It makes sense to build averages over seasons, e.g., averages over all Januaries, Februaries, etc.
- We therefore cut D_t into n slices, $D_t^{(0)}, \ldots, D_t^{(n-1)}$ with

$$D_t^{(i)} := D_{i \cdot s + t}, \quad 0 \leq t < s$$

- And can estimate the seasonal part of Y_t as

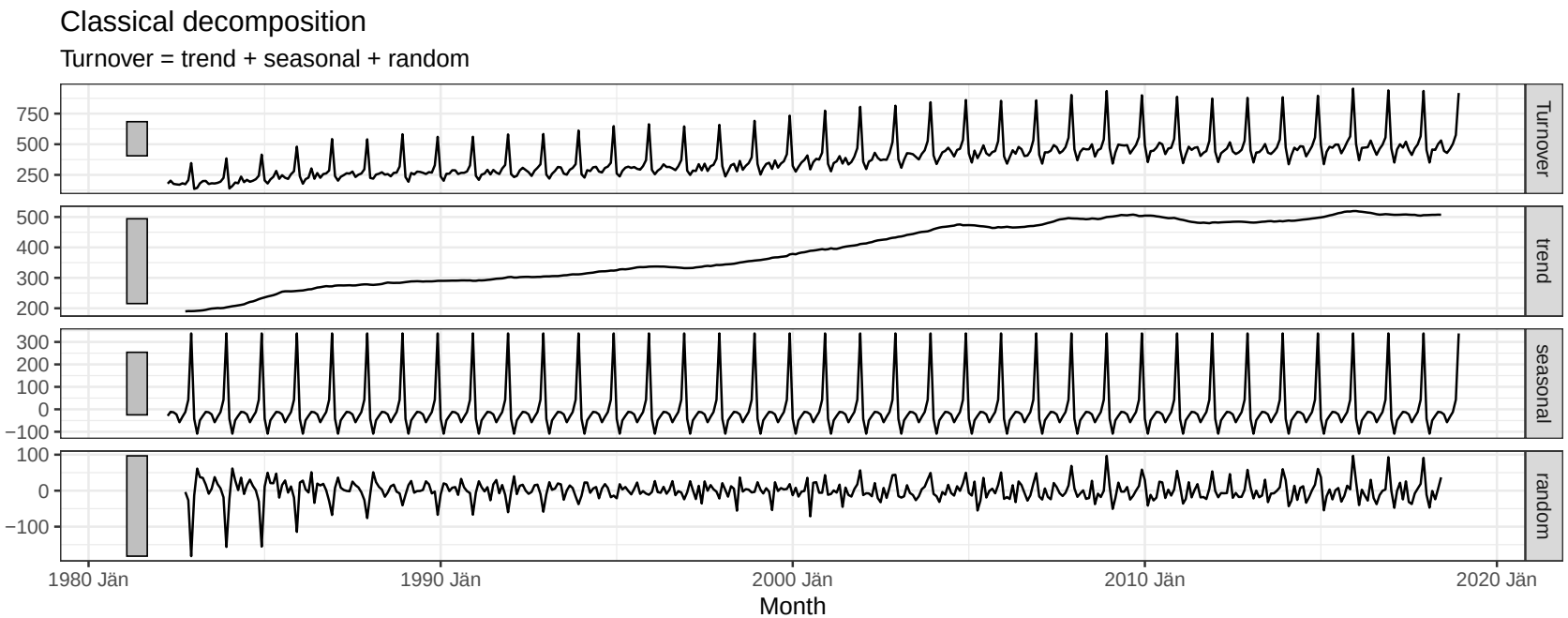
Seasonal component of a time series

$$S_t := \frac{1}{n} \sum_{i=0}^{n-1} D_t^{(i)} \mod s$$

Time Series Decomposition in R

- The {feasts} package does the decomposition for us:

```
tsdata |>
  model(classical_decomposition(Turnover)) |>
  components() |>
  autoplot() + theme_bw()
```



Forecasting

- Notice how the `{feasts}` package automatically determined the correct seasonality for us (how?)
- But you can specify it with `season(period = 12)` manually
- Notice, that the remainder still shows some seasonality
  - Argue why this is the case for this sales dataset?
  - You can use `classical_decomposition(..., type = "multiplicative")` to decompose the time series as $Y_t = T_t \cdot S_t \cdot \epsilon_t$. What would be the advantage of it?

Forecasting

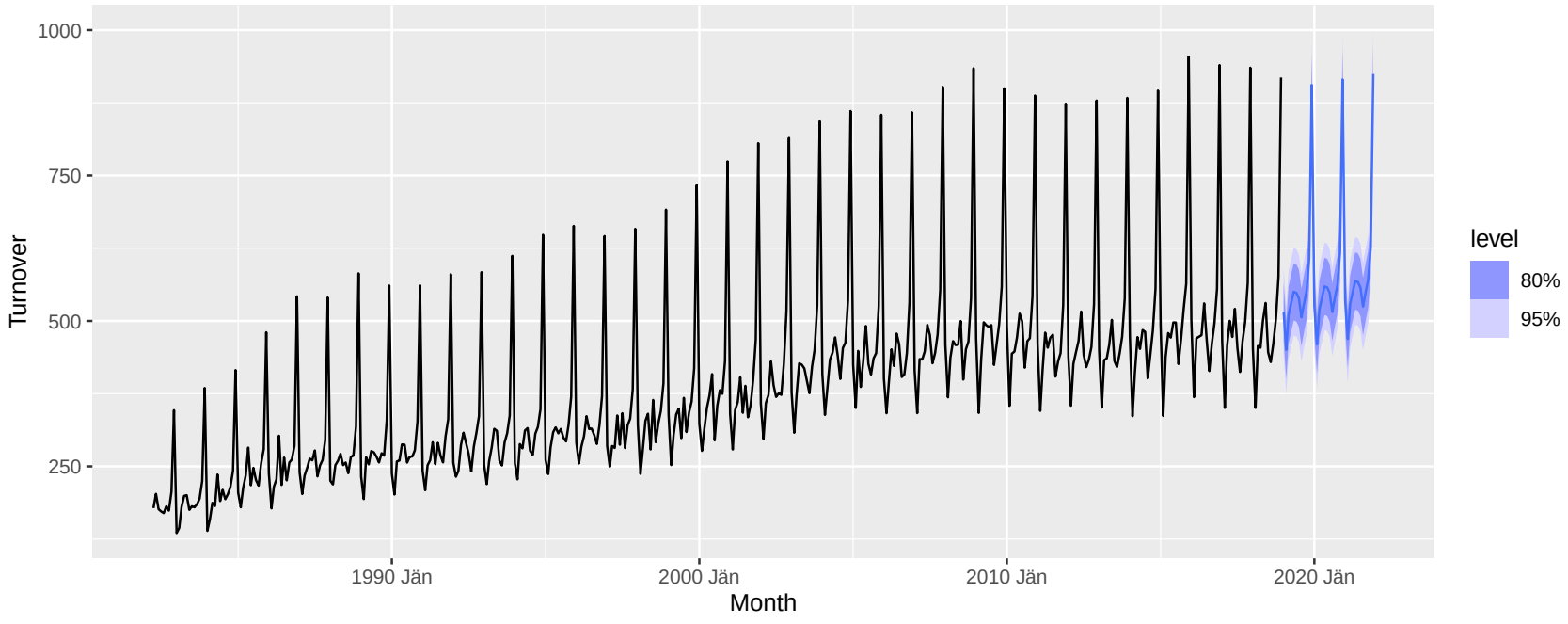
- As stated before, the idea to forecast a time series is to model the trend and then adding the seasonal effects
- For forecasting the timeseries, the trend has to be forecasted as well...moving averages won't be of much help...
- One of the most basic parametric (!) ways of modeling the trend is to fit a simple linear regression to the trend component of the time series, i.e.

$$T_t = \beta_0 + \beta_1 t$$

- In the `{feasts}` package, we can use the `TSLM()` function to model a linear trend and additive seasonality

Forecasting

```
tsdata |>
  model(TSLM(Turnover ~ trend() + season())) |>
  forecast(h = 36) |>
  autoplot(tsdata)
```



Exercise

- You are already familiar with the original ATM dataset
- Load it, decompose the time series into a trend, a seasonal component, and residuals and plot it
- Forecast the time series using a linear model for the trend

Outlook

- The classical decomposition has some disadvantages
  - Only one season is supported
  - The moving average (how we defined it) has always an odd window size, some care is needed when the seasonality period is even
- A better method would be *STL decomposition* that is more robust
  - Trend and seasonality are estimated using LOESS (some sort of locally weighted regression)
  - Iterative refinement is used to improve the decomposition. Notice that we need an estimate for the seasonality to get the trend and we need an estimate for the trend to get the seasonality.
- In the `{feasts}` package the function `STL()` can be used instead of `classical_decomposition()`

Section 2

Stationarity and I(d)

Motivation

- In the last section we have decomposed a time series into a trend and seasonal effects
- Then we modeled the trend as a linear function and did forecasting by just continuing the trend and adding the seasonal component
- In this section we focus on ARIMA models which can be used to further model the residuals
- First, we look at the three components of the ARIMA model: `AR(p)` models, `I(d)` models and `MA(q)` models
- Then we will introduce the general `ARIMA(p,d,q)` approach

Time Series Definition

- We first quickly sketch the mathematical approach to *time series*, i.e., the general framework into which they can be embedded
- Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space
- Then a *stochastic process* is a family of random variables $(X_t)_{t\in\mathbb{N}}$ with $X_t : \Omega \to \mathbb{R}$
- We don't require the random variables to be independent - it's the whole point that there is a dependency structure in a time series
- A *time series* can be seen as a concrete realization of a stochastic process

Cumulative Distribution Function

- Recall the following notions from elementary probability:
- The *cumulative distribution function (cdf)* for a real-valued random variable X is defined as $F_X(x) := \mathbb{P}[X \leq x]$
- If you know the cdf F_X, you know the distribution of X
- For d real-valued random variables the *joint cdf* $F_{X_1,\ldots,X_d}$ is defined as

$$F_{X_1,\ldots,X_d}(x_1, \ldots, x_d) := \mathbb{P}[X_1 \leq x_1, \ldots, X_d \leq x_d]$$

Strict-Sense Stationary Time Series

Strict-sense stationary time series

A time series X_t is called *strict-sense stationary* if for all $n \in \mathbb{N}^+$ and all $\tau, t_1, \ldots, t_n \in \mathbb{N}$ it holds that

$$F_{X_{\tau+t_1},\ldots,X_{\tau+t_n}} = F_{X_{t_1},\ldots,X_{t_n}}$$

- In particular:
  - All statistical properties (mean, variance, ...) of the random variables X_t are independent of the time t.
  - All univariate marginal distributions are the same, i.e., $F_{X_t} = F_{X_s}$ for all t, s

Relevance for Modeling

- When modeling the time series Y_t, we are essentially modeling $\mathbb{E}\left[Y_{t+1} \mid Y_t, Y_{t-1}, \dots, Y_0\right]$ based on the available history $Y_0, \dots, Y_t$ of the time series
- Notice that this conditional expectation can be expressed in terms of the joint cdfs:
- The conditional cdf of Y_{t+1} can be expressed as

$$F_{Y_{t+1}|Y_t,\dots,Y_0}(y_{t+1}|y_t,\dots,y_0) = \frac{F_{Y_{t+1},\dots,Y_0}(y_{t+1},\dots,y_0)}{\mathbb{P}(Y_t = y_t,\dots,Y_0 = y_0)}$$

- ...whereby the left hand side is short for $\mathbb{P}[Y_{t+1} \leq y_{t+1}|Y_t = y_t,\dots,Y_0 = y_0]$
- This is tricky since the event $\{Y_t = y_t,\dots,Y_0 = y_0\}$ might have probability zero - we'll skip technical details

Relevance for Modeling

- Then the conditional density of Y_{t+1} (if it exists) can be calculated as

$$f_{Y_{t+1}|Y_t,\dots,Y_0}(y_{t+1}|y_t,\dots,y_0) = \frac{\partial}{\partial y_{t+1}}\left(\frac{F_{Y_{t+1},\dots,Y_0}(y_{t+1},\dots,y_0)}{\mathbb{P}(Y_t = y_t,\dots,Y_0 = y_0)}\right)$$

- Finally, the conditional expectation is

$$\begin{aligned}\mathbb{E}\left[Y_{t+1}|Y_t = y_t,\dots,Y_0 = y_0\right] &= \\ &= \int_{-\infty}^{\infty} y_{t+1}\frac{\partial}{\partial y_{t+1}}\left(\frac{F_{Y_{t+1},\dots,Y_0}(y_{t+1},\dots,y_0)}{\mathbb{P}(Y_t = y_t,\dots,Y_0 = y_0)}\right)\mathrm{d}y_{t+1}\end{aligned}$$

Relevance for Modeling

- If we want to use a model iteratively in an autoregressive manner, we need the time series to be stationary
- Otherwise a good model trained from historic data is not guaranteed to perform well on future data
- How to make timeseries (more) stationary?

What to do with Non-Stationary Time Series?

- There are two important types of non-stationary time series:
- Trend stationary: The time series is the sum of a deterministic trend plus a stationary time series
  - After modeling and substracting the trend, one is left with a stationary time series
  - This is the reason why we removed the trend in the last section and use ARIMA on the remainder component
- Difference stationary: By repeatedly *differencing* the time series it can be made stationary
  - Differencing a time series Y_t means computing $Y_t - Y_{t-1}$ (first difference/derivative)
  - Higher-order differences are computed by differencing an already differenced time series iteratively

Backshift Notation

- To simlify notation we will work with the *backshift operator*
- The *backshift operator* B maps a time series to another time series and is defined as

$$BY_t = Y_{t-1}$$

- We omit the parentheses in $B(Y_t)$ in this notation
- Repeated applying the operator is written as B^2Y_t, meaning $B^2Y_t = BY_{t-1} = Y_{t-2}$

Exercise

- What is $(1 - B)Y_t$?
- Which operator O achieves $OY_t = Y_t - 2Y_{t-1} + Y_{t-2}$?

Differencing

- As we saw, $(1 - B)Y_t = Y_t - Y_{t-1}$ is the first difference of Y_t
- Higher-order differences are computed by $(1 - B)^d Y_t$, where d is the order
- If $(1 - B)^d Y_t$ is stationary, we say that Y_t is an I(d) process

Estimating the Integration Order in R

- The {feasts} package provides a function to estimate the integration order of a difference stationary time series:

```
tsdata |>
  features(Turnover, unitroot_ndiffs)
```

```
# A tibble: 1 x 1
  ndiffs
   <int>
1      1
```

- So differencing the tsdata time series once will make it stationary (or at least slightly more stationary)

Section 3

AR(p) Models

AR(p) Models

- The core idea behind `AR(p)` models is an affine (or even linear) functin of the past p values
- In other words: If we know the past p values, then we can predict the next value.
- Therefore the wording *autoregressive*

AR(p) model

If Y_t follows an `AR(p)` model, written $Y_t \sim \text{AR}(p)$, then the following relation holds for all $t \geq p$:

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \cdots + \alpha_p Y_{t-p} + \epsilon_t$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are independently normally distributed and $\alpha_1, \ldots, \alpha_p$ are the constant (!) model coefficients.

Using the Backshift Operator

- Using the backshift operator, we can write

$$Y_t = \left(\sum_{i=1}^{p} \alpha_i B^i \right) Y_t + \epsilon_t$$

- Or equivalently

$$\left(1 - \sum_{i=1}^{p} \alpha_i B^i \right) Y_t = \epsilon_t$$

- We call the polynomial $\phi(z) := 1 - \sum_{i=1}^{p} \alpha_i z^i$ the *characteristic polynomial* of the `AR(p)` process Y_t

Stationarity again

- Based on the previous discussion it is clear that we want to model stationary time series with the `AR(p)` process
- But `AR(p)` models are not guaranteed to be stationary
- It can be shown that an `AR(p)` model is stationary and causal if, and only if the roots of its characteristic polynomial $\phi(z)$ lie outside the unit circle, i.e. each (complex) root z_i must satisfy $|z_i| > 1$
- We won't go into further details
- When estimating the model coefficients, the `{fable}` package will take care of those constraints

Section 4

MA(q) Models

MA(q) Models

- `MA(q)` are frequently considered in economics, they capture the idea that Y_t is a linear combination of random *shocks* from the recent past

MA(q) model

If Y_t follows an `MA(q)` model, written $Y_t \sim \text{MA}(q)$, then the following relation holds for all $t \geq q$:

$$Y_t = \epsilon_t + \beta_1 \epsilon_{t-1} + \beta_2 \epsilon_{t-2} + \cdots + \beta_q \epsilon_{t-q}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are independently normally distributed and $\beta_1, \ldots, \beta_q$ are the constant (!) model coefficients.

Additional Intuition

- We will later use the `MA(q)`-model as part of an `ARIMA(p,d,q)`-model
- In that case there is an additional intuition on what the `MA(q)`-part actually models
- For many time series, residuals from past periods are correlated with the residual in the current period
- I.e. an prediction error we make now will also have an influence on the errors in the next q time steps
- This serial correlation of the residuals can be captured by the `MA(q)`-part of the model

Using the Backshift Operator

- Using the backshift operator, we can write

$$Y_t = (1 + \beta_1 B + \cdots + \beta_q B^q)\epsilon_t$$

- Consequently, we define the characteristic polynomial of the `MA(q)` model as

$$\theta(z) := 1 + \sum_{i=1}^{q} \beta_i z^i$$

- For technical reasons we need to require again that all roots z_i of θ satisfy $|z_i| > 1$. Those `MA(q)`-models are called *invertible* for reasons we don't touch here.

Section 5

ARIMA(p,d,q) Models

ARIMA(p,d,q) Models

- We can plug everything together and now define the `ARIMA(p,d,q)` model as:

ARIMA(p,d,q) model

If Y_t follows an `ARIMA(p,d,q)` model, written $Y_t \sim \text{ARIMA}(p, d, q)$, then the following relation holds for all $t \geq \max(p, q)$:

$$\phi(B)(1 - B)^d Y_t = \theta(B)\epsilon_t$$

where $\phi(z)$ is the characteristic polynomial of an `AR(p)` process, d is the integration order, $\theta(z)$ is the characteristic polynomial of an `MA(q)` process and the ϵ_t are i.i.d. normally distributed.

Fitting the Model

- For estimating the parameters p and q there are two common methods:
- Manual estimation
  - For p one can look at the *partial autocorrelation function (PACF)* and look how many lags are significantly non-zero
  - For q one can look at the *autocorrelation function (ACF)* and look how many lags are significantly non-zero
- Automatic estimation: One uses grid search for different values of p and q and chooses the best model based on the *Akaike Information Criteria (AIC)*, which tries to balance goodness-of-fit and complexity of the model

Fitting the Model

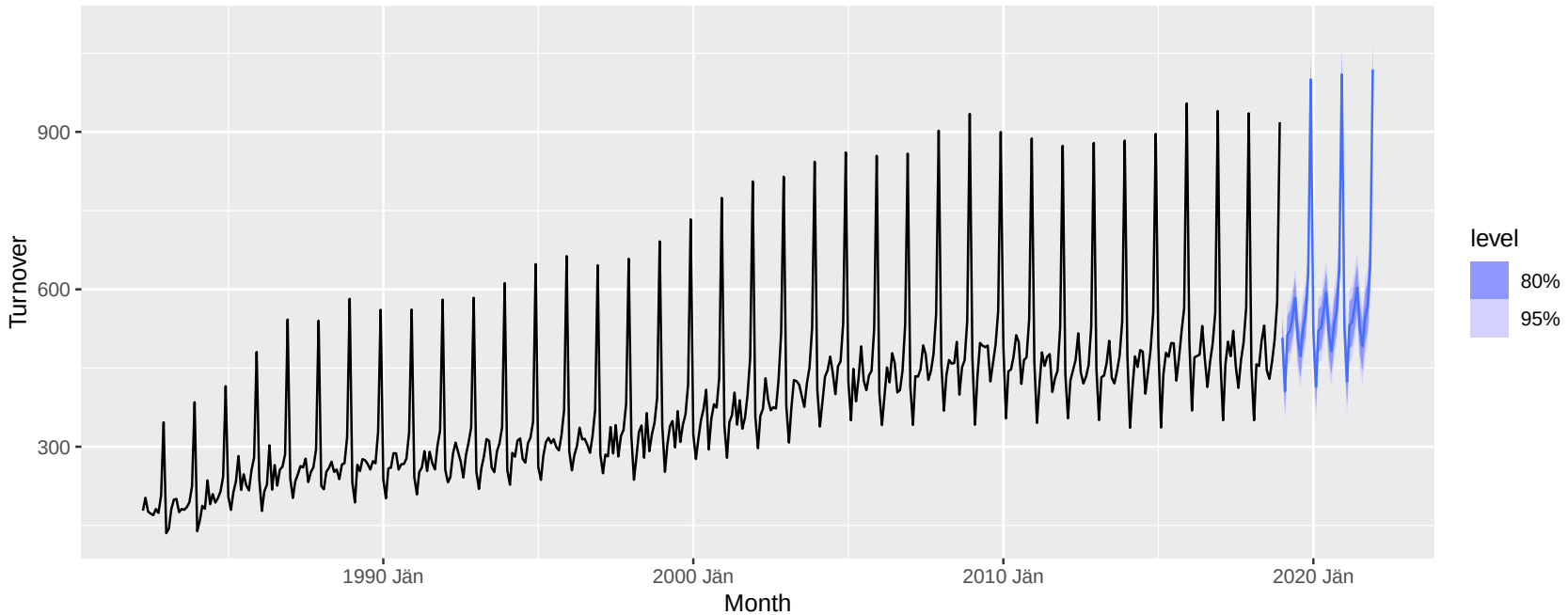
- With the {fable} package an ARIMA model can be fully automatically trained using

```
fitted <- some_time_series |>
  model(ARIMA(the_outcome ~ PDQ(0,0,0)))
```

- There exists a variant of ARIMA, called SARIMA, that supports seasonality and the `ARIMA()` function uses this variant as default
- The `PDQ(0,0,0)` argument tells the (S)ARIMA model to ignore seasonality

Putting everything together

- Our strategy is to decompose the time series into a deterministic trend, a seasonal component and then to further model the random/stochastic remainder as an ARIMA process
- We can do this easily with the {fable} package:

```
tsdata |>
  model(decomposition_model(
    classical_decomposition(Turnover),
    TSLM(trend ~ trend()),
    SNAIVE(seasonal),
    ARIMA(random)
  )) |>
  forecast(h=36) |>
  autoplot(tsdata)
```

Putting everything together