AARHUS UNIVERSITY

Data Science 8

Chris Mathys

Master's Degree Programme in Cognitive Science

Spring 2022

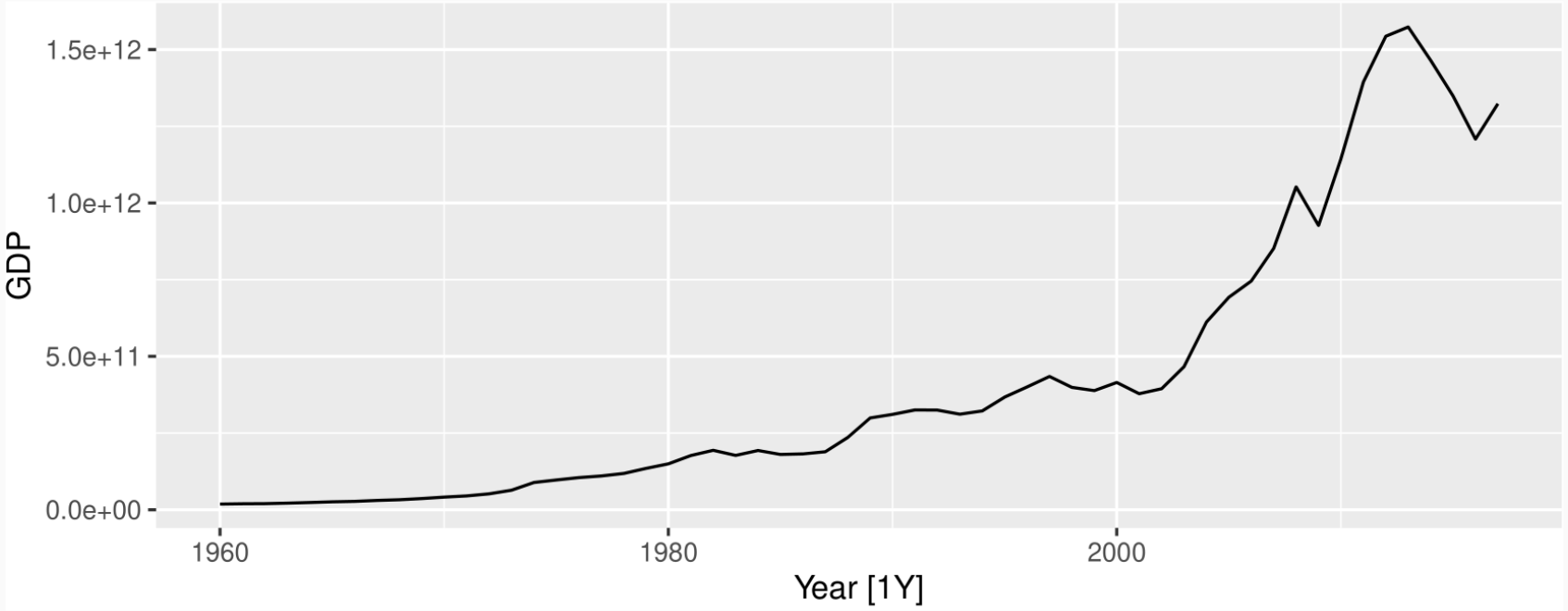
Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Per capita adjustments

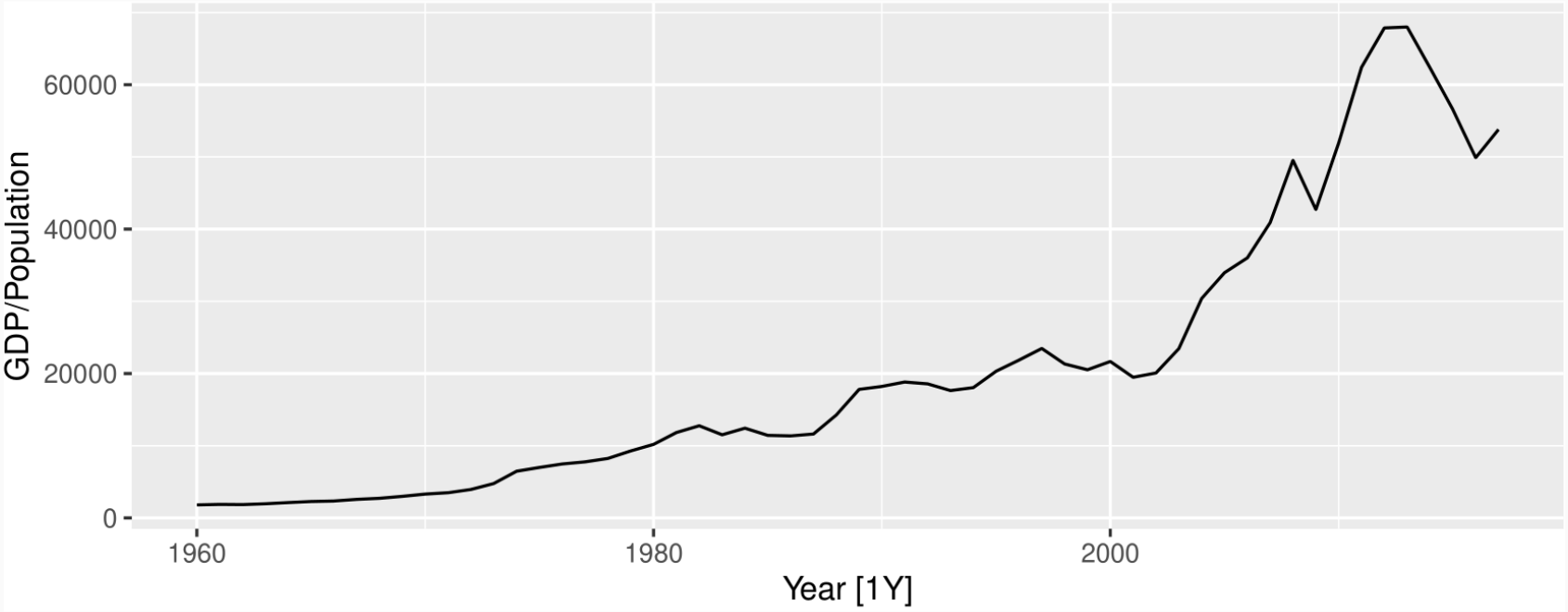
```
global_economy %>%
  filter(Country == "Australia") %>%
  autoplot(GDP)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Per capita adjustments

```
global_economy %>%
  filter(Country == "Australia") %>%
  autoplot(GDP / Population)
```



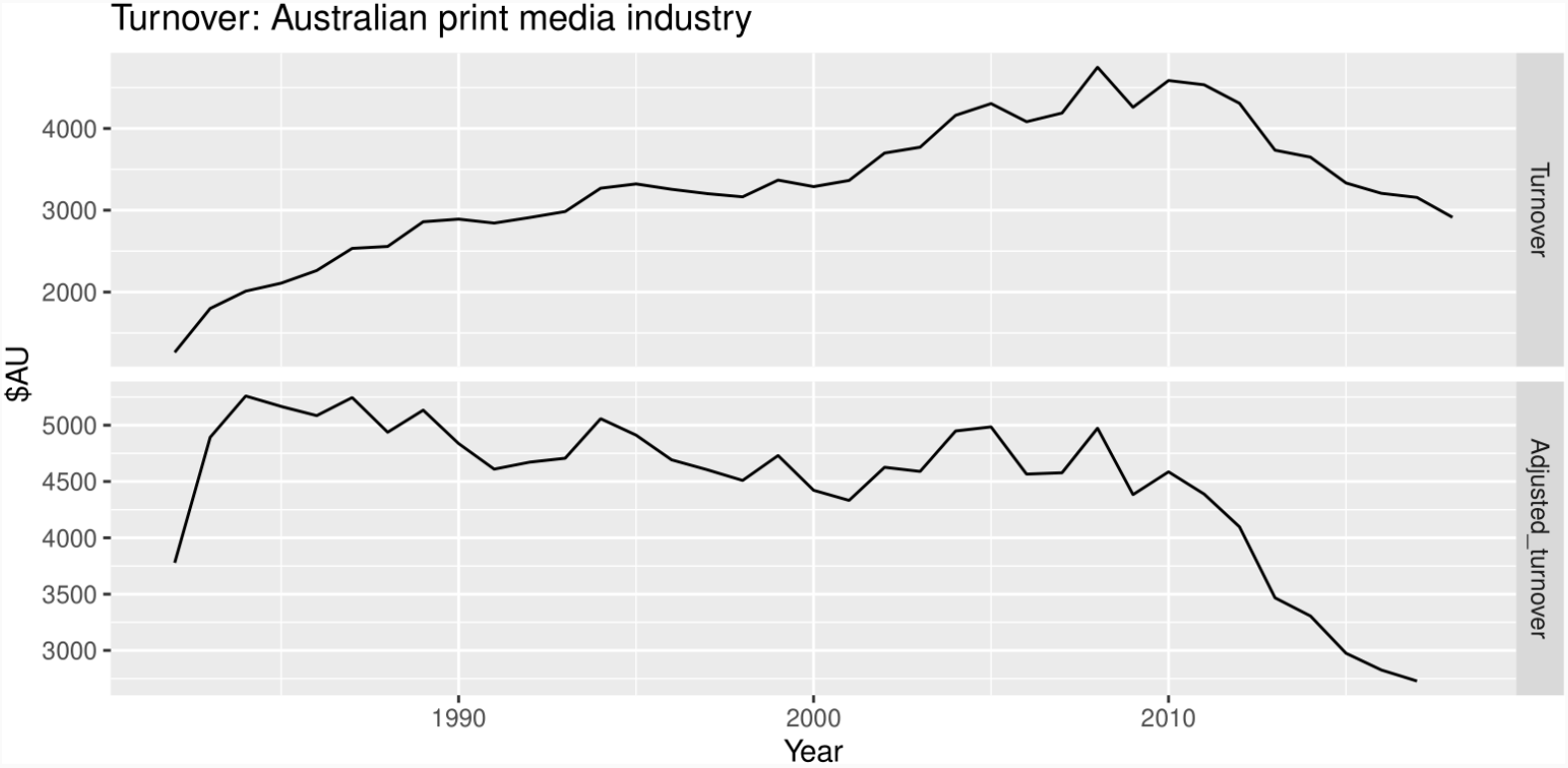
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Inflation adjustments

```
print_retail <- aus_retail %>%
  filter(Industry == "Newspaper and book retailing") %>%
  group_by(Industry) %>%
  index_by(Year = year(Month)) %>%
  summarise(Turnover = sum(Turnover))
aus_economy <- global_economy %>%
  filter(Code == "AUS")
print_retail %>%
  left_join(aus_economy, by = "Year") %>%
  mutate(Adjusted_turnover = Turnover / CPI * 100) %>%
  pivot_longer(c(Turnover, Adjusted_turnover), values_to = "Turnover") %>%
  mutate(name = factor(name, levels=c("Turnover","Adjusted_turnover"))) %>%
  ggplot(aes(x = Year, y = Turnover)) +
  geom_line() +
  facet_grid(name ~ ., scales = "free_y") +
  labs(title = "Turnover: Australian print media industry", y = "$AU")
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Inflation adjustments



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \ldots, y_n$ and transformed observations as $w_1, \ldots, w_n$.

Mathematical transformations for stabilizing variation

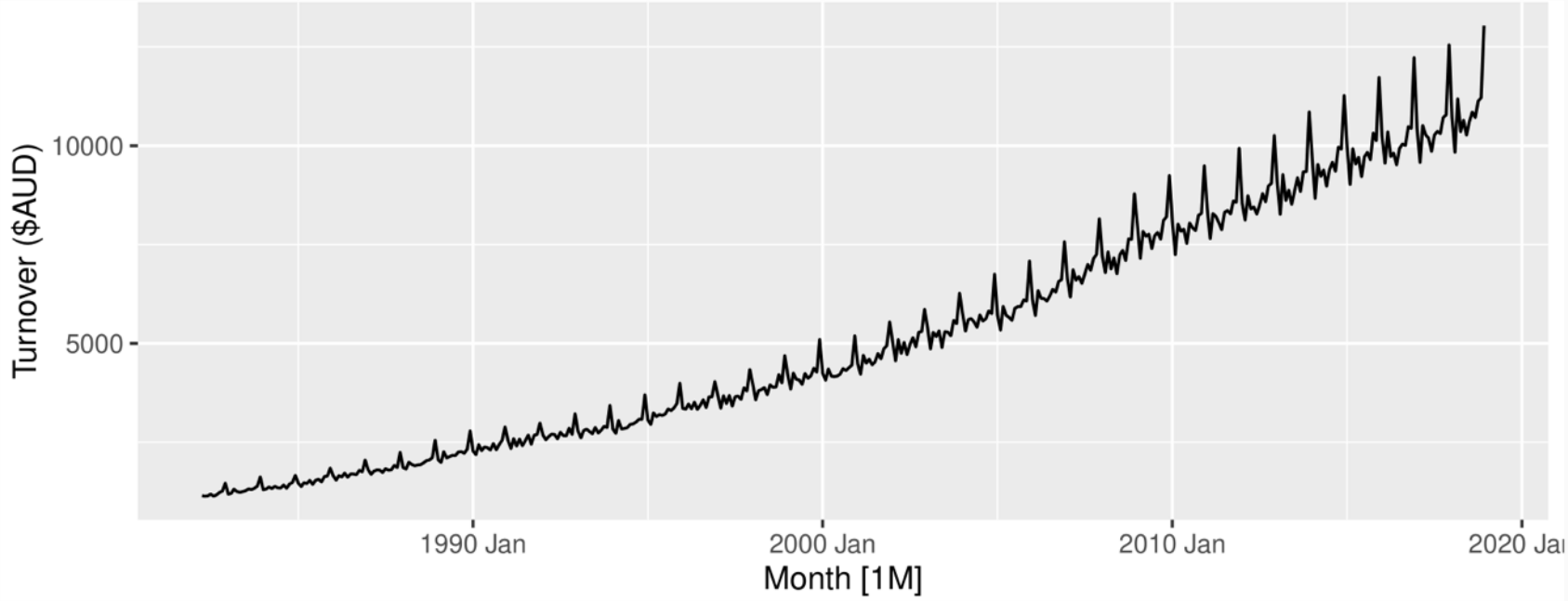
Square root	$w_t = \sqrt{y_t}$	↓
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale**.

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

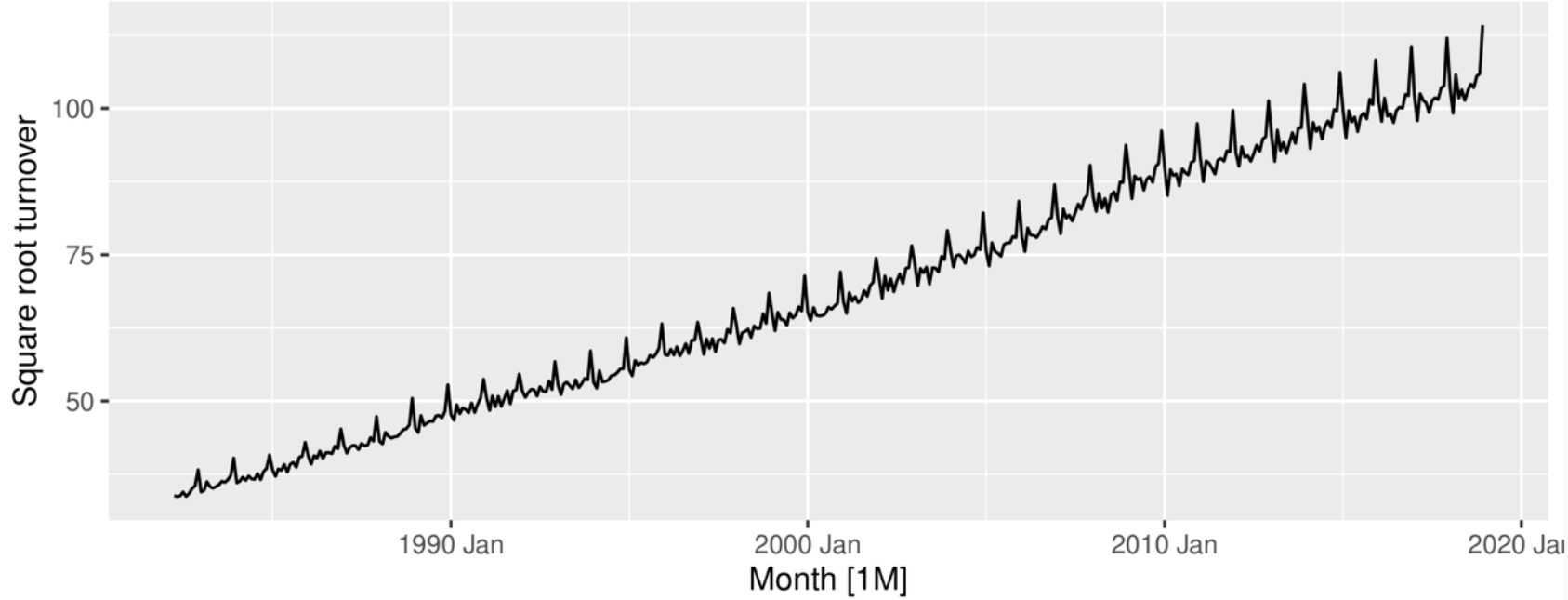
```
food <- aus_retail %>%
  filter(Industry == "Food retailing") %>%
  summarise(Turnover = sum(Turnover))
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

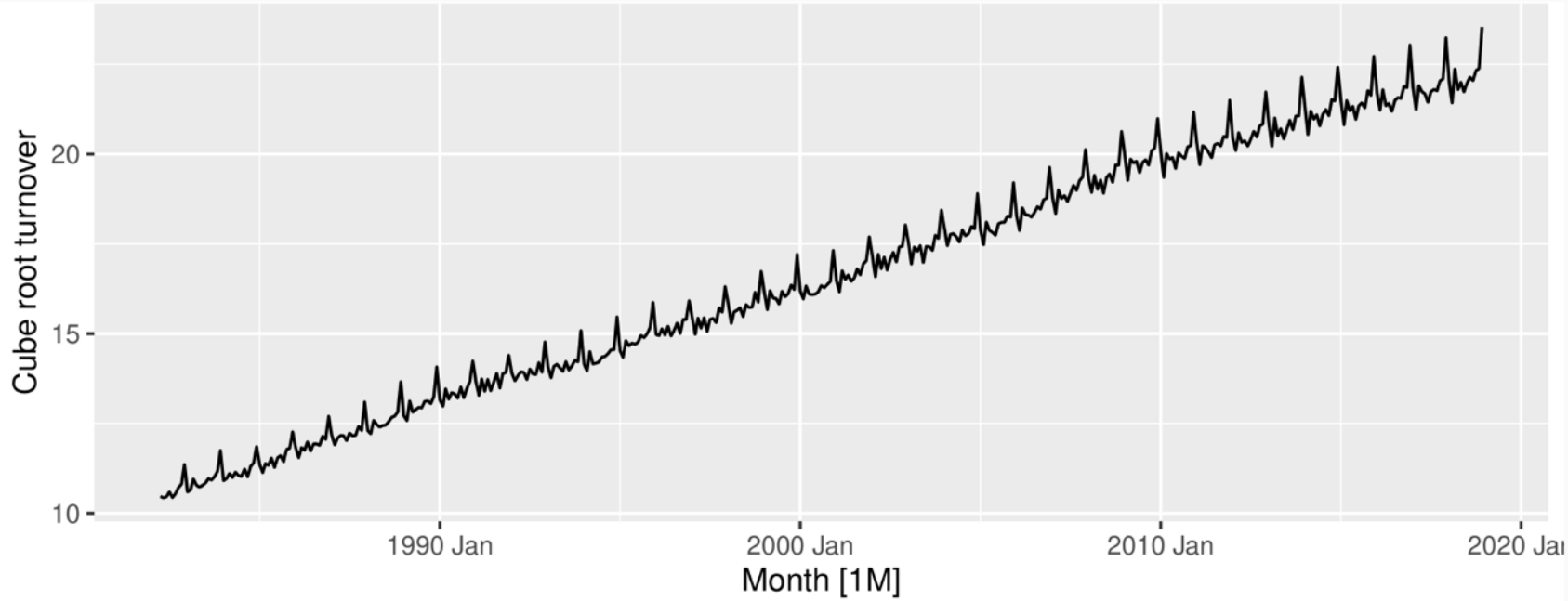
```
food %>% autoplot(sqrt(Turnover)) +
  labs(y = "Square root turnover")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

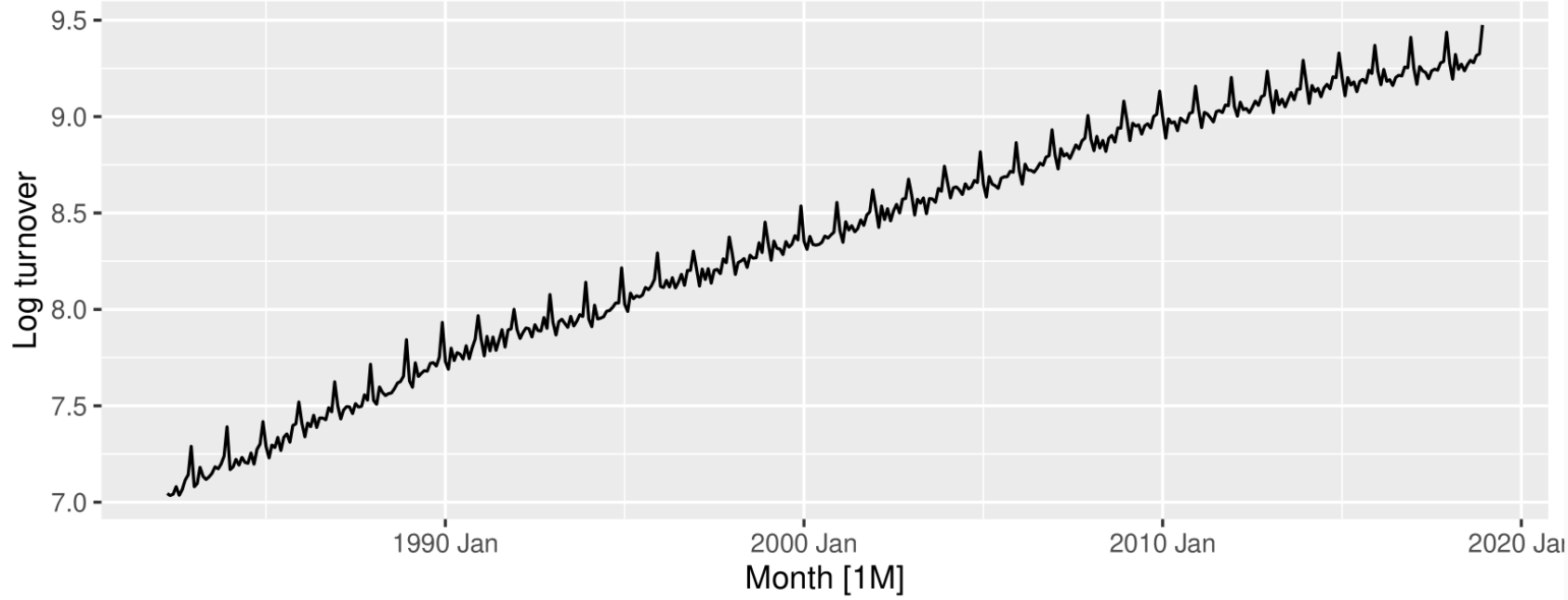
```
food %>% autoplot(Turnover^(1/3)) +
  labs(y = "Cube root turnover")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

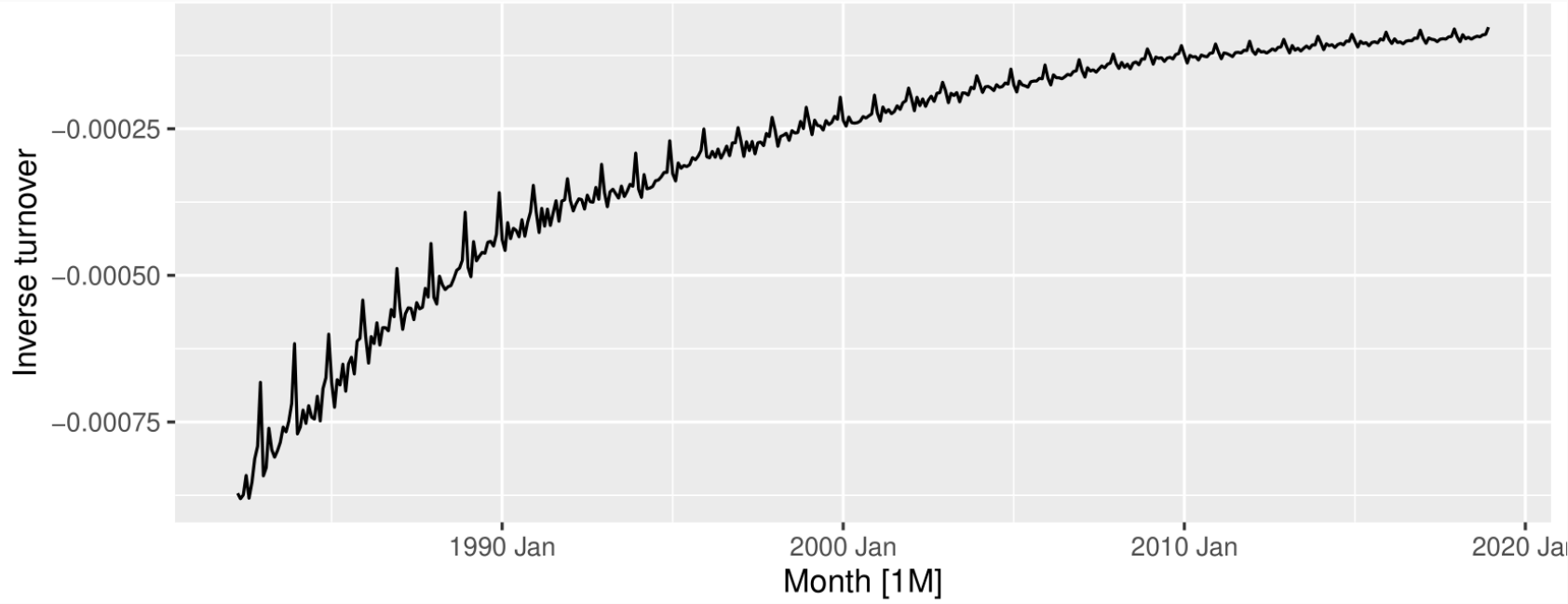
```
food %>% autoplot(log(Turnover)) +
  labs(y = "Log turnover")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Mathematical transformations

```
food %>% autoplot(-1/Turnover) +
  labs(y = "Inverse turnover")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Box-Cox transformations

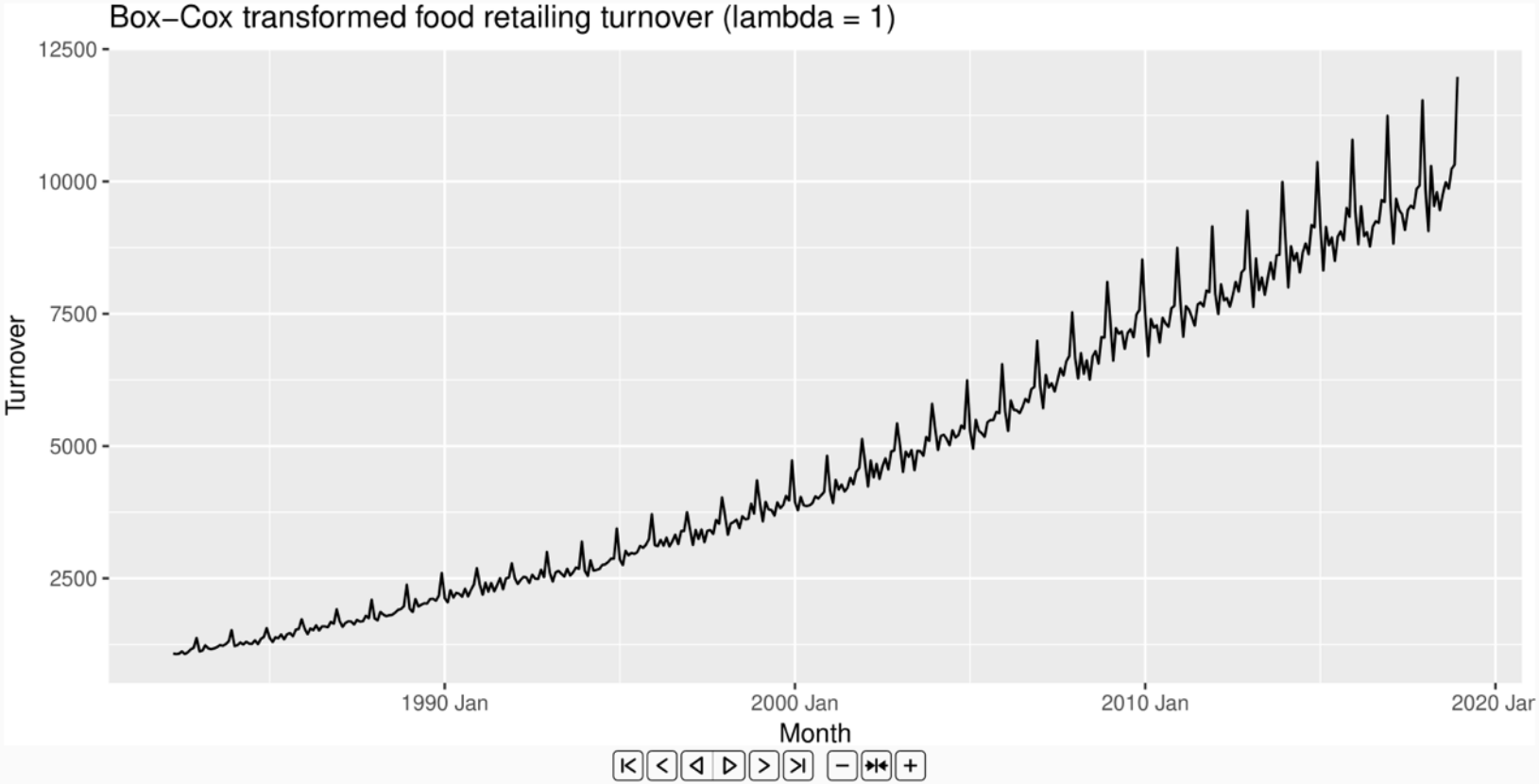
Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (sign(y_t)|y_t|^{\lambda} - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- Actually the Bickel-Doksum transformation (allowing for $y_t < 0$)
- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Box-Cox transformations



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Box-Cox transformations

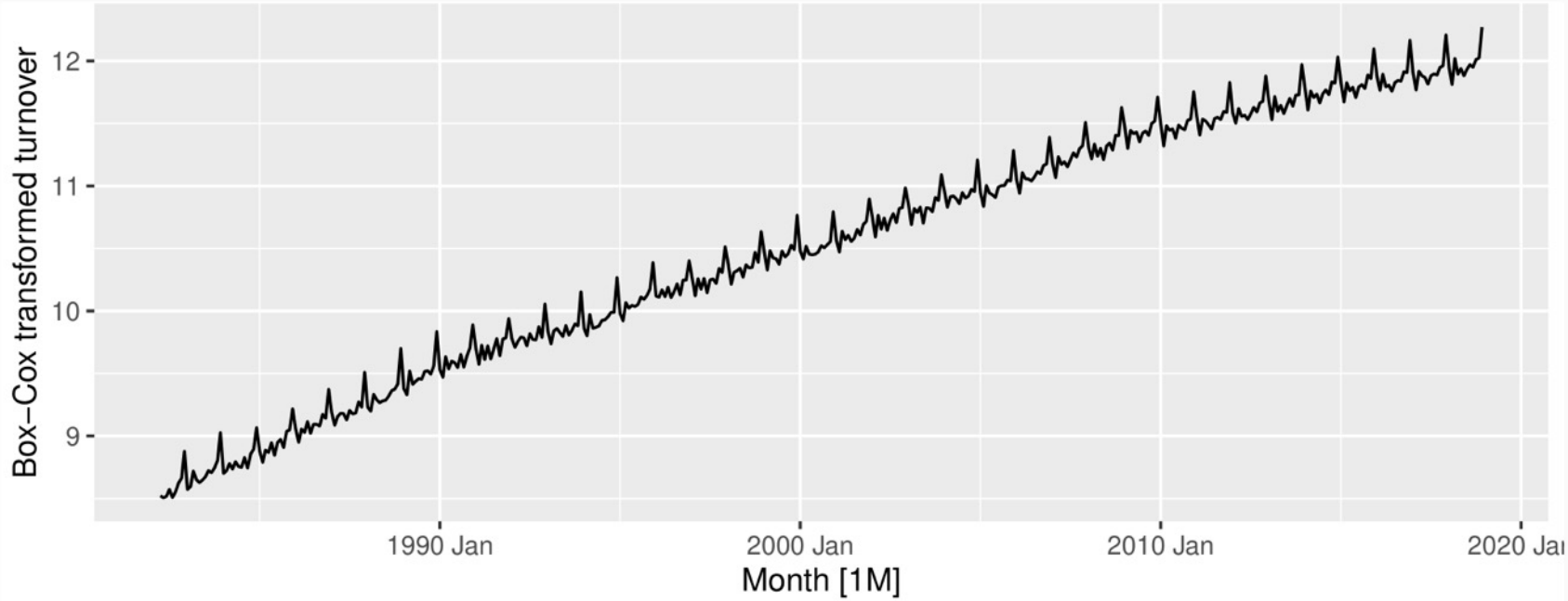
```
food %>%
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1          0.0524
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

Box-Cox transformations

```
food %>% autoplot(box_cox(Turnover, 0.0524)) +
  labs(y = "Box-Cox transformed turnover")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Transformations

- Often no transformation needed.
- Simple transformations are easier to explain and work well enough.
- Transformations can have very large effect on PI.
- If some data are zero or negative, then use $\lambda > 0$.
- `log1p()` can also be useful for data with zeros.
- Choosing logs is a simple way to force forecasts to be positive
- Transformations must be reversed to obtain forecasts on the original scale. (Handled automatically by `fable`.)

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series patterns

Recall

Trend pattern exists when there is a long-term increase or decrease in the data.

Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).

Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series decomposition

$$y_t = f(S_t, T_t, R_t)$$

where y_t = data at period t

T_t = trend-cycle component at period t

S_t = seasonal component at period t

R_t = remainder component at period t

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series decomposition

$$y_t = f(S_t, T_t, R_t)$$

where y_t = data at period t

T_t = trend-cycle component at period t

S_t = seasonal component at period t

R_t = remainder component at period t

Additive decomposition: $y_t = S_t + T_t + R_t$.

Multiplicative decomposition: $y_t = S_t \times T_t \times R_t$.

Time series decomposition

- Additive model appropriate if magnitude of seasonal fluctuations does not vary with level.
- If seasonal are proportional to level of series, then multiplicative model appropriate.
- Multiplicative decomposition more prevalent with economic series
- Alternative: use a Box-Cox transformation, and then use additive decomposition.
- Logs turn multiplicative relationship into an additive relationship:

$$y_t = S_t \times T_t \times R_t \quad \Rightarrow \quad \log y_t = \log S_t + \log T_t + \log R_t.$$

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

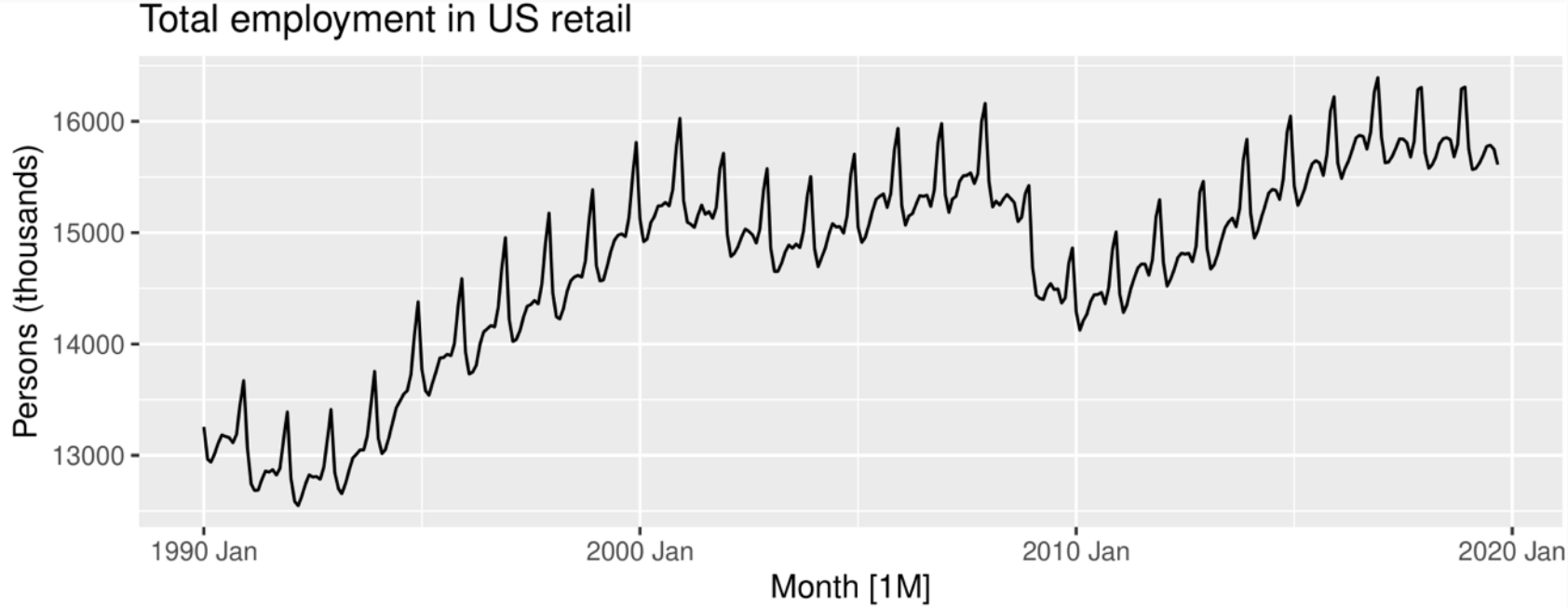
```
us_retail_employment <- us_employment %>%
  filter(year(Month) >= 1990, Title == "Retail Trade") %>%
  select(-Series_ID)
us_retail_employment
```

```
## # A tsibble: 357 x 3 [1M]
##       Month Title        Employed
##       <mth> <chr>           <dbl>
##  1 1990 Jan Retail Trade   13256.
##  2 1990 Feb Retail Trade   12966.
##  3 1990 Mar Retail Trade   12938.
##  4 1990 Apr Retail Trade   13012.
##  5 1990 May Retail Trade   13108.
##  6 1990 Jun Retail Trade   13183.
##  7 1990 Jul Retail Trade   13170.
##  8 1990 Aug Retail Trade   13160.
##  9 1990 Sep Retail Trade   13113
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

```
us_retail_employment %>%
  autoplot(Employed) +
  labs(y="Persons (thousands)", title="Total employment in US retail")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

```
us_retail_employment %>%
  model(stl = STL(Employed))
```

```
## # A mable: 1 x 1
##       stl
##   <model>
## 1   <STL>
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

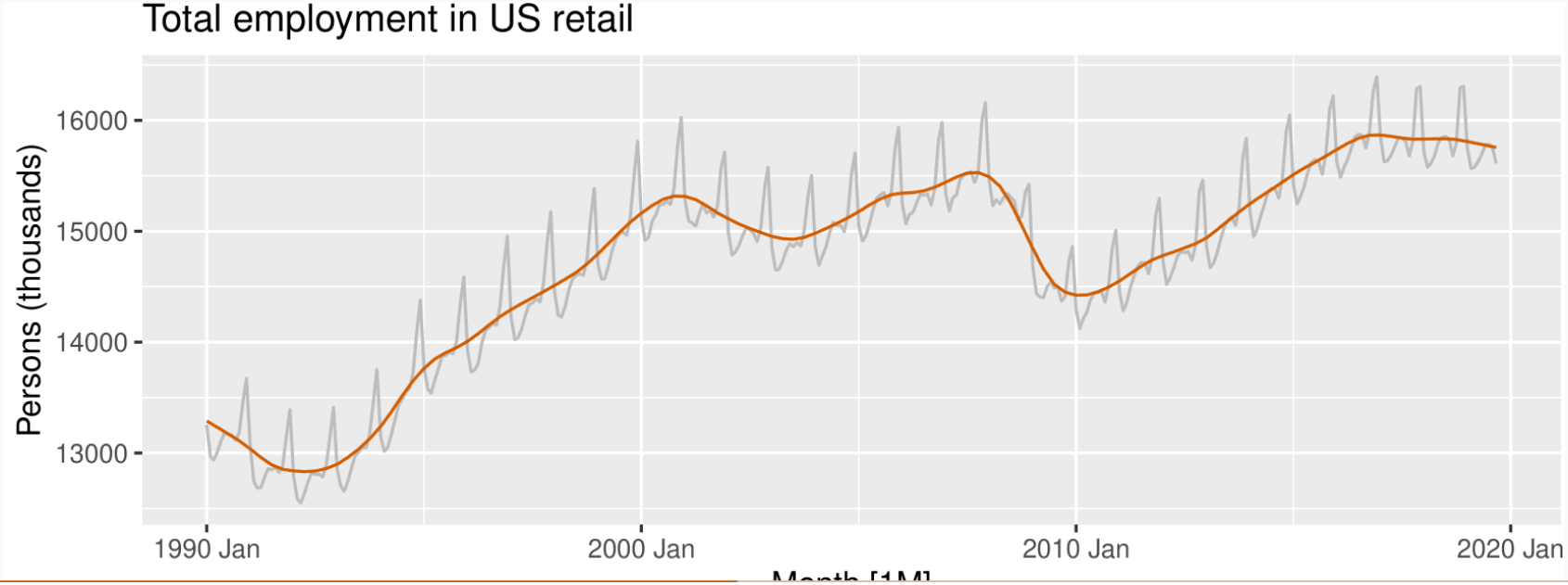
```
dcmp <- us_retail_employment %>%
  model(stl = STL(Employed))
components(dcmp)
```

```
## # A dable: 357 x 7 [1M]
## # Key:     .model [1]
## # :        Employed = trend + season_year + remainder
##    .model    Month Employed  trend season_year remainder season_adjust
##    <chr>     <mth>    <dbl>  <dbl>       <dbl>     <dbl>          <dbl>
##  1 stl    1990 Jan   13256. 13288.      -33.0      0.836         13289.
##  2 stl    1990 Feb   12966. 13269.     -258.     -44.6           13224.
##  3 stl    1990 Mar   12938. 13250.     -290.     -22.1           13228.
##  4 stl    1990 Apr   13012. 13231.     -220.       1.05          13232.
##  5 stl    1990 May   13108. 13211.     -114.      11.3           13223.
##  6 stl    1990 Jun   13183. 13192.      -24.3     15.5           13207.
##  7 stl    1990 Jul   13170. 13172.      -23.2     21.6           13193.
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

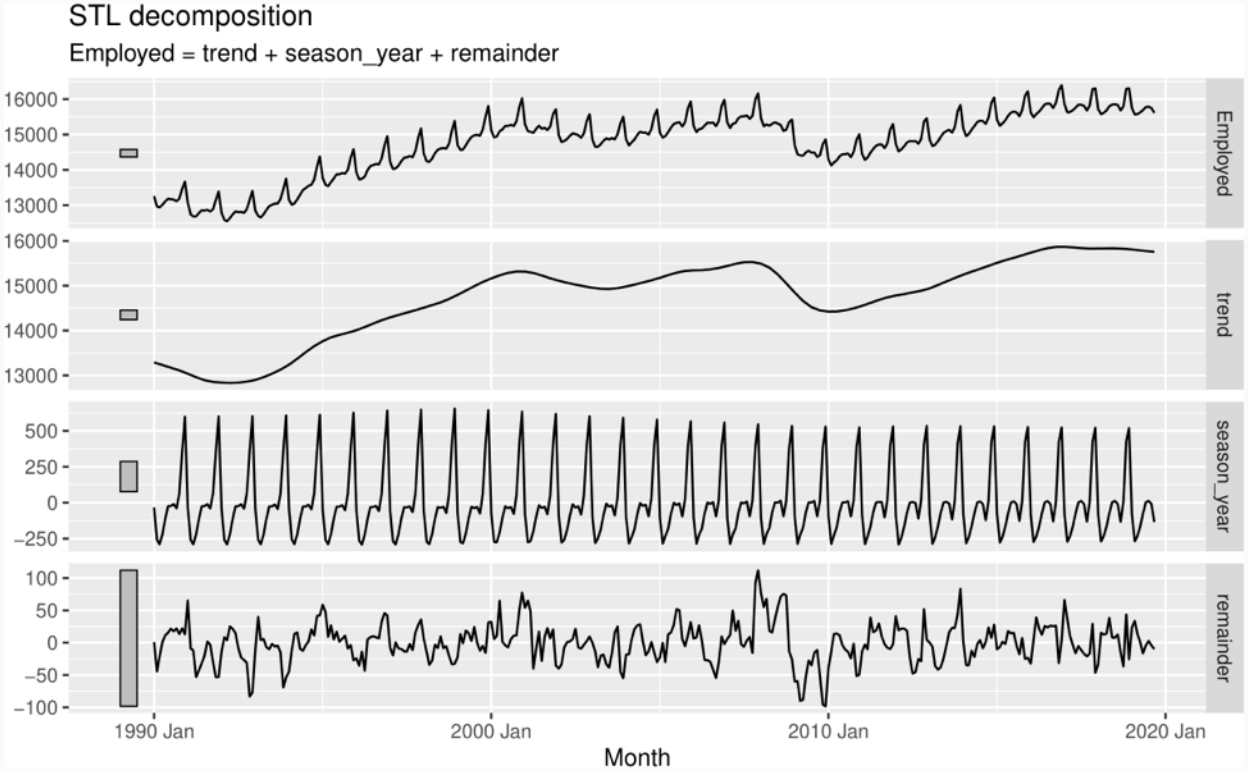
```
us_retail_employment %>%
  autoplot(Employed, color='gray') +
  autolayer(components(dcmp), trend, color='#D55E00') +
  labs(y="Persons (thousands)", title="Total employment in US retail")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

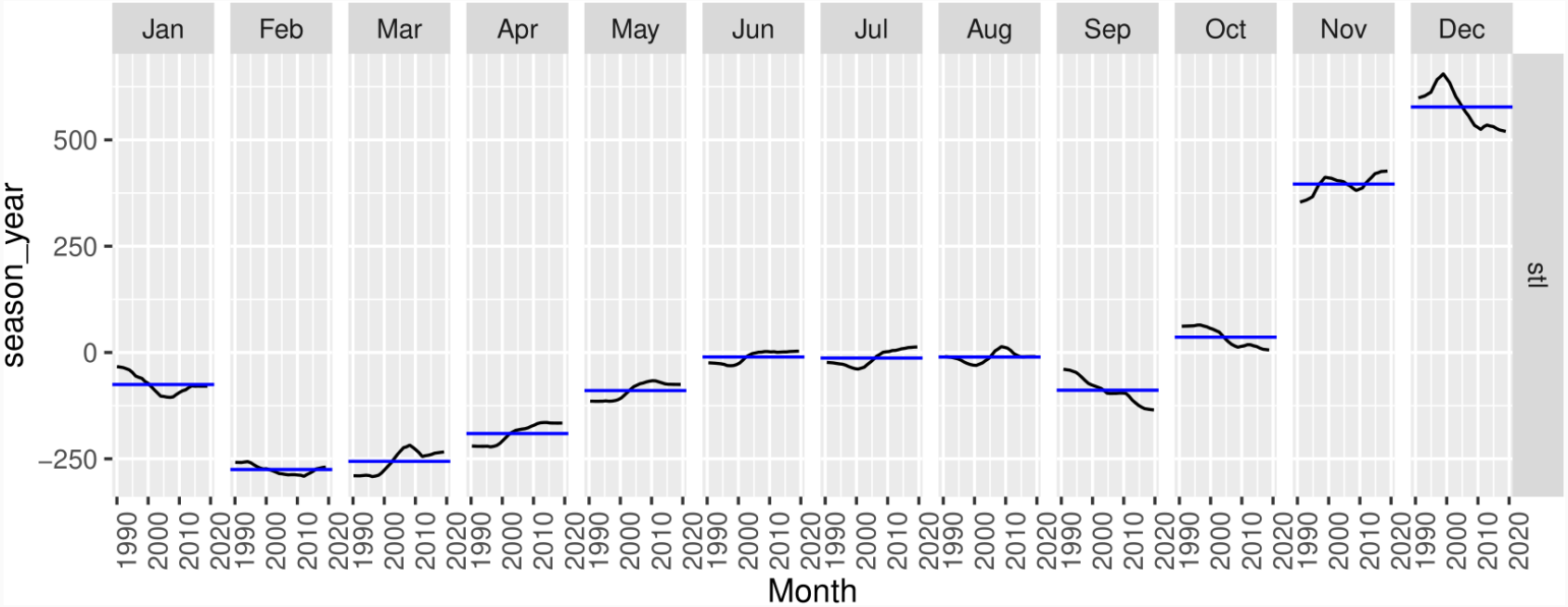
```
components(dcmp) %>% autoplot()
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

```
components(dcmp) %>% gg_subseries(season_year)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Seasonal adjustment

- Useful by-product of decomposition: an easy way to calculate seasonally adjusted data.
- Additive decomposition: seasonally adjusted data given by

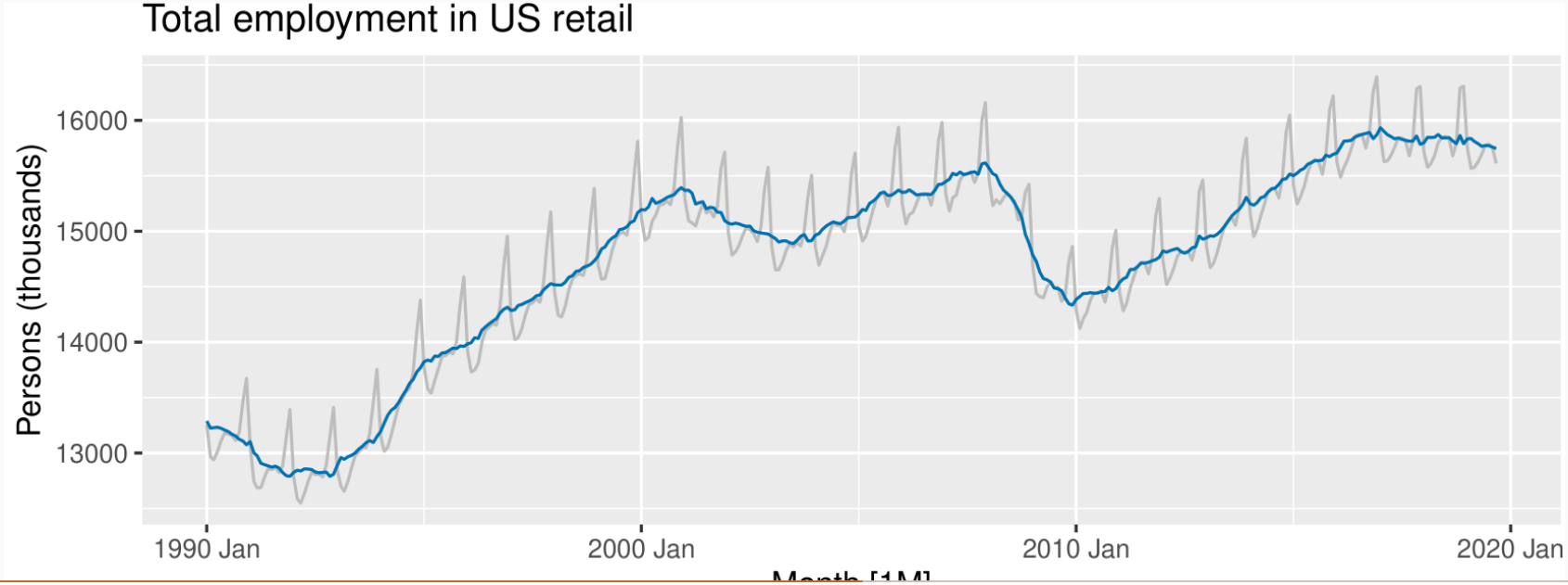
$$y_t - S_t = T_t + R_t$$

- Multiplicative decomposition: seasonally adjusted data given by

$$y_t / S_t = T_t \times R_t$$

US Retail Employment

```
us_retail_employment %>%
  autoplot(Employed, color='gray') +
  autolayer(components(dcmp), season_adjust, color='#0072B2') +
  labs(y="Persons (thousands)", title="Total employment in US retail")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Seasonal adjustment

- We use estimates of S based on past values to seasonally adjust a current value.
- Seasonally adjusted series reflect **remainders** as well as **trend**. Therefore they are not "smooth" and "downturns" or "upturns" can be misleading.
- It is better to use the trend-cycle component to look for turning points.

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

History of time series decomposition

- Classical method originated in 1920s.
- Census II method introduced in 1957. Basis for X-11 method and variants (including X-12-ARIMA, X-13-ARIMA)
- STL method introduced in 1983
- TRAMO/SEATS introduced in 1990s.

National Statistics Offices

- ABS uses X-12-ARIMA
- US Census Bureau uses X-13ARIMA-SEATS
- Statistics Canada uses X-12-ARIMA
- ONS (UK) uses X-12-ARIMA
- EuroStat use X-13ARIMA-SEATS

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

X-11 decomposition

Advantages

- Relatively robust to outliers
- Completely automated choices for trend and seasonal changes
- Very widely tested on economic data over a long period of time.

Disadvantages

- No prediction/confidence intervals
- Ad hoc method with no underlying model
- Only developed for quarterly and monthly data

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Extensions: X-12-ARIMA and X-13-ARIMA

- The X-11, X-12-ARIMA and X-13-ARIMA methods are based on Census II decomposition.
- These allow adjustments for trading days and other explanatory variables.
- Known outliers can be omitted.
- Level shifts and ramp effects can be modelled.
- Missing values estimated and replaced.
- Holiday factors (e.g., Easter, Labour Day) can be estimated.

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

X-13ARIMA-SEATS

Advantages

- Model-based
- Smooth trend estimate
- Allows estimates at end points
- Allows changing seasonality
- Developed for economic data

Disadvantages

- Only developed for quarterly and monthly data

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition

- STL: “Seasonal and Trend decomposition using Loess”
- Very versatile and robust.
- Unlike X-12-ARIMA, STL will handle any type of seasonality.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Robust to outliers
- Not trading day or calendar adjustments.
- Only additive.
- Take logs to get multiplicative decomposition.
- Use Box-Cox transformations to get other decompositions.

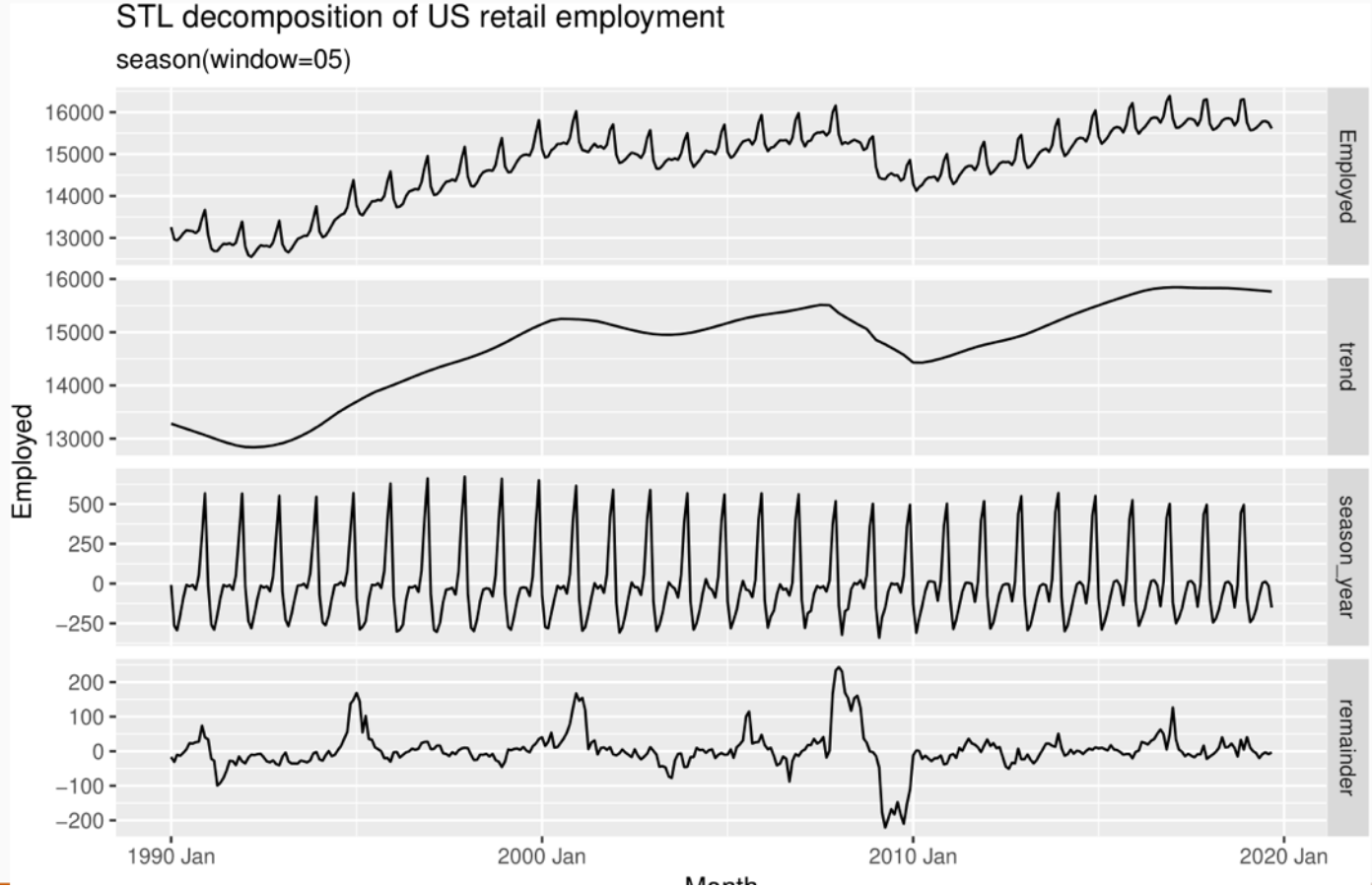
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition

```
us_retail_employment %>%
  model(STL(Employed ~ season(window=9), robust=TRUE)) %>%
  components() %>% autoplot() +
    labs(title = "STL decomposition: US retail employment")
```

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition

```
us_retail_employment %>%
  model(STL(Employed ~ season(window=5))) %>%
  components()

us_retail_employment %>%
  model(STL(Employed ~ trend(window=15) +
                       season(window="periodic"),
            robust = TRUE)
  ) %>% components()
```

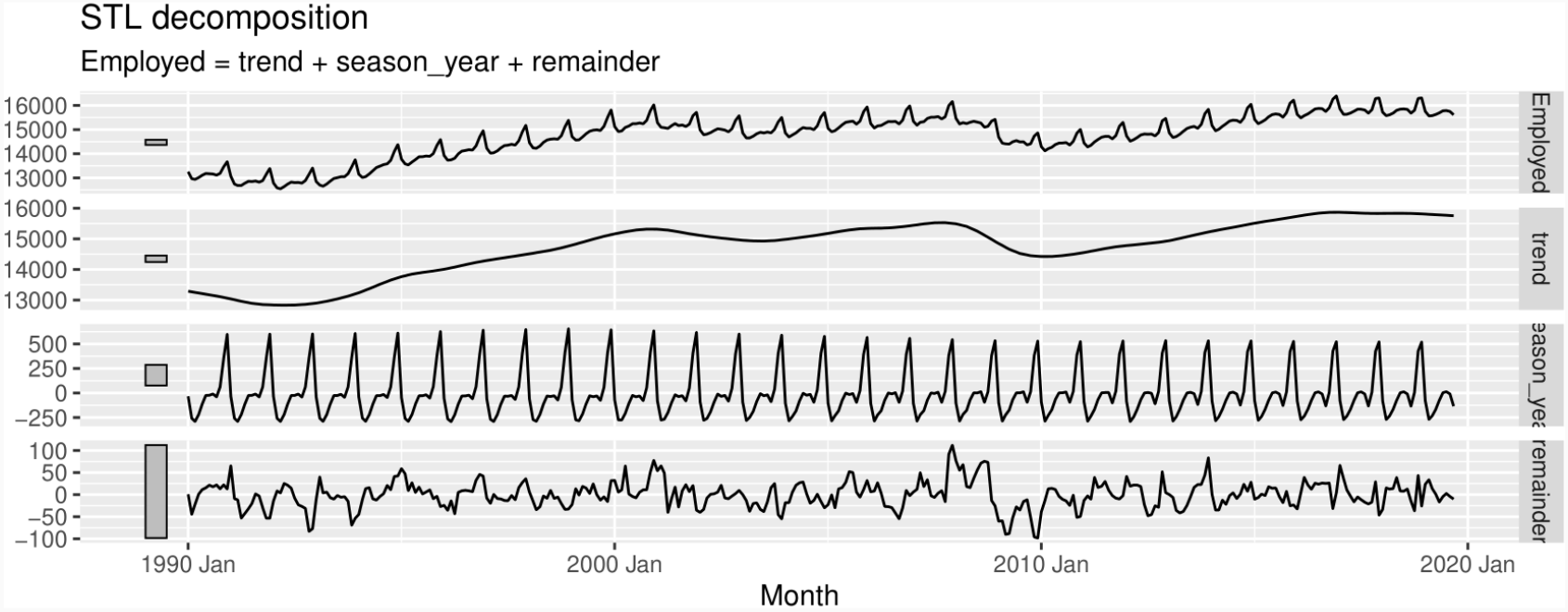
- `trend(window = ?)` controls wiggliness of trend component.
- `season(window = ?)` controls variation on seasonal component.
- `season(window = 'periodic')` is equivalent to an infinite window.

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition

- `STL()` chooses `season(window=13)` by default
- Can include transformations.

```
us_retail_employment %>%
  model(STL(Employed)) %>%
  components() %>% autoplot()
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

STL decomposition

- Algorithm that updates trend and seasonal components iteratively.
- Starts with $\hat{T}_t = 0$
- Uses a mixture of loess and moving averages to successively refine the trend and seasonal estimates.
- The trend window controls loess bandwidth applied to deasonalised values.
- The season window controls loess bandwidth applied to detrended subseries.
- Robustness weights based on remainder.
- Default season `window = 13`
- Default trend `window = nextodd(ceiling((1.5*period)/(1-(1.5/s.window))))`

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

NEWS

LOCATION: Clayton, Vic Change

Just In | Australia | World | Business | Sport | Analysis & Opinion | Fact Check | Programs

BREAKING NEWS Police arrest man in connection with stabbing death of 17-year-old Masa Vukotic in M

Print | Email | Facebook | Twitter | More

Treasurer Joe Hockey calls for answers over Australian Bureau of Statistics jobs data

By Michael Vincent and Simon Frazer

Updated 9 Oct 2014, 12:17pm

Federal Treasurer Joe Hockey says he wants answers to the problems the Australian Bureau of Statistics (ABS) has had with unemployment figures.

Mr Hockey, who is in the US to discuss Australia's G20 agenda, said last month's unemployment figures were "extraordinary".

The rate was 6.1 per cent after jumping to a 12-year high of 6.4 per cent the previous month.

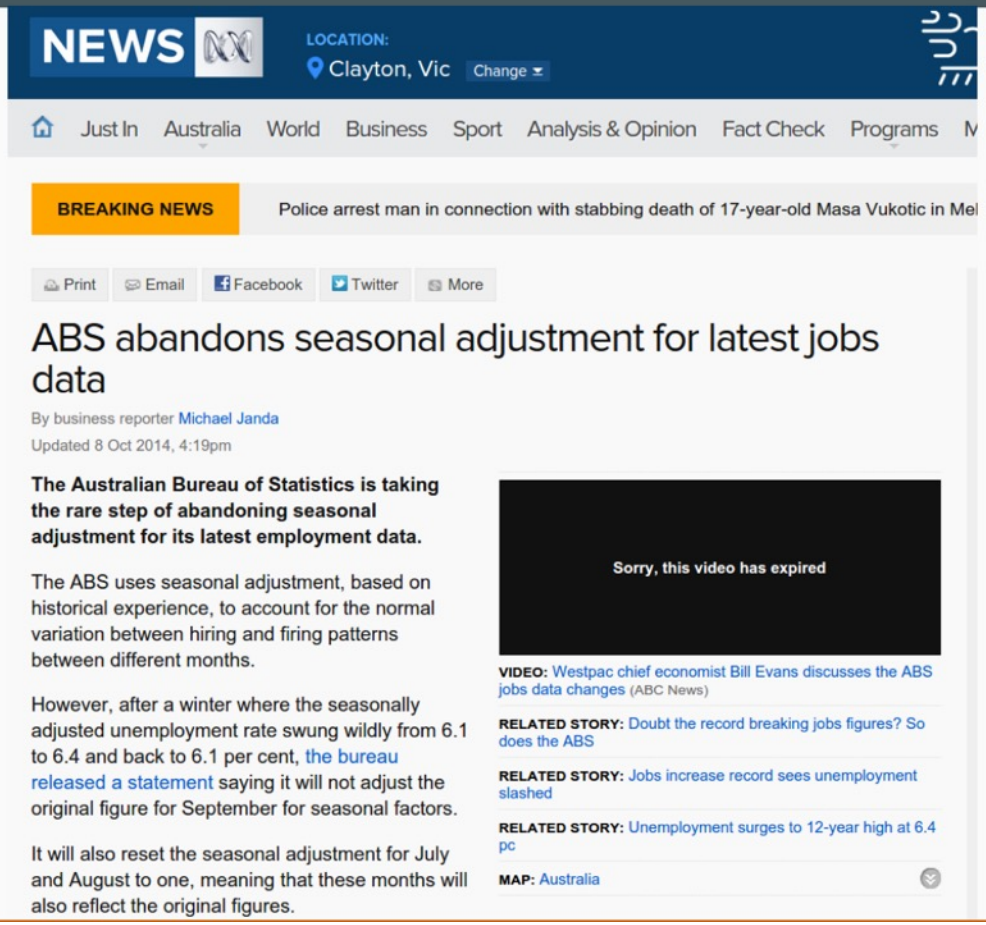
The ABS has now taken the rare step of abandoning seasonal adjustment for its latest employment data.

PHOTO: Joe Hockey says he is unhappy with the volatility of ABS unemployment figures. (AAP: Alan Porritt)

RELATED STORY: ABS abandons seasonal adjustment for latest jobs data

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up



NEWS

LOCATION: Clayton, Vic Change

Just In | Australia | World | Business | Sport | Analysis & Opinion | Fact Check | Programs

BREAKING NEWS Police arrest man in connection with stabbing death of 17-year-old Masa Vukotic in Mel

Print | Email | Facebook | Twitter | More

ABS abandons seasonal adjustment for latest jobs data

By business reporter Michael Janda

Updated 8 Oct 2014, 4:19pm

The Australian Bureau of Statistics is taking the rare step of abandoning seasonal adjustment for its latest employment data.

The ABS uses seasonal adjustment, based on historical experience, to account for the normal variation between hiring and firing patterns between different months.

However, after a winter where the seasonally adjusted unemployment rate swung wildly from 6.1 to 6.4 and back to 6.1 per cent, the bureau released a statement saying it will not adjust the original figure for September for seasonal factors.

It will also reset the seasonal adjustment for July and August to one, meaning that these months will also reflect the original figures.

Sorry, this video has expired

VIDEO: Westpac chief economist Bill Evans discusses the ABS jobs data changes (ABC News)

RELATED STORY: Doubt the record breaking jobs figures? So does the ABS

RELATED STORY: Jobs increase record sees unemployment slashed

RELATED STORY: Unemployment surges to 12-year high at 6.4 pc

MAP: Australia

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

ABS jobs and unemployment figures – key questions answered by an expert

A professor of statistics at Monash University explains exactly what is seasonal adjustment, why it matters and what went wrong in the July and August figures

School leavers come on to the jobs market at the same time, causing a seasonal fluctuation. Photograph: Brian Snyder/Reuters

The Australian Bureau of Statistics has retracted its seasonally adjusted employment data for July and August, which recorded huge swings in the jobless rate. The ABS is also planning to review the methods it uses for seasonal adjustment to ensure its figures are as accurate as possible. Rob Hyndman, a professor of statistics at Monash University and member of the bureau's methodology advisory board, answers our questions:

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

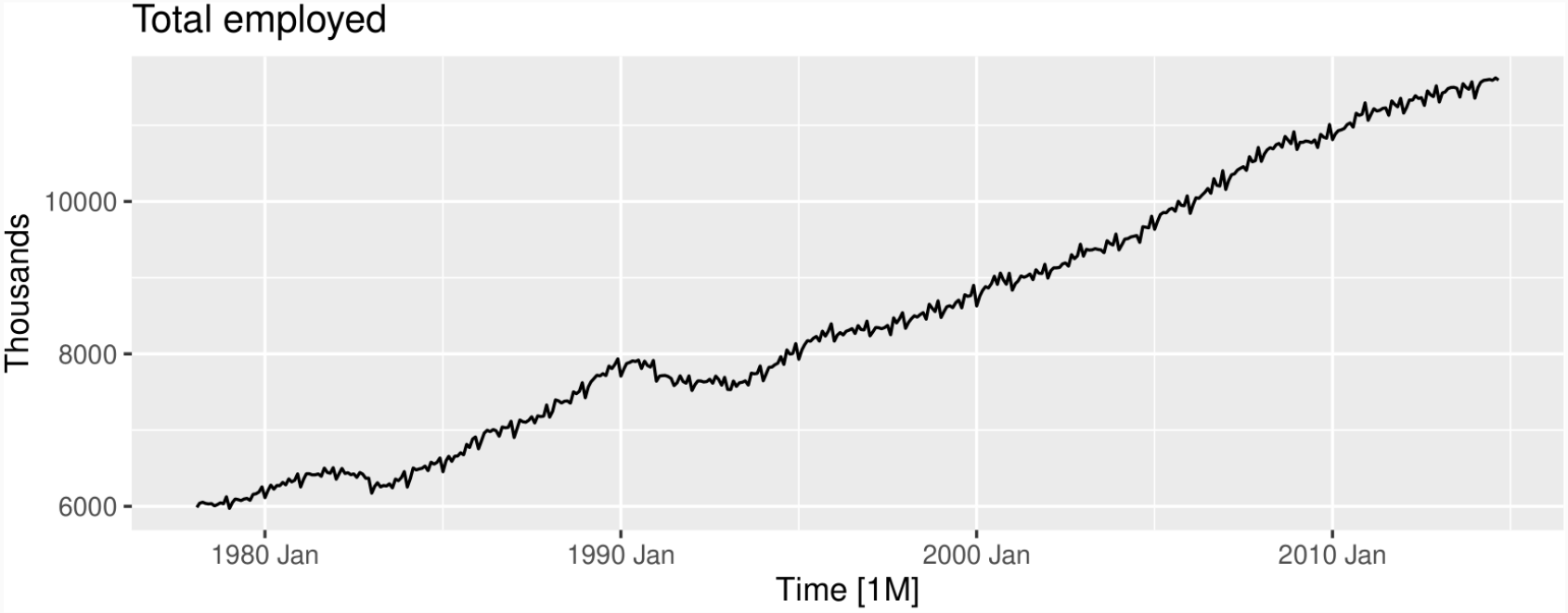
```
employed
```

```
## # A tsibble: 440 x 4 [1M]
##        Time Month  Year Employed
##       <mth> <ord> <dbl>    <dbl>
##  1 1978 Feb Feb    1978    5986.
##  2 1978 Mar Mar    1978    6041.
##  3 1978 Apr Apr    1978    6054.
##  4 1978 May May    1978    6038.
##  5 1978 Jun Jun    1978    6031.
##  6 1978 Jul Jul    1978    6036.
##  7 1978 Aug Aug    1978    6005.
##  8 1978 Sep Sep    1978    6024.
##  9 1978 Oct Oct    1978    6046.
## 10 1978 Nov Nov    1978    6034.
## # ... with 430 more rows
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

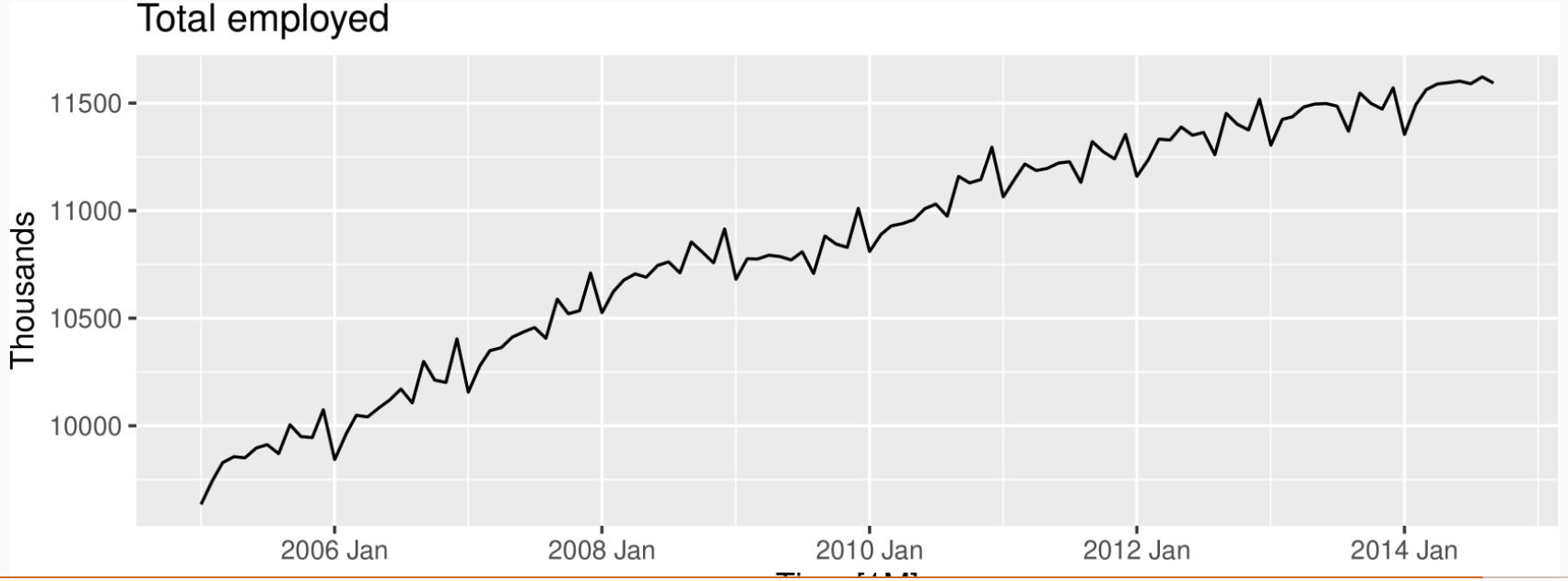
```
employed %>%
  autoplot(Employed) +
  labs(title = "Total employed", y = "Thousands")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

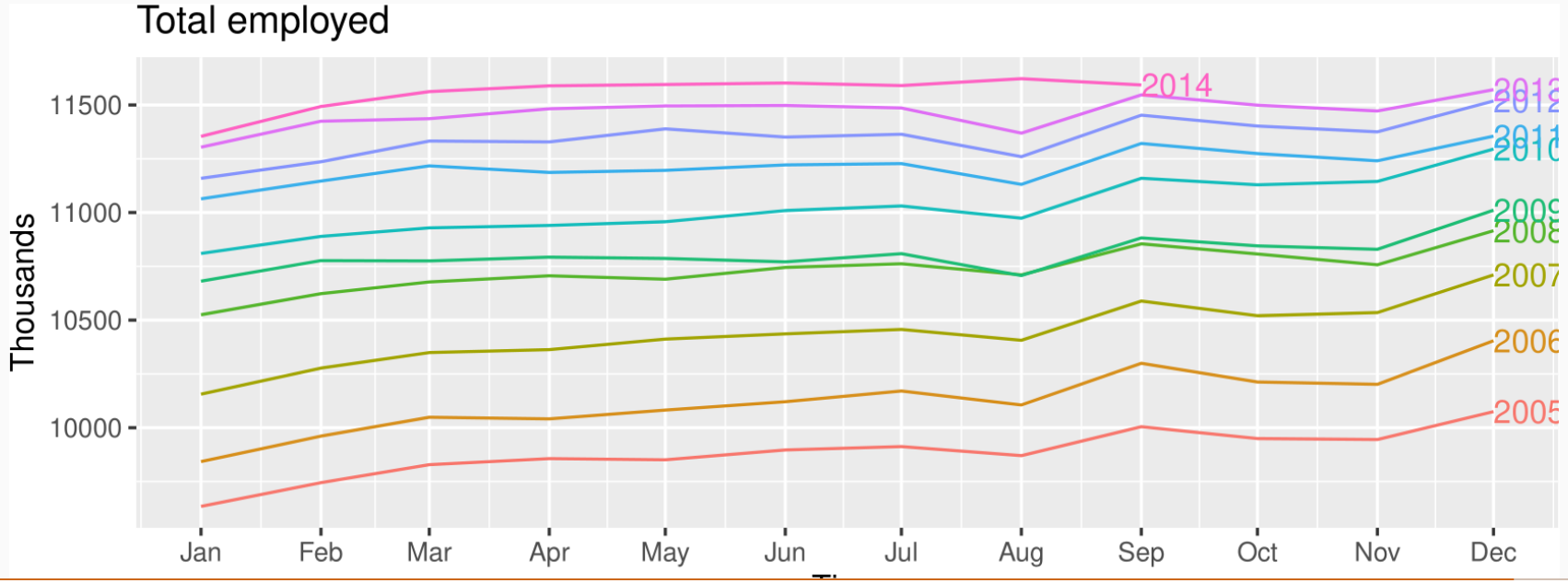
```
employed %>%
  filter(Year >= 2005) %>%
  autoplot(Employed) +
  labs(title = "Total employed", y = "Thousands")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

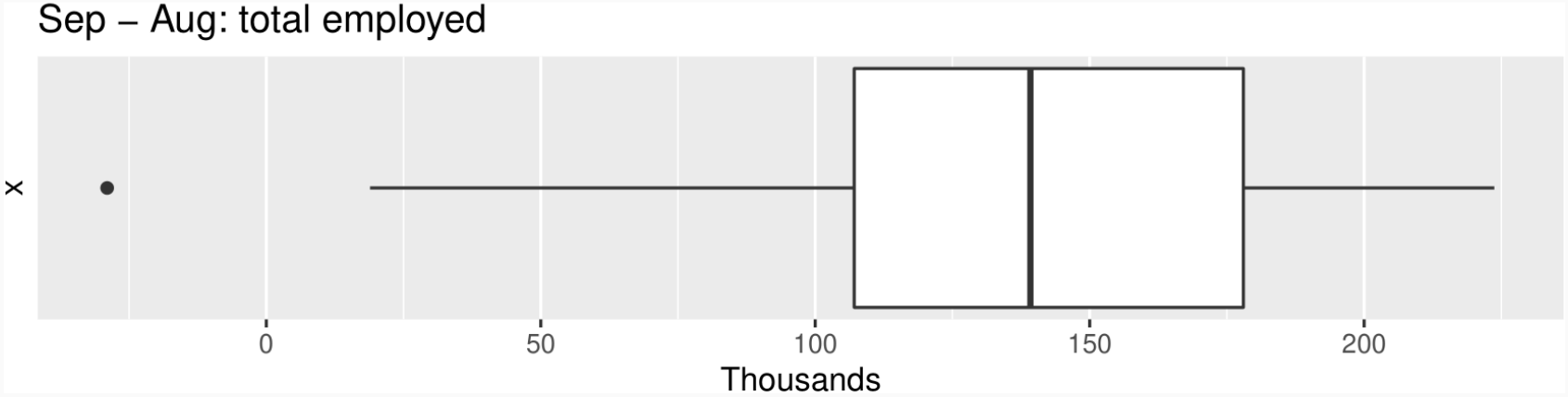
```
employed %>%
  filter(Year >= 2005) %>%
  gg_season(Employed, label = "right") +
  labs(title = "Total employed", y = "Thousands")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

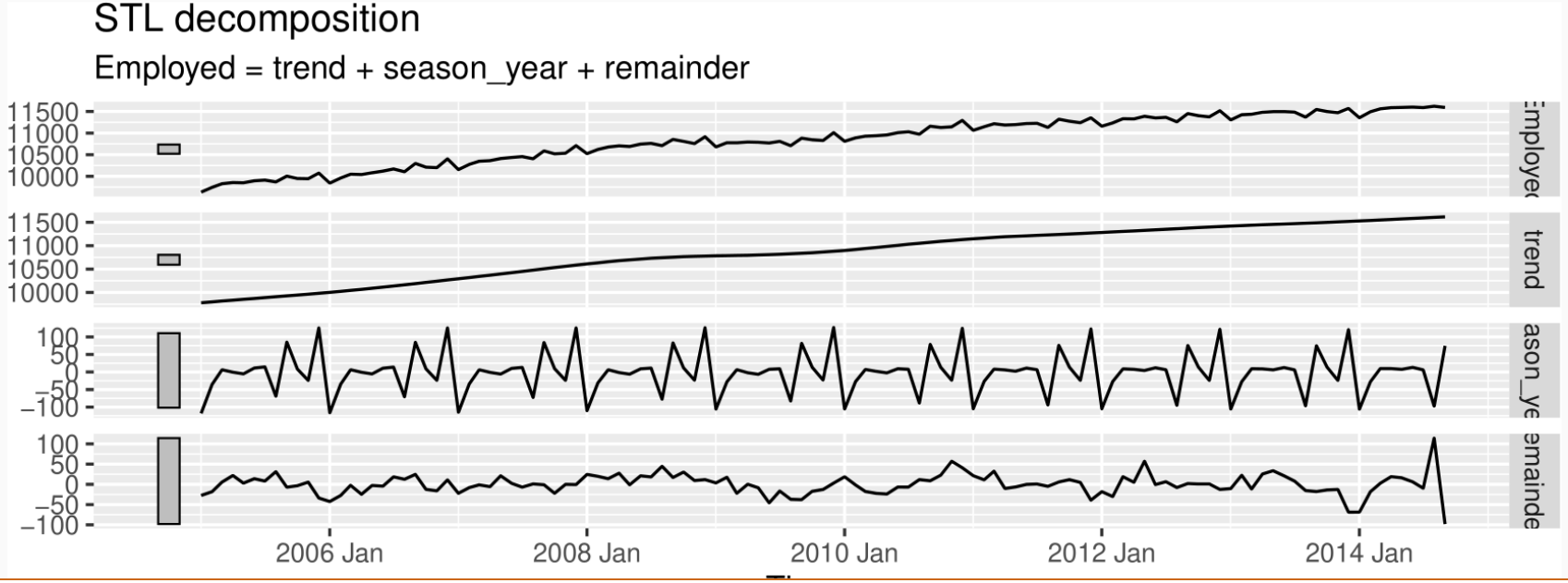
```
employed %>%
  mutate(diff = difference(Employed)) %>%
  filter(Month == "Sep") %>%
  ggplot(aes(y = diff, x = 1)) +
  geom_boxplot() + coord_flip() +
  labs(title = "Sep - Aug: total employed", y = "Thousands") +
  scale_x_continuous(breaks = NULL, labels = NULL)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

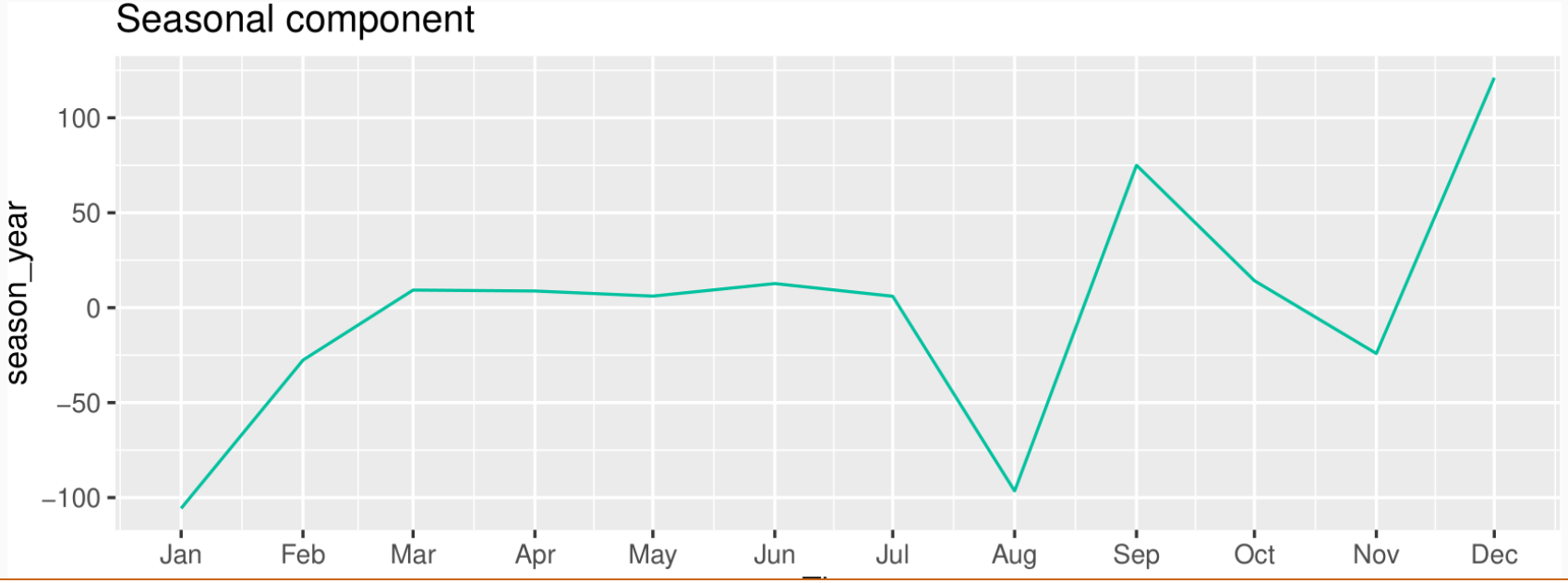
```
dcmp <- employed %>%
  filter(Year >= 2005) %>%
  model(stl = STL(Employed ~ season(window = 11), robust = TRUE))
components(dcmp) %>% autoplot()
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

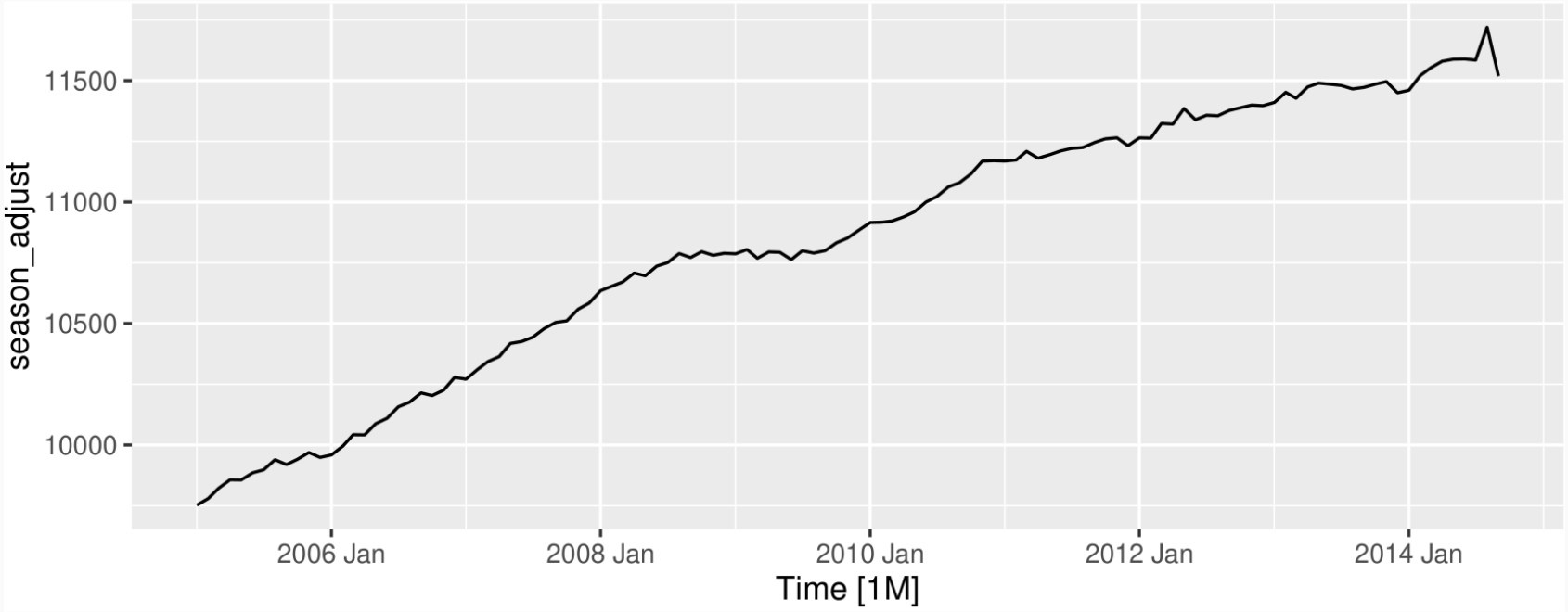
```
components(dcmp) %>%
  filter(year(Time) == 2013) %>%
  gg_season(season_year) +
  labs(title = "Seasonal component") + guides(colour = "none")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

```
components(dcmp) %>%
  as_tsibble() %>%
  autoplot(season_adjust)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

The ABS stuff-up

- August 2014 employment numbers higher than expected.
- Supplementary survey usually conducted in August for employed people.
- Most likely, some employed people were claiming to be unemployed in August to avoid supplementary questions.
- Supplementary survey not run in 2014, so no motivation to lie about employment.
- In previous years, seasonal adjustment fixed the problem.
- The ABS has now adopted a new method to avoid the bias.

Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. Distributional forecasts and prediction intervals
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

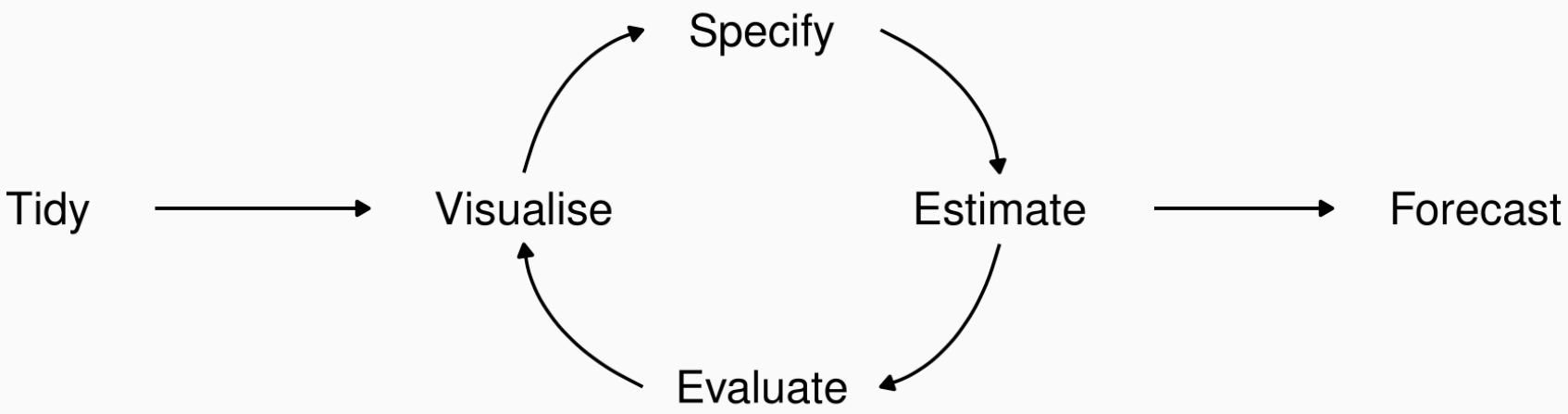
A tidy forecasting workflow

The process of producing forecasts can be split up into a few fundamental steps.

1. Preparing data
2. Data visualisation
3. Specifying a model
4. Model estimation
5. Accuracy & performance evaluation
6. Producing forecasts

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

A tidy forecasting workflow



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Data preparation (tidy)

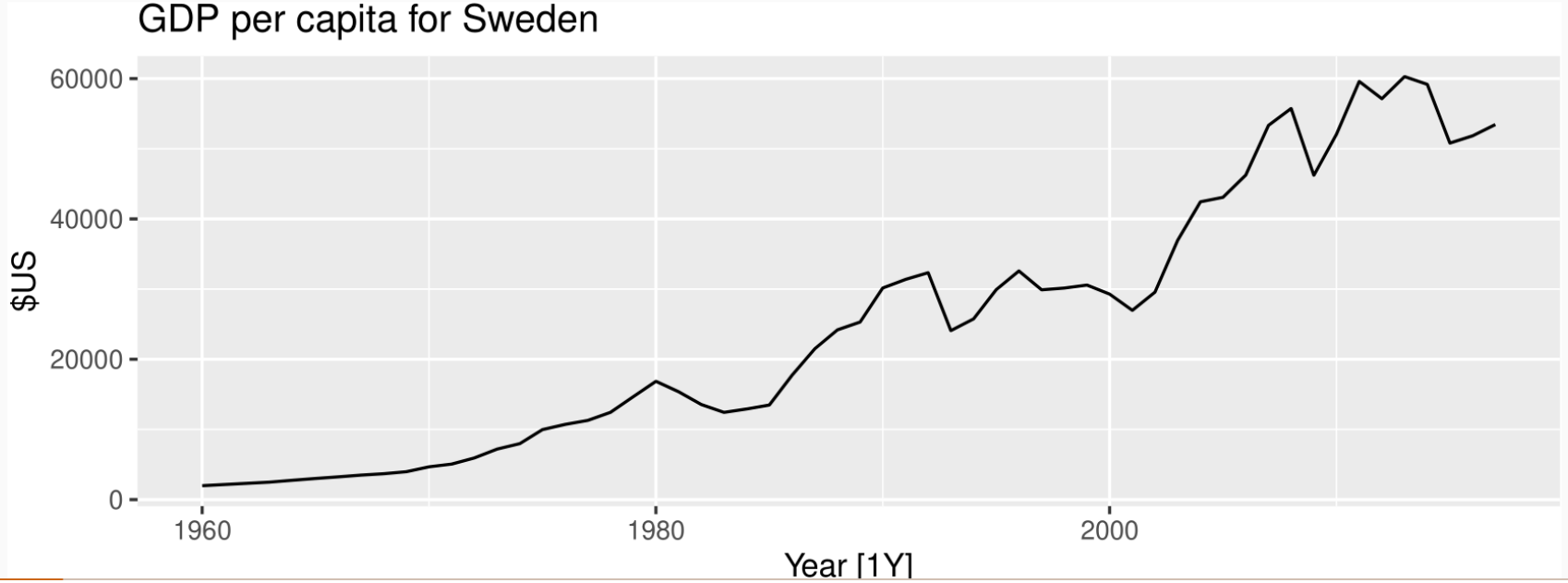
```
gdppc <- global_economy %>%
  mutate(GDP_per_capita = GDP/Population) %>%
  select(Year, Country, GDP, Population, GDP_per_capita)
gdppc
```

```
## # A tsibble: 15,150 x 5 [1Y]
## # Key:       Country [263]
##     Year Country              GDP Population GDP_per_capita
##    <dbl> <fct>                <dbl>      <dbl>          <dbl>
##  1  1960 Afghanistan  537777811.    8996351           59.8
##  2  1961 Afghanistan  548888896.    9166764           59.9
##  3  1962 Afghanistan  546666678.    9345868           58.5
##  4  1963 Afghanistan  751111191.    9533954           78.8
##  5  1964 Afghanistan  800000044.    9731361           82.2
##  6  1965 Afghanistan 1006666638.    9938414          101.
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Data visualisation

```
gdppc %>%
  filter(Country=="Sweden") %>%
  autoplot(GDP_per_capita) +
    labs(title = "GDP per capita for Sweden", y = "$US")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Model estimation

The `model()` function trains models to data.

```
fit <- gdppc %>%
  model(trend_model = TSLM(GDP_per_capita ~ trend()))
fit
```

```
## # A mable: 263 x 2
## # Key:     Country [263]
##    Country              trend_model
##    <fct>                    <model>
##  1 Afghanistan               <TSLM>
##  2 Albania                   <TSLM>
##  3 Algeria                   <TSLM>
##  4 American Samoa            <TSLM>
##  5 Andorra                   <TSLM>
```

A `mable` is a model table, each cell corresponds to a fitted model.

Producing forecasts

```
fit %>% forecast(h = "3 years")
```

```
## # A fable: 789 x 5 [1Y]
## # Key:     Country, .model [263]
##    Country        .model       Year   GDP_per_capita  .mean
##    <fct>          <chr>       <dbl>           <dist>  <dbl>
##  1 Afghanistan    trend_model  2018     N(526, 9653)   526.
##  2 Afghanistan    trend_model  2019     N(534, 9689)   534.
##  3 Afghanistan    trend_model  2020     N(542, 9727)   542.
##  4 Albania        trend_model  2018  N(4716, 476419)  4716.
##  5 Albania        trend_model  2019  N(4867, 481086)  4867.
##  6 Albania        trend_model  2020  N(5018, 486012)  5018.
##  7 Algeria        trend_model  2018  N(4410, 643094)  4410.
##  8 Algeria        trend_model  2019  N(4489, 645311)  4489.
```

Producing forecasts

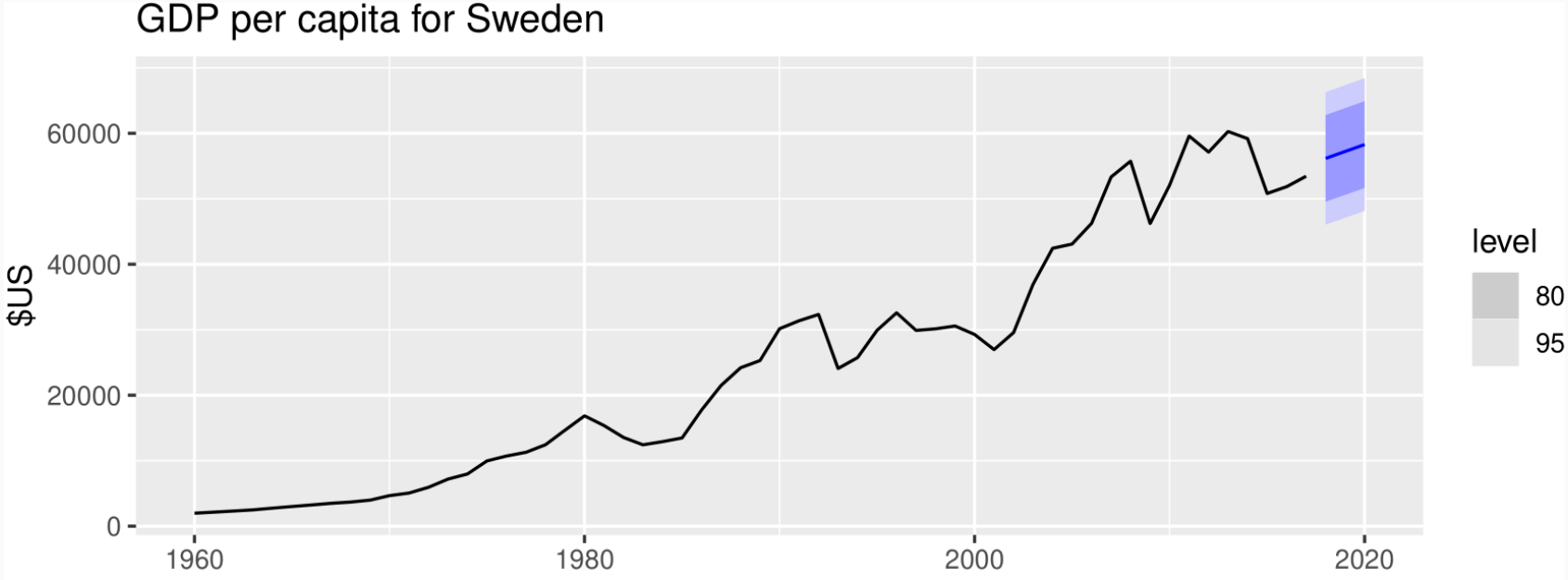
```
fit %>% forecast(h = "3 years")
```

A `fable` is a forecast table with point forecasts and distributions.

```
## # A fable: 789 x 5 [1Y]
## # Key:     Country, .model [263]
##    Country        .model       Year   GDP_per_capita  .mean
##    <fct>          <chr>       <dbl>           <dist>  <dbl>
##  1 Afghanistan    trend_model  2018     N(526, 9653)   526.
##  2 Afghanistan    trend_model  2019     N(534, 9689)   534.
##  3 Afghanistan    trend_model  2020     N(542, 9727)   542.
##  4 Albania        trend_model  2018  N(4716, 476419)  4716.
##  5 Albania        trend_model  2019  N(4867, 481086)  4867.
##  6 Albania        trend_model  2020  N(5018, 486012)  5018.
##  7 Algeria        trend_model  2018  N(4410, 643094)  4410.
##  8 Algeria        trend_model  2019  N(4489, 645311)  4489.
```

Visualising forecasts

```
fit %>% forecast(h = "3 years") %>%
  filter(Country=="Sweden") %>%
  autoplot(gdppc) +
    labs(title = "GDP per capita for Sweden", y = "$US")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

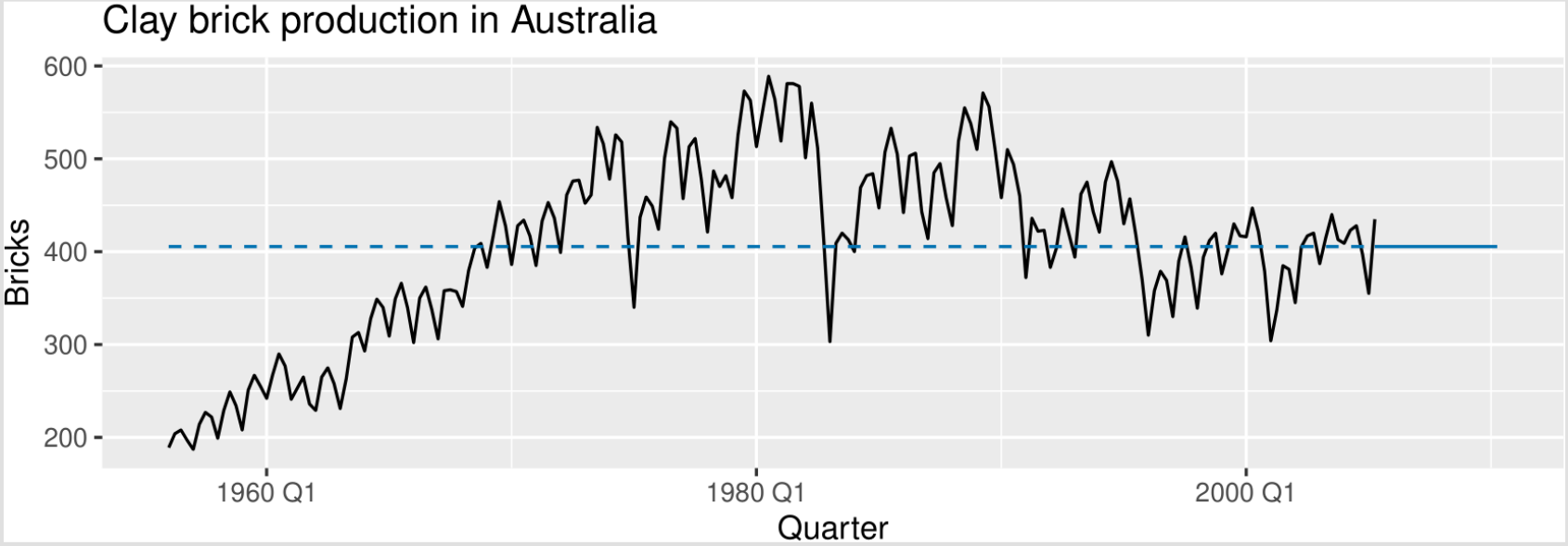
Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. Distributional forecasts and prediction intervals
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Some simple forecasting methods

`MEAN(y)`: Average method

- Forecast of all future values is equal to mean of historical data $\{y_1, \ldots, y_T\}$.
- Forecasts: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$

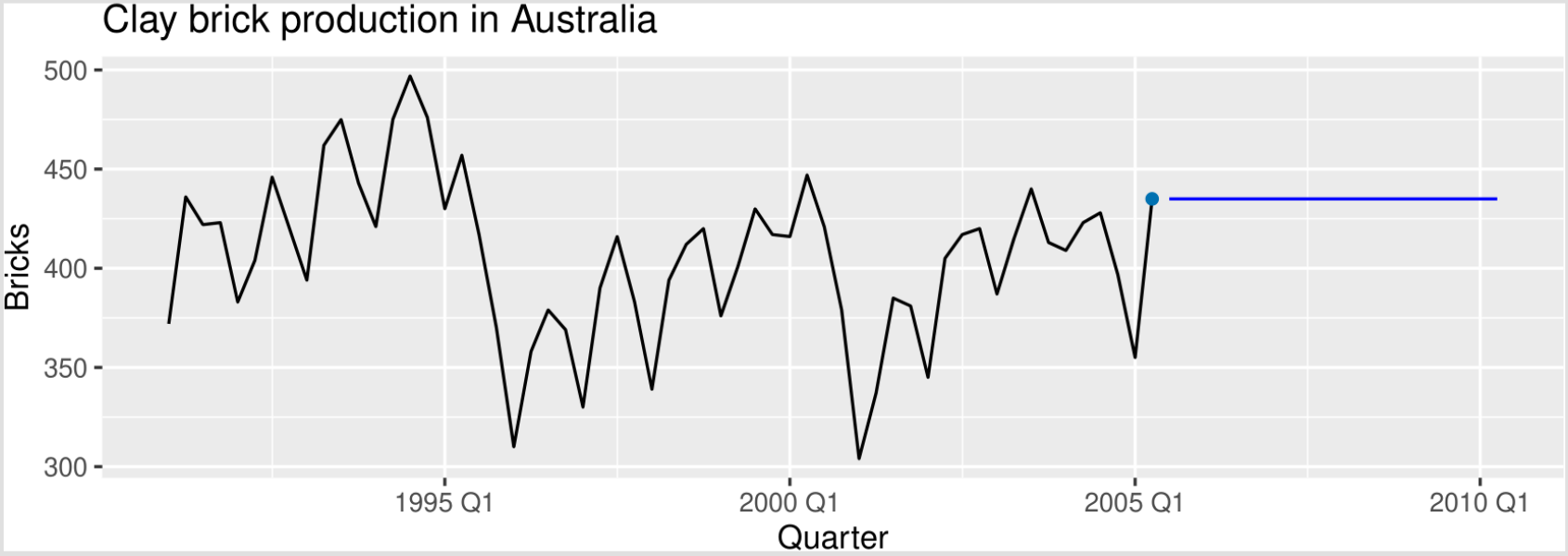


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Some simple forecasting methods

`NAIVE(y)`: Naïve method

- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{T+h|T} = y_T$.
- Consequence of efficient market hypothesis.

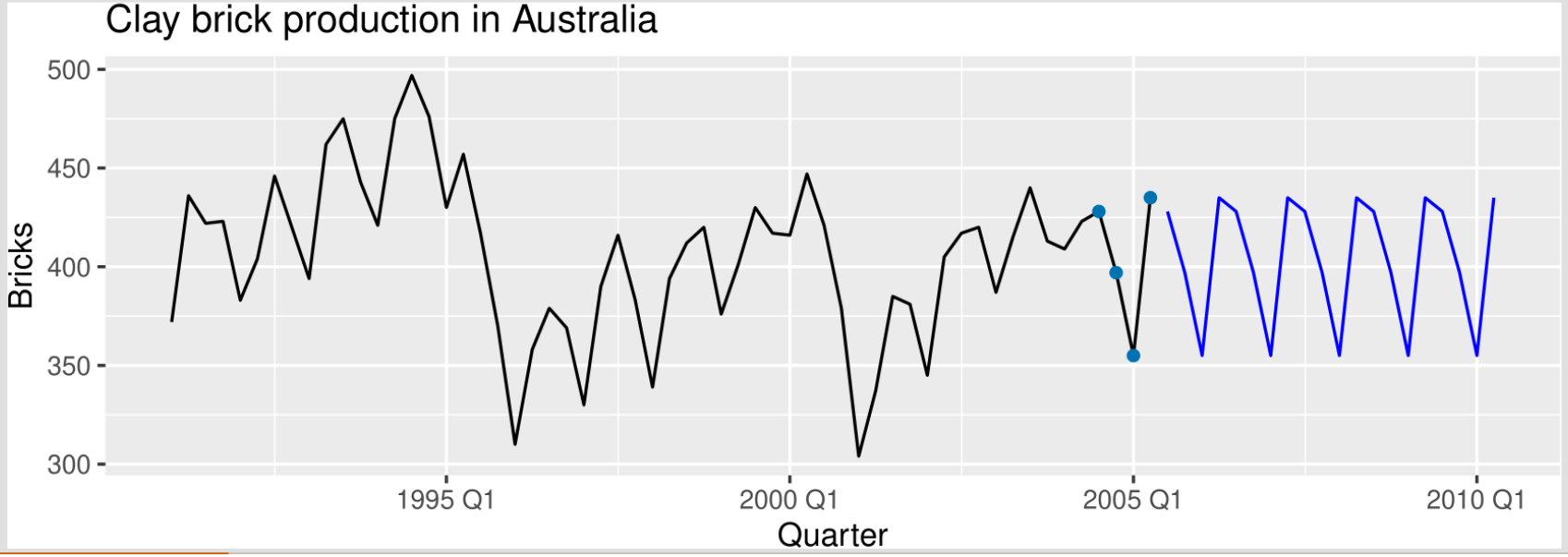


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Some simple forecasting methods

`SNAIVE(y ~ lag(m))`: Seasonal naïve method

- Forecasts equal to last value from same season.
- Forecasts: $\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$, where m = seasonal period and k is the integer part of $(h-1)/m$.



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Some simple forecasting methods

`RW(y ~ drift())`: Drift method

- Forecasts equal to last value plus average change.
- Forecasts:

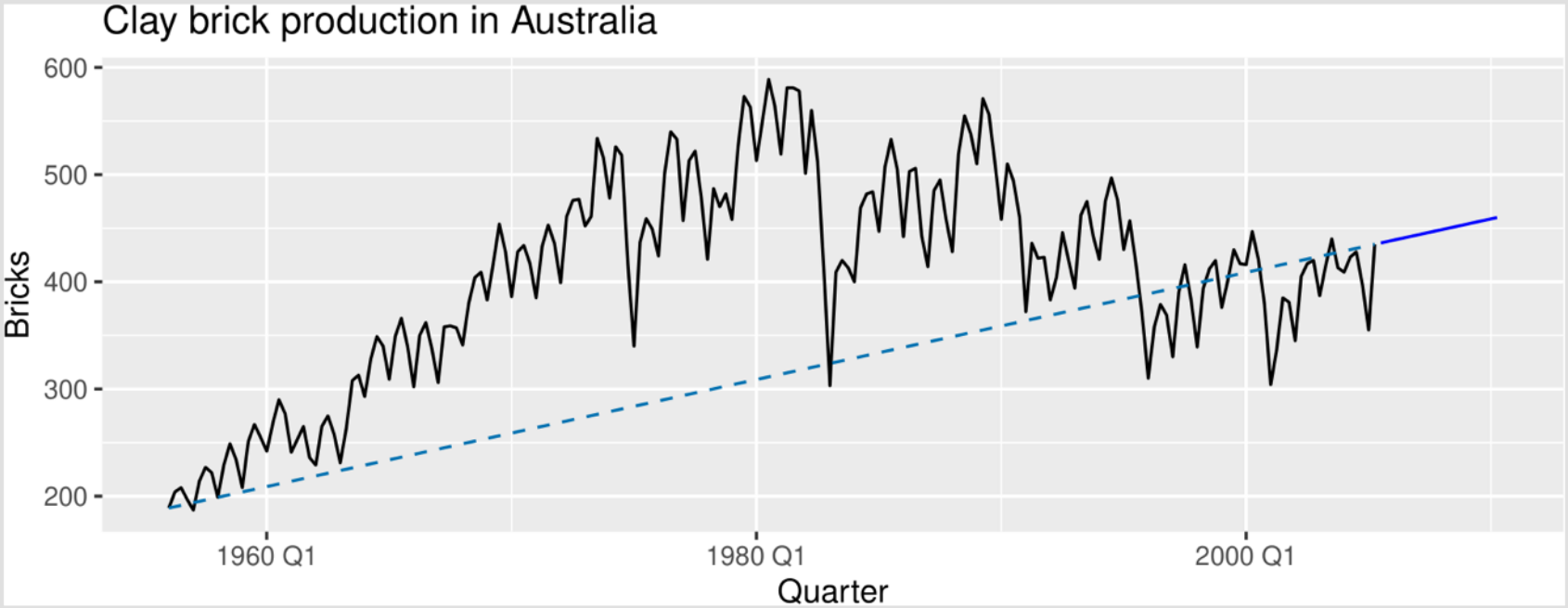
$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})$$

$$= y_T + \frac{h}{T-1}(y_T - y_1).$$

- Equivalent to extrapolating a line drawn between first and last observations.

Some simple forecasting methods

Drift method



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Model fitting

The `model()` function trains models to data.

```
brick_fit <-  aus_production %>%
  filter(!is.na(Bricks)) %>%
  model(
    Seasonal_naive = SNAIVE(Bricks),
    Naive = NAIVE(Bricks),
    Drift = RW(Bricks ~ drift()),
    Mean = MEAN(Bricks)
  )
```

```
## # A mable: 1 x 4
##   Seasonal_naive   Naive          Drift    Mean
##           <model> <model>        <model> <model>
## 1       <SNAIVE> <NAIVE> <RW w/ drift>  <MEAN>
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

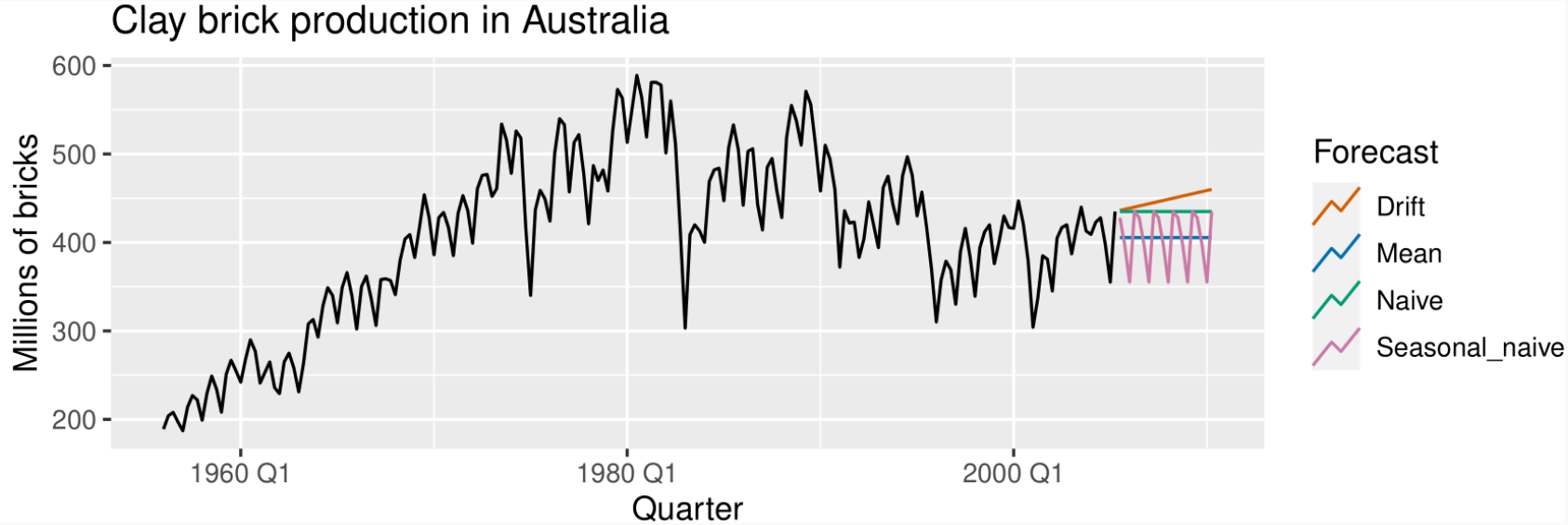
Producing forecasts

```
brick_fc <- brick_fit %>%
  forecast(h = "5 years")
```

```
## # A fable: 80 x 4 [1Q]
## # Key:     .model [4]
##   .model         Quarter       Bricks .mean
##   <chr>            <qtr>       <dist> <dbl>
## 1 Seasonal_naive 2005 Q3 N(428, 2336)   428
## 2 Seasonal_naive 2005 Q4 N(397, 2336)   397
## 3 Seasonal_naive 2006 Q1 N(355, 2336)   355
## 4 Seasonal_naive 2006 Q2 N(435, 2336)   435
## # ... with 76 more rows
```

Visualising forecasts

```
brick_fc %>%
  autoplot(aus_production, level = NULL) +
  labs(title = "Clay brick production in Australia",
       y = "Millions of bricks") +
  guides(colour = guide_legend(title = "Forecast"))
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

```
# Extract training data
fb_stock <- gafa_stock %>%
  filter(Symbol == "FB") %>%
  mutate(trading_day = row_number()) %>%
  update_tsibble(index=trading_day, regular=TRUE)

# Specify, estimate and forecast
fb_stock %>%
  model(
    Mean = MEAN(Close),
    Naive = NAIVE(Close),
    Drift = RW(Close ~ drift())
  ) %>%
  forecast(h=42) %>%
  autoplot(fb_stock, level = NULL) +
  labs(title = "Facebook closing stock price", y="$US") +
  guides(colour=guide_legend(title="Forecast"))
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Outline

1 A tidy forecasting workflow

2 Some simple forecasting methods

3 Residual diagnostics

4 Distributional forecasts and prediction intervals

5 Forecasting with transformations

6 Forecasting and decomposition

7 Evaluating forecast accuracy

Fitted values

- $\hat{y}_{t|t-1}$ is the forecast of y_t based on observations $y_1, \ldots, y_{t-1}$.
- We call these "fitted values".
- Sometimes drop the subscript: $\hat{y}_t \equiv \hat{y}_{t|t-1}$.
- Often not true forecasts since parameters are estimated on all data.

For example:

- $\hat{y}_t = \bar{y}$ for average method.
- $\hat{y}_t = y_{t-1} + (y_T - y_1)/(T-1)$ for drift method.

Forecasting residuals

Residuals in forecasting: difference between observed value and its fitted value: $e_t = y_t - \hat{y}_{t|t-1}$.

Assumptions

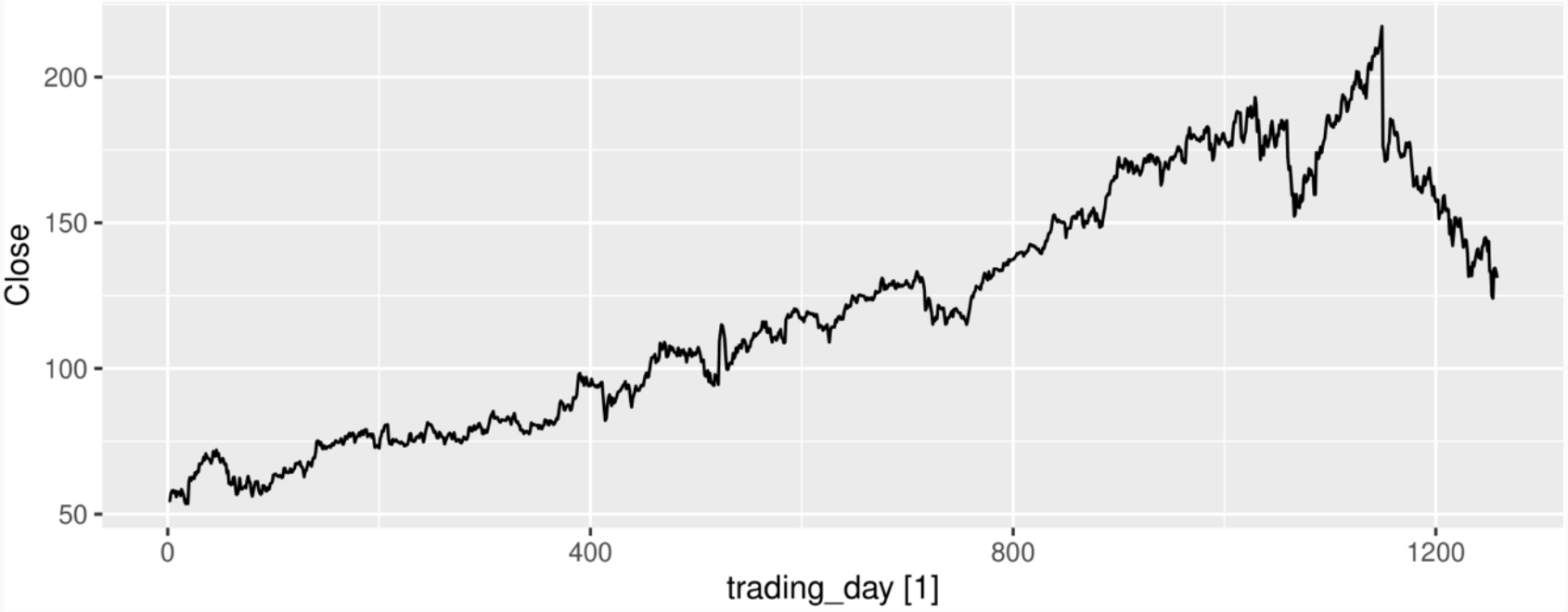
1. $\{e_t\}$ uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero. If they don't, then forecasts are biased.

Useful properties (for distributions & prediction intervals)

3. $\{e_t\}$ have constant variance.
4. $\{e_t\}$ are normally distributed.

Facebook closing stock price

```
fb_stock %>% autoplot(Close)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

```
fit <- fb_stock %>% model(NAIVE(Close))
augment(fit)
```

```
## # A tsibble: 1,258 x 7 [1]
## # Key:       Symbol, .model [1]
##    Symbol .model       trading_day Close .fitted .resid .innov
##    <chr>  <chr>              <int> <dbl>   <dbl>  <dbl>  <dbl>
##  1 FB     NAIVE(Close)           1  54.7    NA    NA     NA
##  2 FB     NAIVE(Close)           2  54.6    54.7 -0.150 -0.150
##  3 FB     NAIVE(Close)           3  57.2    54.6  2.64   2.64
##  4 FB     NAIVE(Close)           4  57.9    57.2  0.720  0.720
##  5 FB     NAIVE(Close)           5  58.2    57.9  0.310  0.310
##  6 FB     NAIVE(Close)           6  57.2    58.2 -1.01  -1.01
##  7 FB     NAIVE(Close)           7  57.9    57.2  0.720  0.720
##  8 FB     NAIVE(Close)           8  55.9    57.9 -2.03  -2.03
##  9 FB     NAIVE(Close)           9  57.7    55.9  1.83   1.83
## 10 FB     NAIVE(Close)          10  57.6    57.7 -0.140 -0.140
## # ... with 1,248 more rows
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

```
fit <- fb_stock %>% model(NAIVE(Close))
augment(fit)
```

```
## # A tsibble: 1,258 x 7 [1]
## # Key:       Symbol, .model [1]
##    Symbol .model       trading_day Close .fitted .resid .innov
##    <chr>  <chr>              <int> <dbl>   <dbl>  <dbl>  <dbl>
##  1 FB     NAIVE(Close)           1  54.7    NA    NA      NA
##  2 FB     NAIVE(Close)           2  54.6    54.7 -0.150 -0.150
##  3 FB     NAIVE(Close)           3  57.2    54.6  2.64   2.64
##  4 FB     NAIVE(Close)           4  57.9    57.2  0.720  0.720
##  5 FB     NAIVE(Close)           5  58.2    57.9  0.310  0.310
##  6 FB     NAIVE(Close)           6  57.2    58.2 -1.01  -1.01
##  7 FB     NAIVE(Close)           7  57.9    57.2  0.720  0.720
##  8 FB     NAIVE(Close)           8  55.9    57.9 -2.03  -2.03
##  9 FB     NAIVE(Close)           9  57.7    55.9  1.83   1.83
## 10 FB     NAIVE(Close)          10  57.6    57.7 -0.140 -0.140
## # ... with 1,248 more rows
```

$\hat{y}_{t|t-1}$ e_t

Naïve forecasts:

$$\hat{y}_{t|t-1} = y_{t-1}$$

$$e_t = y_t - \hat{y}_{t|t-1} = y_t - y_{t-1}$$

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

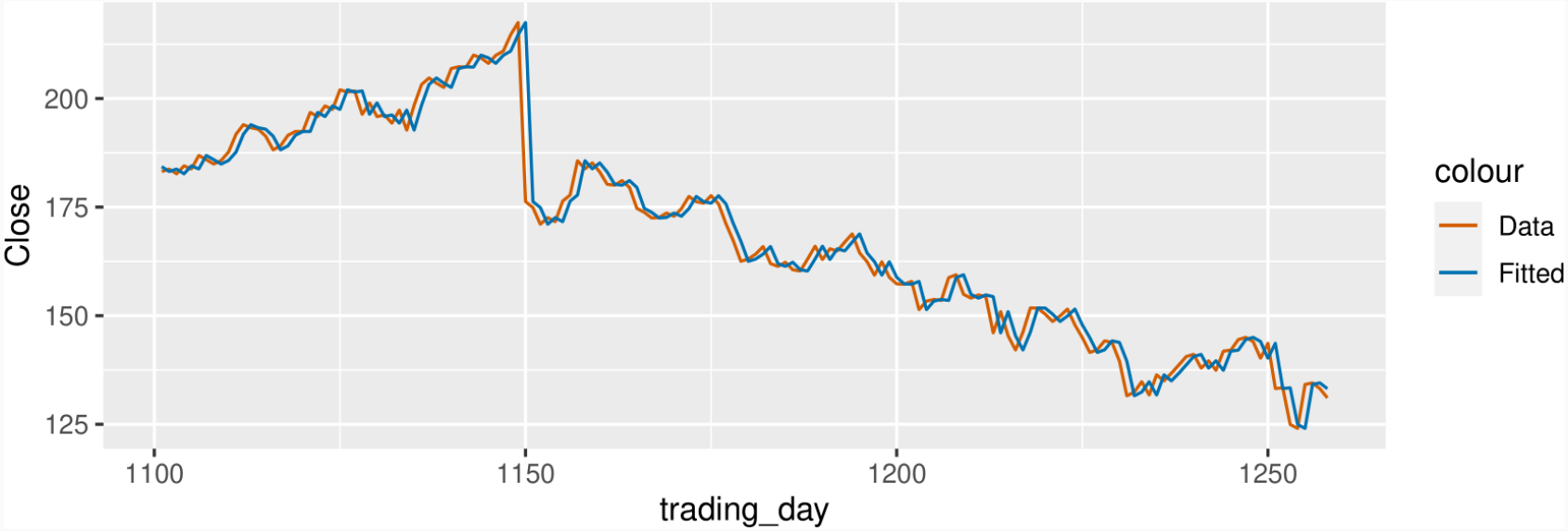
Facebook closing stock price

```
augment(fit) %>%
  ggplot(aes(x = trading_day)) +
  geom_line(aes(y = Close, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted"))
```

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

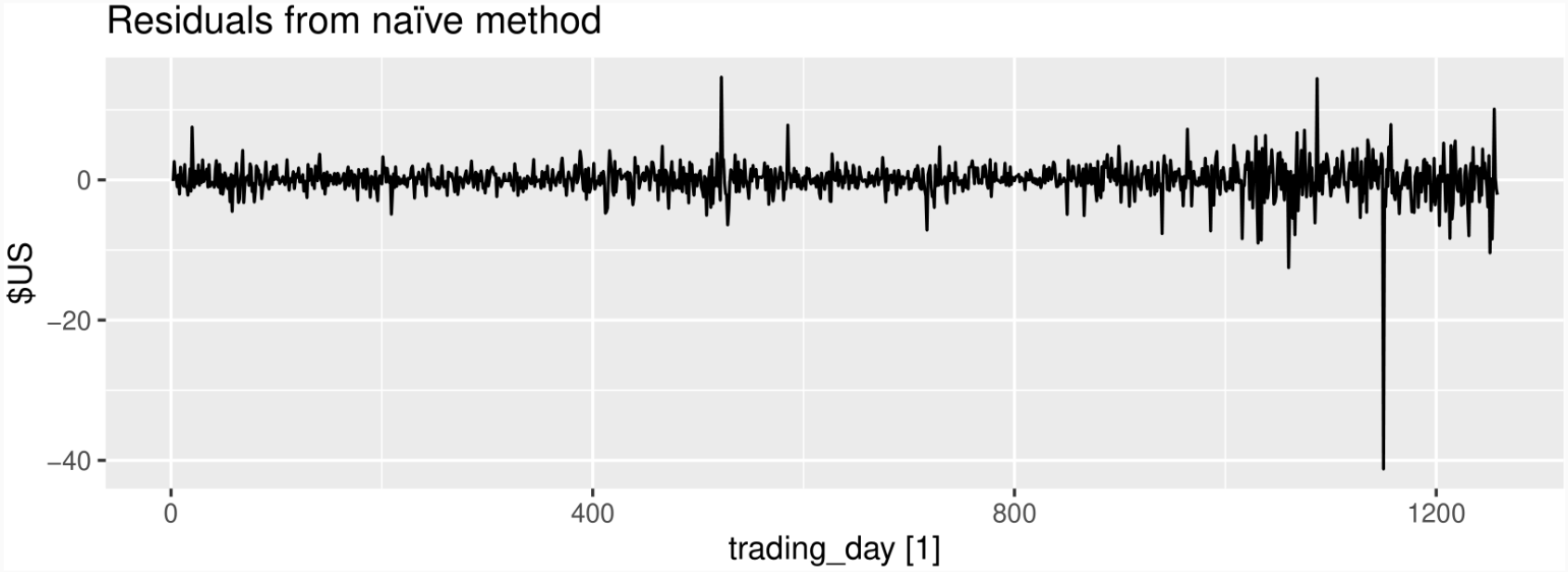
```
augment(fit) %>%
  filter(trading_day > 1100) %>%
  ggplot(aes(x = trading_day)) +
  geom_line(aes(y = Close, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

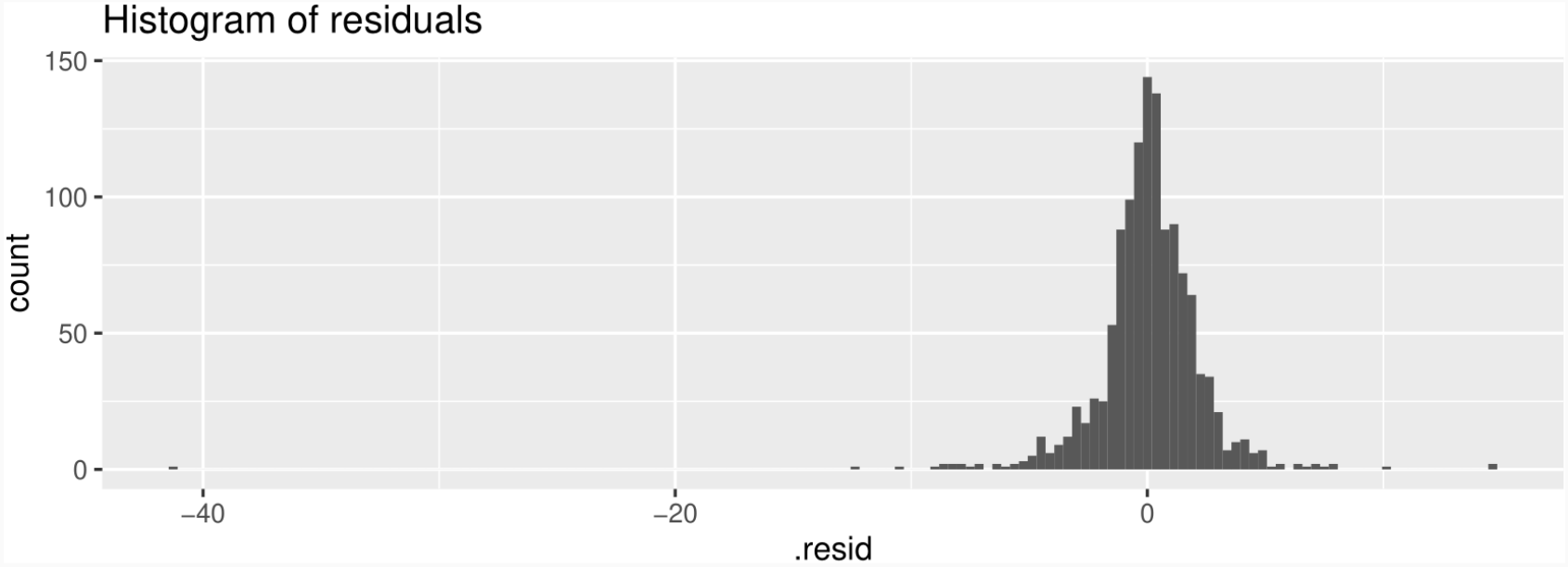
```
augment(fit) %>%
  autoplot(.resid) +
  labs(y = "$US",
       title = "Residuals from naïve method")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

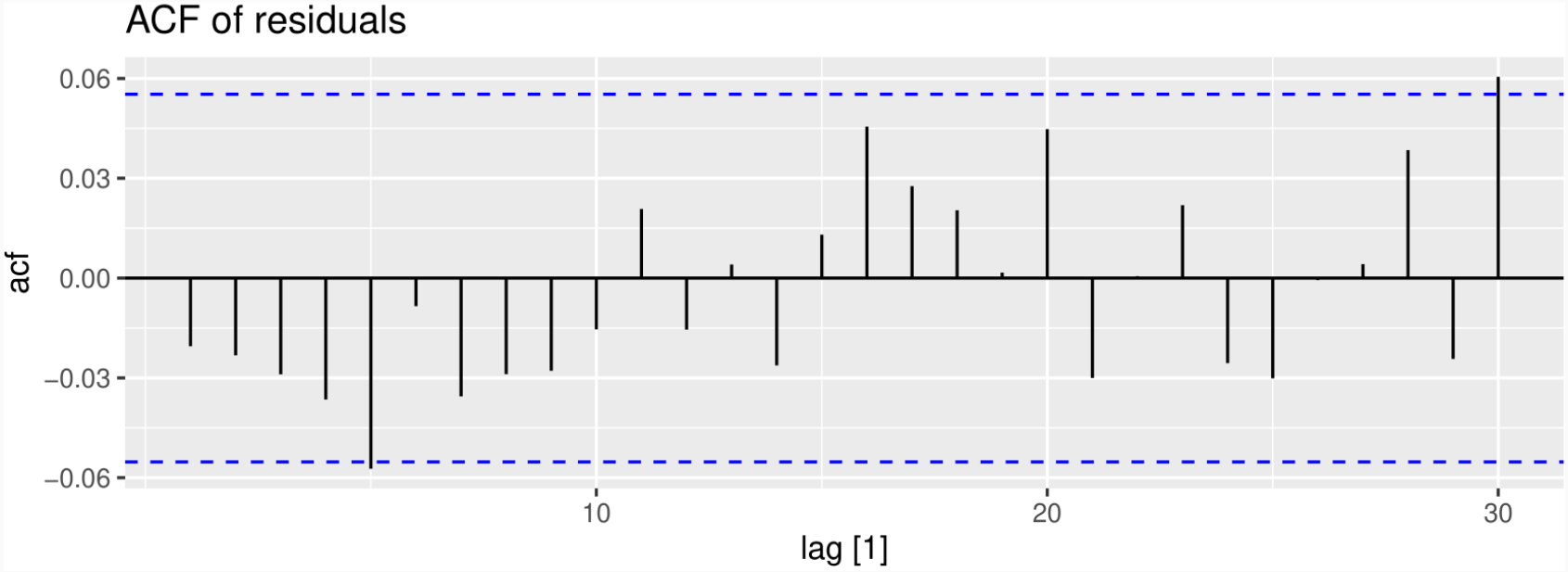
```
augment(fit) %>%
  ggplot(aes(x = .resid)) +
  geom_histogram(bins = 150) +
  labs(title = "Histogram of residuals")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Facebook closing stock price

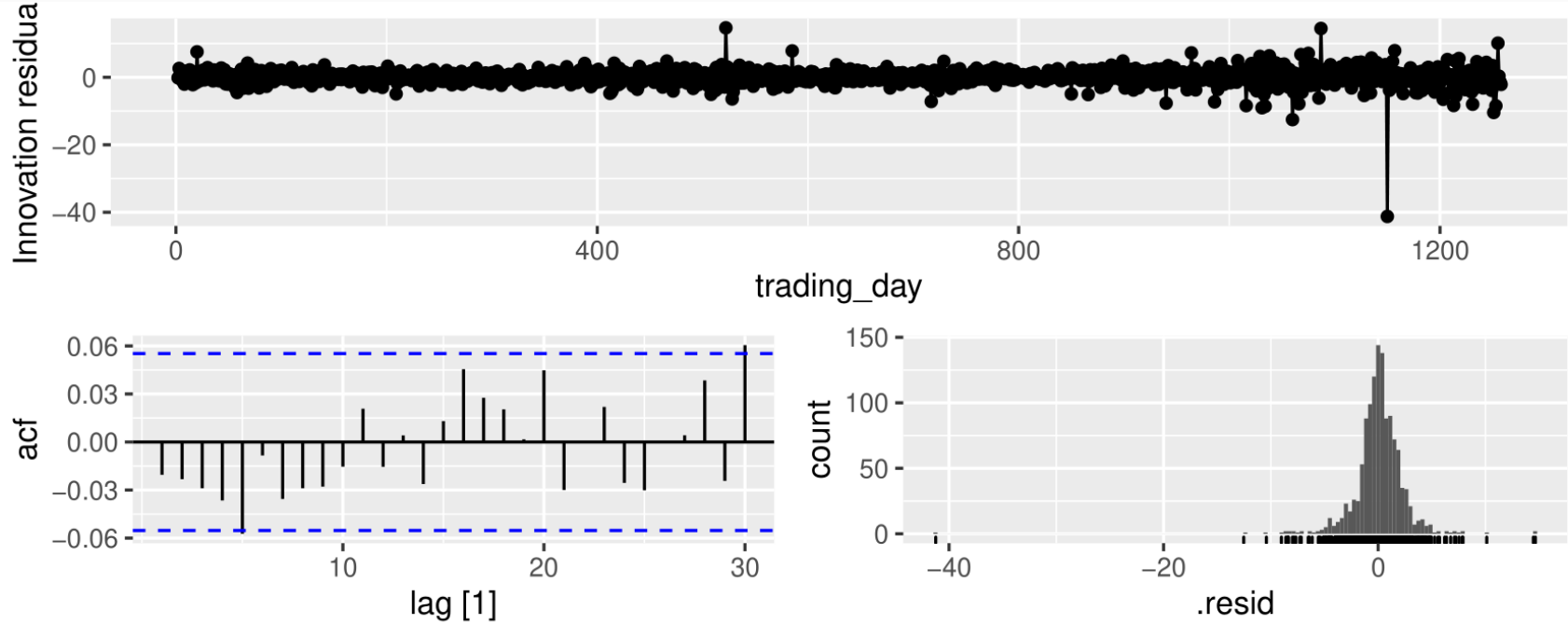
```
augment(fit) %>%
  ACF(.resid) %>%
  autoplot() + labs(title = "ACF of residuals")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

gg_tsresiduals() function

```
gg_tsresiduals(fit)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

ACF of residuals

- We assume that the residuals are white noise (uncorrelated, mean zero, constant variance). If they aren't, then there is information left in the residuals that should be used in computing forecasts.
- So a standard residual diagnostic is to check the ACF of the residuals of a forecasting method.
- We *expect* these to look like white noise.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Box-Pierce test

$$Q = T\sum_{k=1}^{\ell} r_k^2$$

where ℓ is max lag being considered and T is number of observations.

- If each r_k close to zero, Q will be **small**.
- If some r_k values large (positive or negative), Q will be **large**.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Ljung-Box test

$$Q^* = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2$$

where ℓ is max lag being considered and T is number of observations.

- My preferences: $\ell = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- Better performance, especially in small samples.

Portmanteau tests

- If data are WN, Q^* has χ^2 distribution with $(\ell - K)$ degrees of freedom where K = no. parameters in model.
- When applied to raw data, set $K = 0$.
- `lag` = ℓ, `dof` = K

```
augment(fit) %>%
  features(.resid, ljung_box, lag=10, dof=0)
```

```
## # A tibble: 1 x 4
##   Symbol .model         lb_stat lb_pvalue
##   <chr>  <chr>            <dbl>     <dbl>
## 1 FB     NAIVE(Close)      12.1     0.276
```

Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. **Distributional forecasts and prediction intervals**
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Forecast distributions

- A forecast $\hat{y}_{T+h|T}$ is (usually) the mean of the conditional distribution $y_{T+h} \mid y_1, \ldots, y_T$.
- Most time series models produce normally distributed forecasts.
- The forecast distribution describes the probability of observing any future value.

Forecast distributions

Assuming residuals are normal, uncorrelated, sd = $\hat{\sigma}$:

Mean:	$\hat{y}_{T+h\|T} \sim N(\bar{y}, (1+1/T)\hat{\sigma}^2)$
Naïve:	$\hat{y}_{T+h\|T} \sim N(y_T, h\hat{\sigma}^2)$
Seasonal naïve:	$\hat{y}_{T+h\|T} \sim N(y_{T+h-m(k+1)}, (k+1)\hat{\sigma}^2)$
Drift:	$\hat{y}_{T+h\|T} \sim N(y_T + \frac{h}{T-1}(y_T - y_1), h\frac{T+h}{T}\hat{\sigma}^2)$

where k is the integer part of $(h-1)/m$.

Note that when $h = 1$ and T is large, these all give the same approximate forecast variance: $\hat{\sigma}^2$.

Prediction intervals

- A prediction interval gives a region within which we expect y_{T+h} to lie with a specified probability.
- Assuming forecast errors are normally distributed, then a 95% PI is

$$\hat{y}_{T+h|T} \pm 1.96\hat{\sigma}_h$$

where $\hat{\sigma}_h$ is the st dev of the h-step distribution.

- When $h = 1$, $\hat{\sigma}_h$ can be estimated from the residuals.

Prediction intervals

```
brick_fc %>% hilo(level = 95)
```

```
## # A tsibble: 80 x 5 [1Q]
## # Key:       .model [4]
##    .model          Quarter       Bricks .mean         `95%`
##    <chr>             <qtr>       <dist> <dbl>        <hilo>
##  1 Seasonal_naive 2005 Q3 N(428, 2336)   428 [333, 523]95
##  2 Seasonal_naive 2005 Q4 N(397, 2336)   397 [302, 492]95
##  3 Seasonal_naive 2006 Q1 N(355, 2336)   355 [260, 450]95
##  4 Seasonal_naive 2006 Q2 N(435, 2336)   435 [340, 530]95
##  5 Seasonal_naive 2006 Q3 N(428, 4672)   428 [294, 562]95
##  6 Seasonal_naive 2006 Q4 N(397, 4672)   397 [263, 531]95
##  7 Seasonal_naive 2007 Q1 N(355, 4672)   355 [221, 489]95
##  8 Seasonal_naive 2007 Q2 N(435, 4672)   435 [301, 569]95
##  9 Seasonal_naive 2007 Q3 N(428, 7008)   428 [264, 592]95
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Prediction intervals

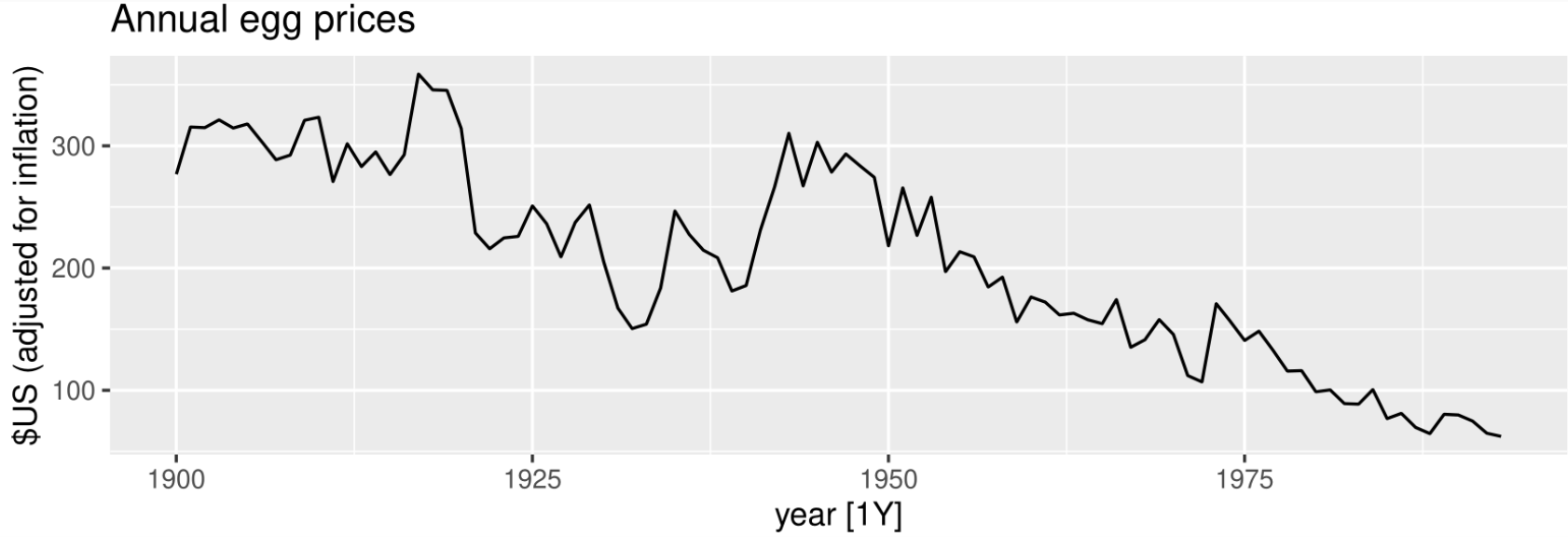
- Point forecasts often useless without a measure of uncertainty (such as prediction intervals).
- Prediction intervals require a stochastic model (with random errors, etc).
- For most models, prediction intervals get wider as the forecast horizon increases.
- Use `level` argument to control coverage.
- Check residual assumptions before believing them.
- Usually too narrow due to unaccounted uncertainty.

Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. Distributional forecasts and prediction intervals
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Modelling with transformations

```
eggs <- prices %>%
  filter(!is.na(eggs)) %>% select(eggs)
eggs %>% autoplot() +
  labs(title="Annual egg prices",
       y="$US (adjusted for inflation)")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Modelling with transformations

Transformations used in the left of the formula will be automatically back-transformed. To model log-transformed egg prices, you could use:

```
fit <- eggs %>%
  model(RW(log(eggs) ~ drift()))
fit
```

```
## # A mable: 1 x 1
##   `RW(log(eggs) ~ drift())`
##                      <model>
## 1              <RW w/ drift>
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

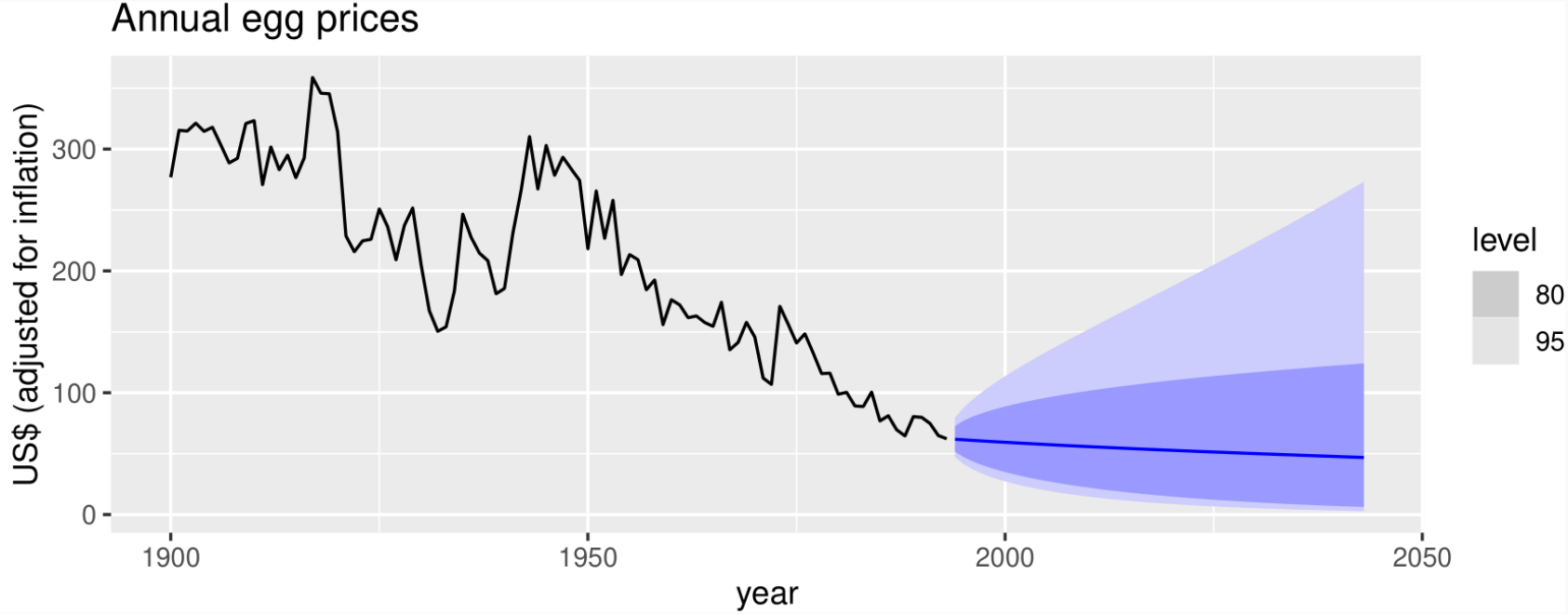
Forecasting with transformations

```
fc <- fit %>%
  forecast(h = 50)
fc
```

```
## # A fable: 50 x 4 [1Y]
## # Key:     .model [1]
##    .model                    year             eggs .mean
##    <chr>                    <dbl>           <dist> <dbl>
##  1 RW(log(eggs) ~ drift())  1994 t(N(4.1, 0.018))  61.8
##  2 RW(log(eggs) ~ drift())  1995 t(N(4.1, 0.036))  61.4
##  3 RW(log(eggs) ~ drift())  1996 t(N(4.1, 0.054))  61.0
##  4 RW(log(eggs) ~ drift())  1997 t(N(4.1, 0.073))  60.5
##  5 RW(log(eggs) ~ drift())  1998 t(N(4.1, 0.093))  60.1
##  6 RW(log(eggs) ~ drift())  1999    t(N(4, 0.11))  59.7
##  7 RW(log(eggs) ~ drift())  2000    t(N(4, 0.13))  59.3
##  8 RW(log(eggs) ~ drift())  2001    t(N(4, 0.15))  58.9
```

Forecasting with transformations

```
fc %>% autoplot(eggs) +
  labs(title="Annual egg prices",
       y="US$ (adjusted for inflation)")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Bias adjustment

- Back-transformed point forecasts are medians.
- Back-transformed PI have the correct coverage.

Back-transformed means

Let X be have mean μ and variance σ^2.

Let $f(x)$ be back-transformation function, and $Y = f(X)$.

Taylor series expansion about μ:

$$f(X) = f(\mu) + (X - \mu)f'(\mu) + \frac{1}{2}(X - \mu)^2 f''(\mu).$$

$$\mathrm{E}[Y] = \mathrm{E}[f(X)] = f(\mu) + \tfrac{1}{2}\sigma^2 f''(\mu)$$

Bias adjustment

Box-Cox back-transformation:

$$y_t = \begin{cases} \exp(w_t) & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

$$f(x) = \begin{cases} e^x & \lambda = 0; \\ (\lambda x + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

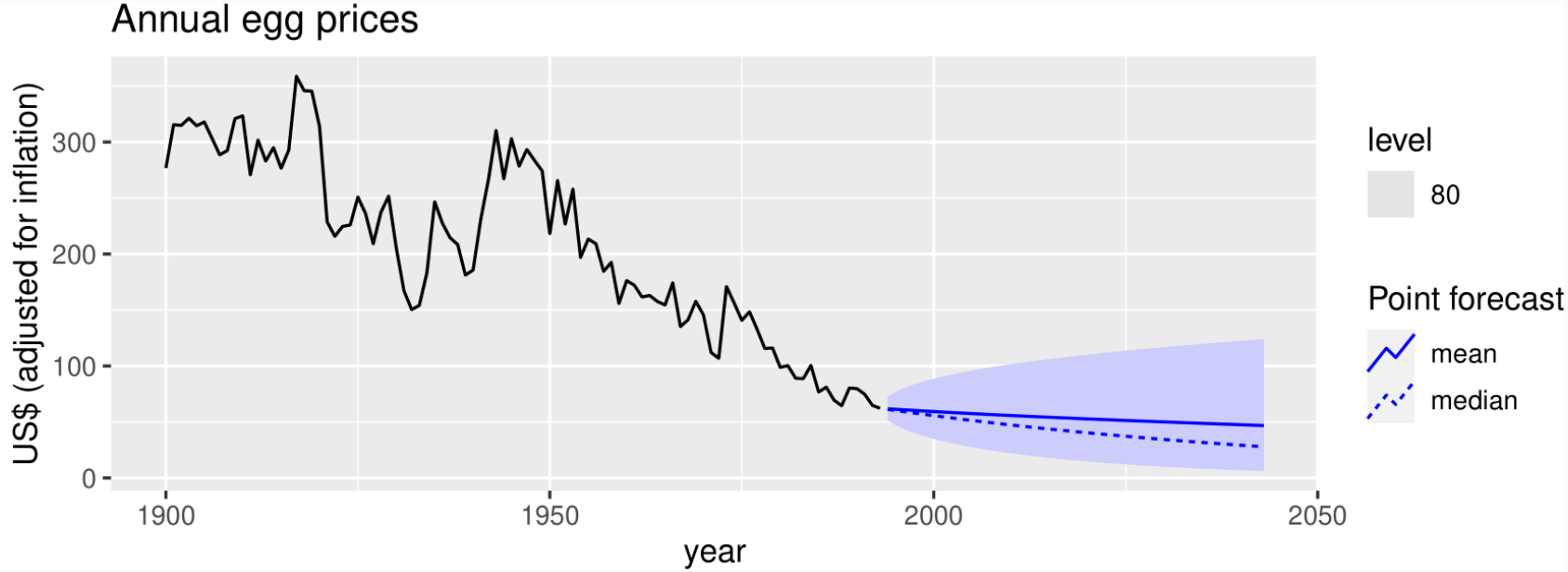
$$f''(x) = \begin{cases} e^x & \lambda = 0; \\ (1-\lambda)(\lambda x + 1)^{1/\lambda - 2} & \lambda \neq 0. \end{cases}$$

$$\mathrm{E}[Y] = \begin{cases} e^{\mu}\left[1 + \frac{\sigma^2}{2}\right] & \lambda = 0; \\ (\lambda\mu + 1)^{1/\lambda}\left[1 + \frac{\sigma^2(1-\lambda)}{2(\lambda\mu+1)^2}\right] & \lambda \neq 0. \end{cases}$$

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Bias adjustment

```
fc %>%
  autoplot(eggs, level = 80, point_forecast = lst(mean, median)) +
  labs(title="Annual egg prices",
       y="US$ (adjusted for inflation)")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. Distributional forecasts and prediction intervals
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Forecasting and decomposition

$$y_t = \hat{S}_t + \hat{A}_t$$

- $\hat{A}_t$ is seasonally adjusted component
- $\hat{S}_t$ is seasonal component.

- Forecast $\hat{S}_t$ using SNAIVE.
- Forecast $\hat{A}_t$ using non-seasonal time series method.
- Combine forecasts of $\hat{S}_t$ and $\hat{A}_t$ to get forecasts of original data.

US Retail Employment

```
us_retail_employment <- us_employment %>%
  filter(year(Month) >= 1990, Title == "Retail Trade") %>%
  select(-Series_ID)
us_retail_employment
```

```
## # A tsibble: 357 x 3 [1M]
##       Month Title        Employed
##       <mth> <chr>           <dbl>
##  1 1990 Jan Retail Trade   13256.
##  2 1990 Feb Retail Trade   12966.
##  3 1990 Mar Retail Trade   12938.
##  4 1990 Apr Retail Trade   13012.
##  5 1990 May Retail Trade   13108.
##  6 1990 Jun Retail Trade   13183.
##  7 1990 Jul Retail Trade   13170.
##  8 1990 Aug Retail Trade   13160.
##  9 1990 Sep Retail Trade   13113
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

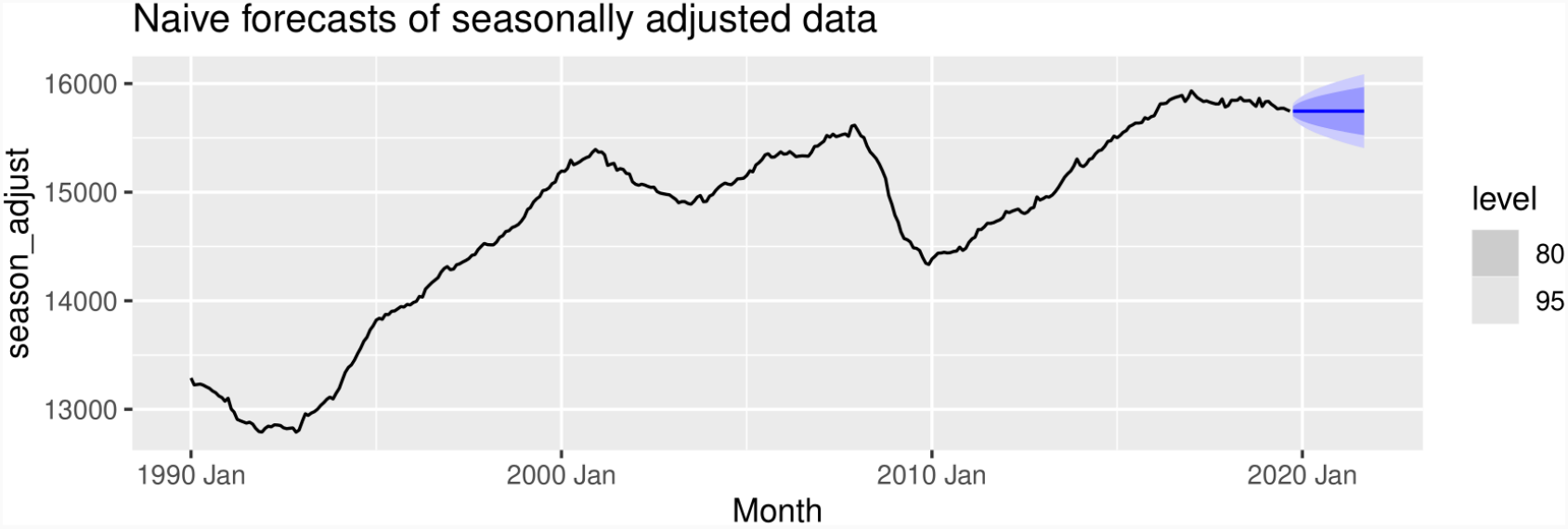
```
dcmp <- us_retail_employment %>%
  model(STL(Employed)) %>%
  components() %>% select(-.model)
dcmp
```

```
## # A tsibble: 357 x 6 [1M]
##       Month Employed  trend season_year remainder season_adjust
##       <mth>    <dbl>  <dbl>       <dbl>     <dbl>          <dbl>
##  1 1990 Jan   13256. 13288.      -33.0      0.836         13289.
##  2 1990 Feb   12966. 13269.     -258.     -44.6           13224.
##  3 1990 Mar   12938. 13250.     -290.     -22.1           13228.
##  4 1990 Apr   13012. 13231.     -220.       1.05          13232.
##  5 1990 May   13108. 13211.     -114.      11.3           13223.
##  6 1990 Jun   13183. 13192.      -24.3     15.5           13207.
##  7 1990 Jul   13170. 13172.      -23.2     21.6           13193.
##  8 1990 Aug   13160. 13151.       -9.52    17.8           13169.
##  9 1990 Sep   13113. 13131.      -39.5     22.0           13153.
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

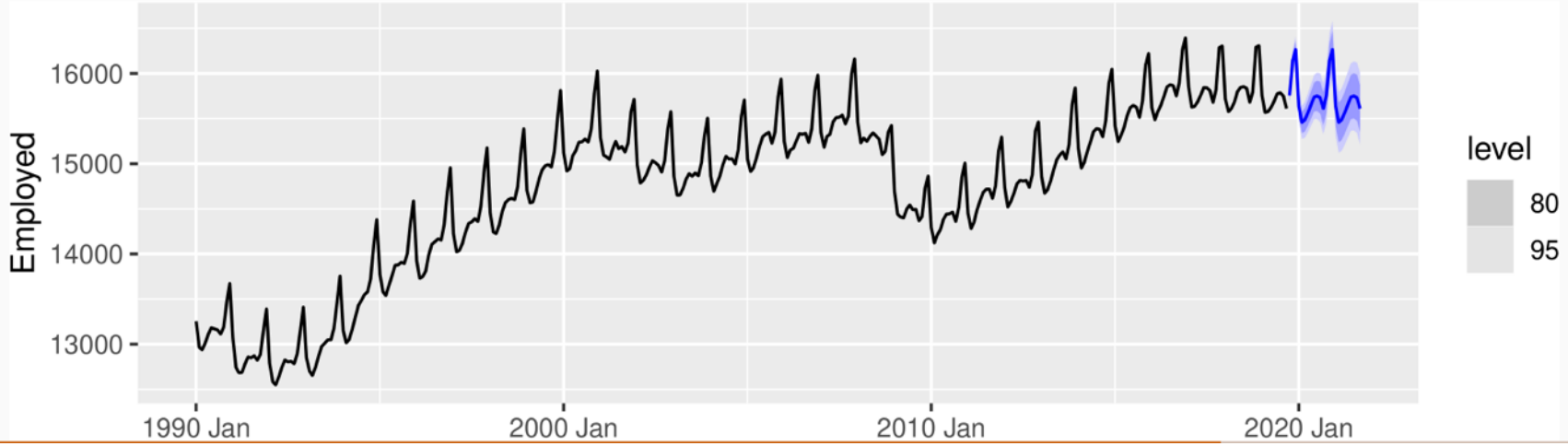
```
dcmp %>%
  model(NAIVE(season_adjust)) %>%
  forecast() %>%
  autoplot(dcmp) +
  labs(title = "Naive forecasts of seasonally adjusted data")
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

US Retail Employment

```
us_retail_employment %>%
  model(stlf = decomposition_model(
    STL(Employed ~ trend(window = 7), robust = TRUE),
    NAIVE(season_adjust)
  )) %>%
  forecast() %>%
  autoplot(us_retail_employment)
```



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Decomposition models

`decomposition_model()` creates a decomposition model

- You must provide a method for forecasting the `season_adjust` series.
- A seasonal naive method is used by default for the `seasonal` components.
- The variances from both the seasonally adjusted and seasonal forecasts are combined.

Outline

1. A tidy forecasting workflow
2. Some simple forecasting methods
3. Residual diagnostics
4. Distributional forecasts and prediction intervals
5. Forecasting with transformations
6. Forecasting and decomposition
7. Evaluating forecast accuracy

Training and test sets

- A model which fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.
- The test set must not be used for *any* aspect of model development or calculation of forecasts.
- Forecast accuracy is based only on the test set.

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Forecast errors

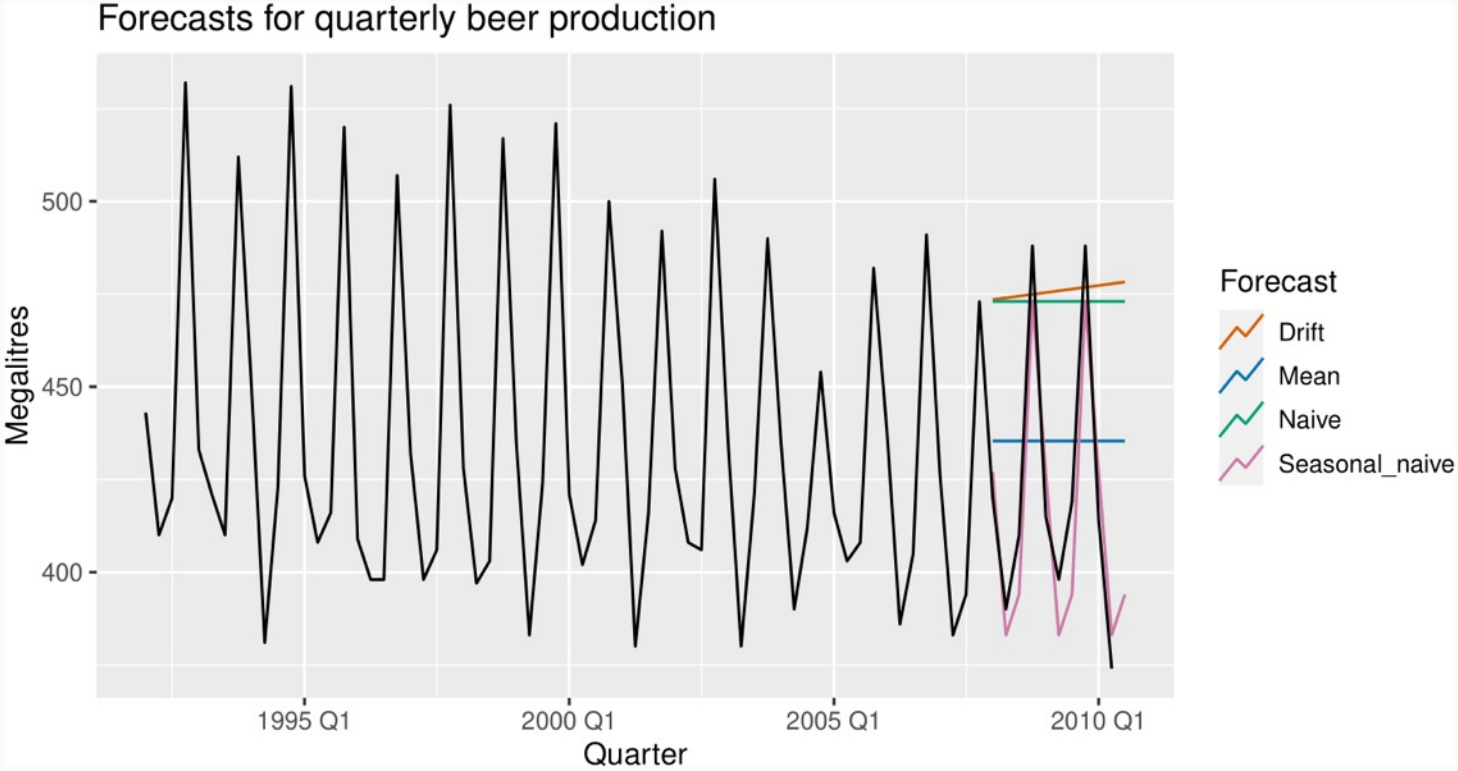
Forecast "error": the difference between an observed value and its forecast.

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

where the training data is given by $\{y_1, \ldots, y_T\}$

- Unlike residuals, forecast errors on the test set involve multi-step forecasts.
- These are *true* forecast errors as the test data is not used in computing $\hat{y}_{T+h|T}$.

Measures of forecast accuracy



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Measures of forecast accuracy

y_{T+h} = $(T+h)$th observation, $h = 1, \ldots, H$

$\hat{y}_{T+h|T}$ = its forecast based on data up to time T.

e_{T+h} = $y_{T+h} - \hat{y}_{T+h|T}$

$$\text{MAE} = \text{mean}(|e_{T+h}|)$$

$$\text{MSE} = \text{mean}(e_{T+h}^2) \qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)}$$

$$\text{MAPE} = 100\text{mean}(|e_{T+h}|/|y_{T+h}|)$$

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Measures of forecast accuracy

y_{T+h} = $(T+h)$th observation, $h = 1, \ldots, H$

$\hat{y}_{T+h|T}$ = its forecast based on data up to time T.

e_{T+h} = $y_{T+h} - \hat{y}_{T+h|T}$

$$\text{MAE} = \text{mean}(|e_{T+h}|)$$

$$\text{MSE} = \text{mean}(e_{T+h}^2) \qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)}$$

$$\text{MAPE} = 100\text{mean}(|e_{T+h}|/|y_{T+h}|)$$

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all t, and y has a natural zero.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006).

For non-seasonal time series,

$$Q = (T-1)^{-1} \sum_{t=2}^{T} |y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naïve method.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

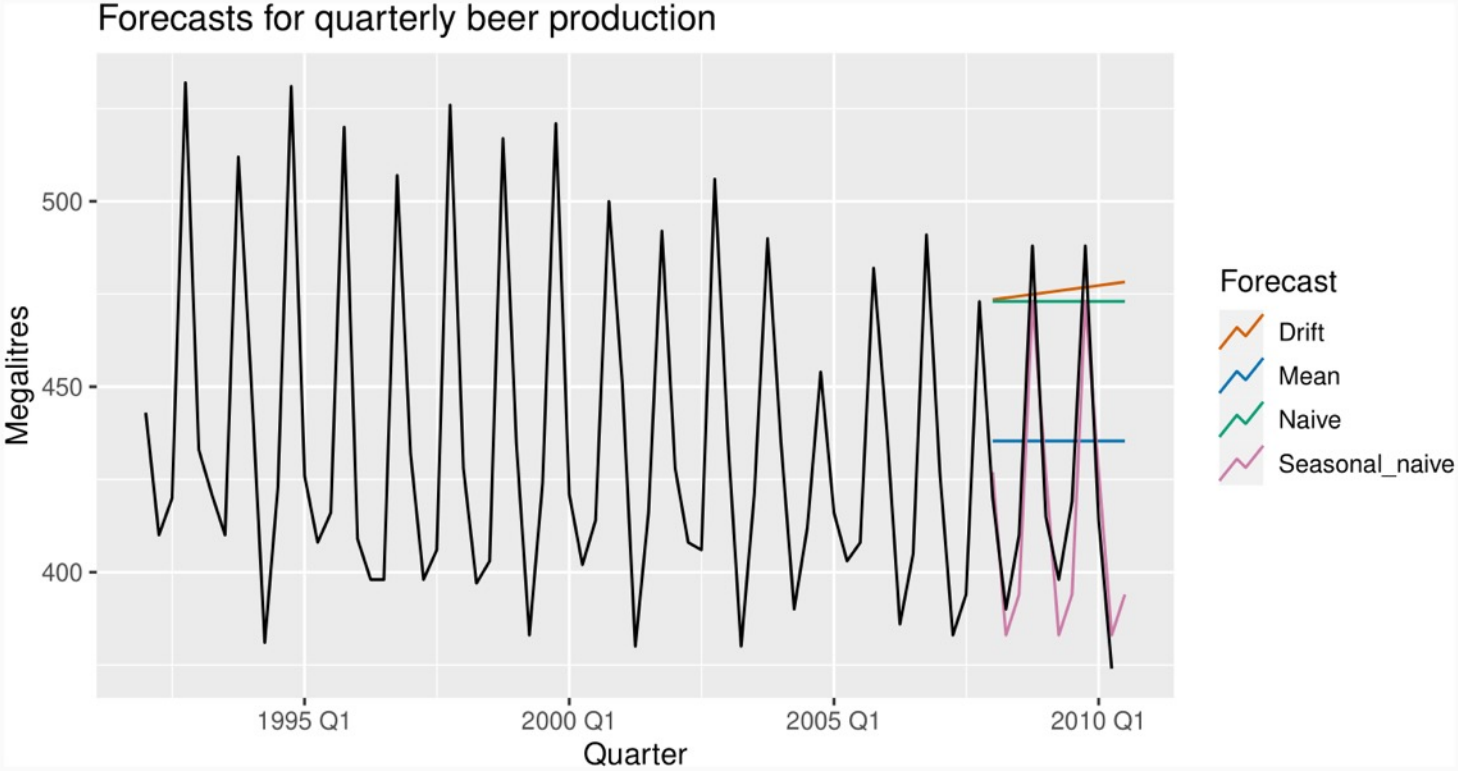
Proposed by Hyndman and Koehler (IJF, 2006).

For seasonal time series,

$$Q = (T-m)^{-1} \sum_{t=m+1}^{T} |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naïve method.

Measures of forecast accuracy



Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Measures of forecast accuracy

```
recent_production <- aus_production %>%
  filter(year(Quarter) >= 1992)
train <- recent_production %>%
  filter(year(Quarter) <= 2007)
beer_fit <- train %>%
  model(
    Mean = MEAN(Beer),
    Naive = NAIVE(Beer),
    Seasonal_naive = SNAIVE(Beer),
    Drift = RW(Beer ~ drift())
  )
beer_fc <- beer_fit %>%
  forecast(h = 10)
```

Measures of forecast accuracy

```
accuracy(beer_fit)
```

```
## # A tibble: 4 x 6
##   .model         .type     RMSE   MAE  MAPE  MASE
##   <chr>          <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 Drift          Training  65.3  54.8 12.2   3.83
## 2 Mean           Training  43.6  35.2  7.89  2.46
## 3 Naive          Training  65.3  54.7 12.2   3.83
## 4 Seasonal_naive Training  16.8  14.3  3.31  1
```

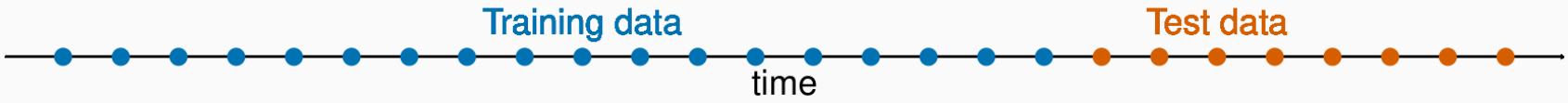
```
accuracy(beer_fc, recent_production)
```

```
## # A tibble: 4 x 6
##   .model         .type  RMSE   MAE  MAPE  MASE
##   <chr>          <chr> <dbl> <dbl> <dbl> <dbl>
## 1 Drift          Test   64.9  58.9 14.6  4.12
## 2 Mean           Test   38.4  34.8  8.28 2.44
## 3 Naive          Test   62.7  57.4 14.2  4.01
## 4 Seasonal_naive Test   14.3  13.4  3.17 0.937
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

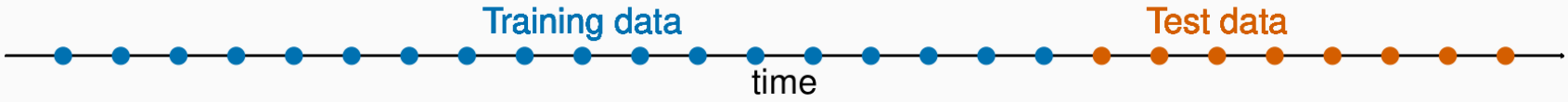
Traditional evaluation



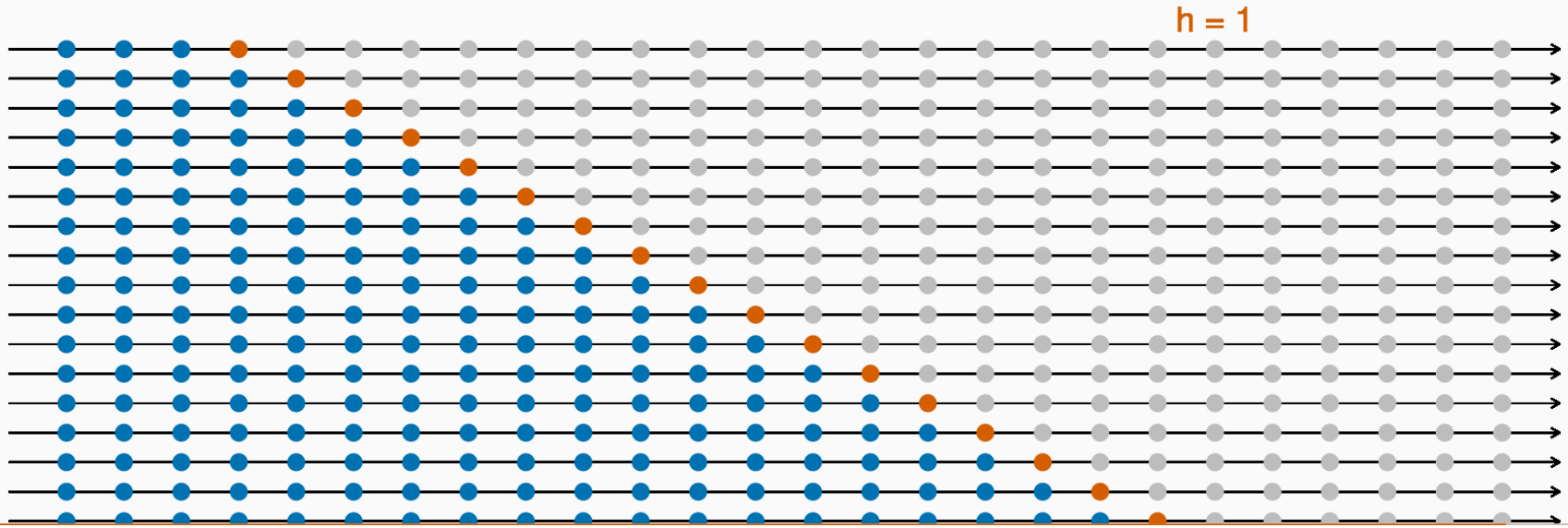
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Traditional evaluation



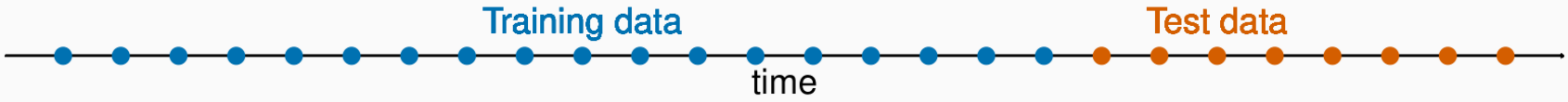
Time series cross-validation



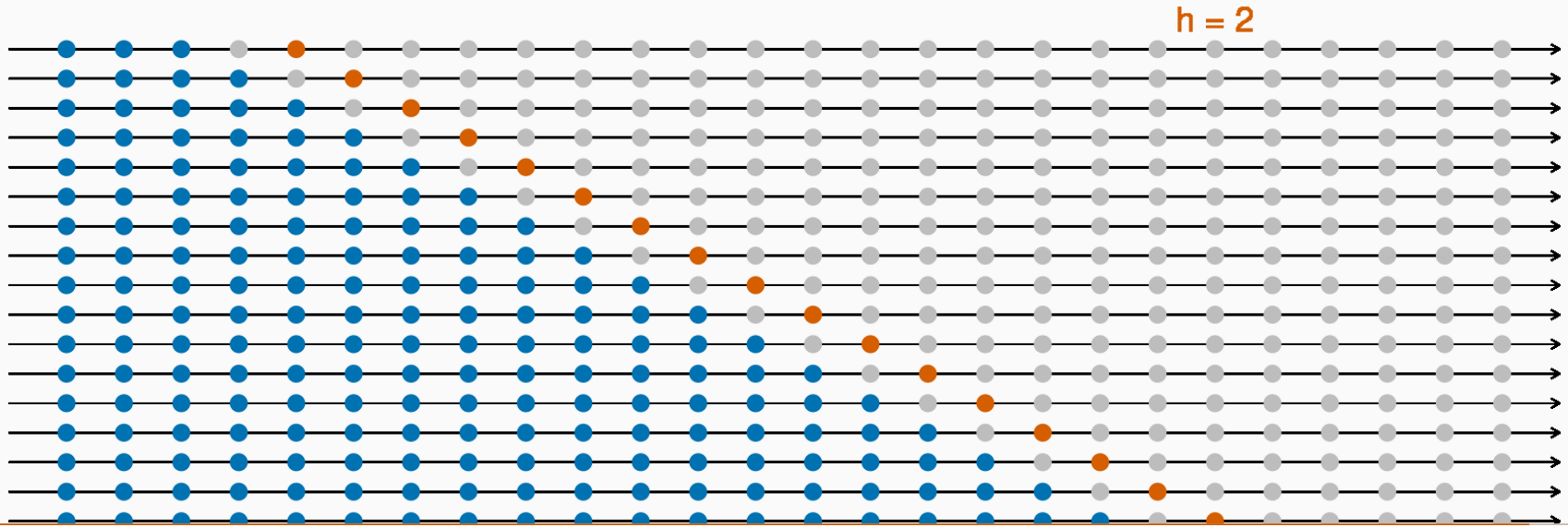
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Traditional evaluation



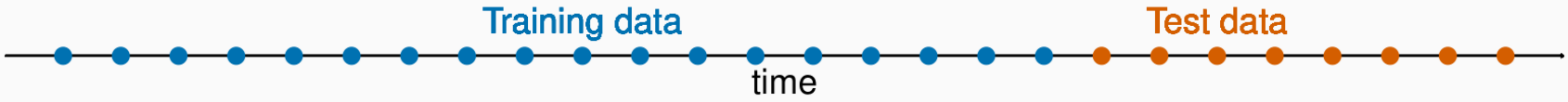
Time series cross-validation



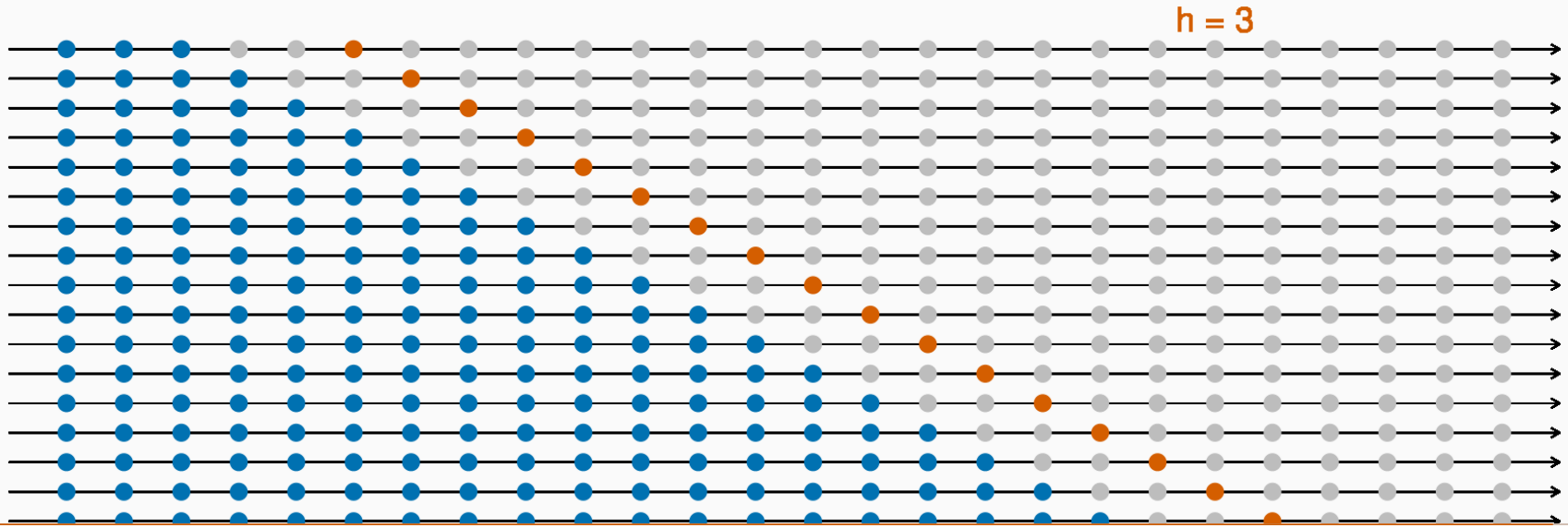
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Traditional evaluation



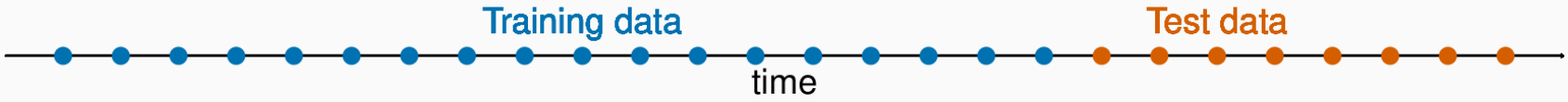
Time series cross-validation



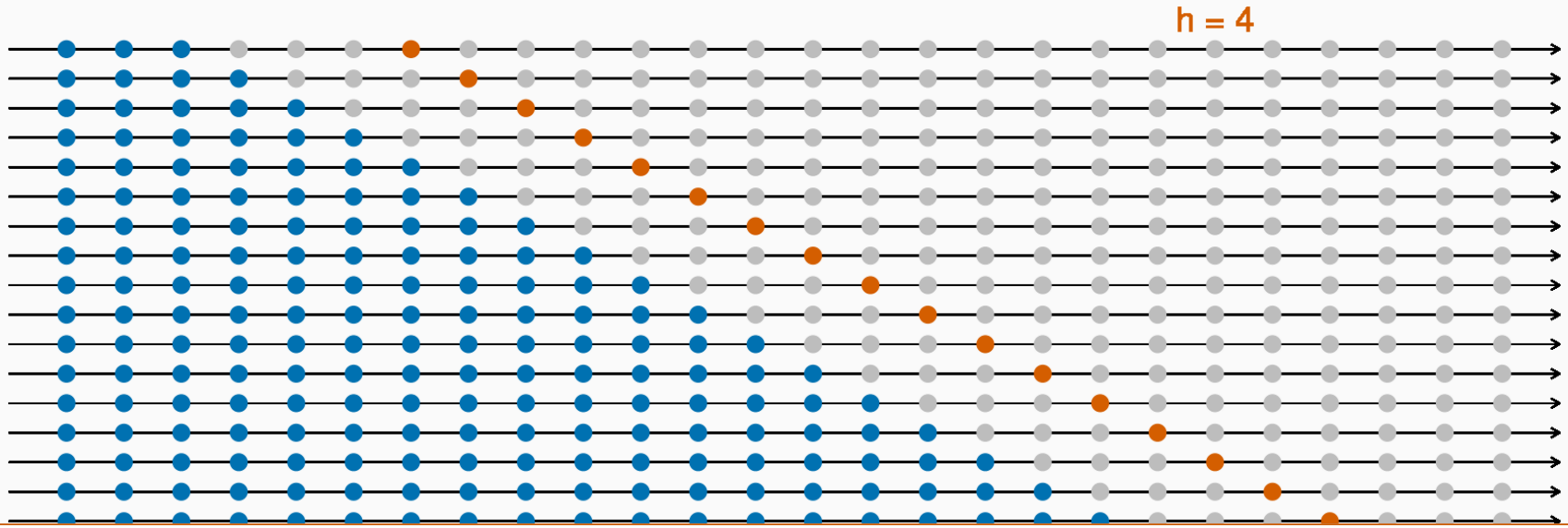
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Traditional evaluation



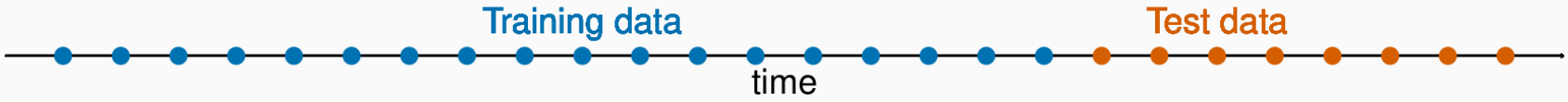
Time series cross-validation



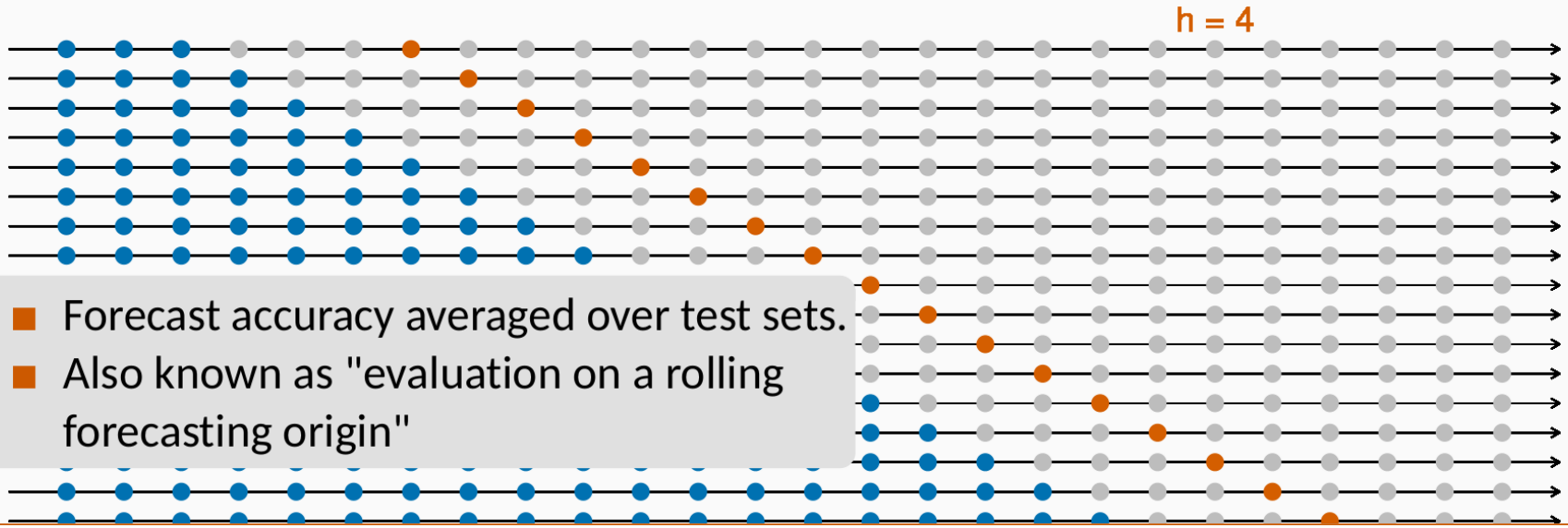
Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Traditional evaluation



Time series cross-validation



- Forecast accuracy averaged over test sets.
- Also known as "evaluation on a rolling forecasting origin"

Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Stretch with a minimum length of 3, growing by 1 each step.

```
fb_stretch <- fb_stock %>%
  stretch_tsibble(.init = 3, .step = 1) %>%
  filter(.id != max(.id))
```

```
## # A tsibble: 790,650 x 4 [1]
## # Key:       .id [1,255]
##   Date       Close trading_day   .id
##   <date>     <dbl>       <int> <int>
## 1 2014-01-02  54.7           1     1
## 2 2014-01-03  54.6           2     1
## 3 2014-01-06  57.2           3     1
## 4 2014-01-02  54.7           1     2
## 5 2014-01-03  54.6           2     2
## 6 2014-01-06  57.2           3     2
## 7 2014-01-07  57.9           4     2
```

Time series cross-validation

Estimate RW w/ drift models for each window.

```
fit_cv <- fb_stretch %>%
  model(RW(Close ~ drift()))
```

```
## # A mable: 1,255 x 3
## # Key:     .id, Symbol [1,255]
##     .id Symbol `RW(Close ~ drift())`
##   <int> <chr>                 <model>
## 1     1 FB             <RW w/ drift>
## 2     2 FB             <RW w/ drift>
## 3     3 FB             <RW w/ drift>
## 4     4 FB             <RW w/ drift>
## # ... with 1,251 more rows
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

Produce one step ahead forecasts from all models.

```
fc_cv <- fit_cv %>%
  forecast(h=1)
```

```
## # A fable: 1,255 x 5 [1]
## # Key:     .id, Symbol [1,255]
##     .id Symbol trading_day      Close .mean
##   <int> <chr>        <dbl>     <dist> <dbl>
## 1     1 FB               4 N(58, 3.9)  58.4
## 2     2 FB               5   N(59, 2)  59.0
## 3     3 FB               6 N(59, 1.5)  59.1
## 4     4 FB               7 N(58, 1.8)  57.7
## # ... with 1,251 more rows
```


Source: Rob J Hyndman, https://github.com/robjhyndman/ETC3550Slides

Time series cross-validation

```
# Cross-validated
fc_cv %>% accuracy(fb_stock)
# Training set
fb_stock %>% model(RW(Close ~ drift())) %>% accuracy()
```

	RMSE	MAE	MAPE
Cross-validation	2.418	1.469	1.266
Training	2.414	1.465	1.261

A good way to choose the best forecasting model is to find the model with the smallest RMSE computed using time series cross-validation.