workshop.nectric.com.au/tidyfc2024

Time Series Analysis & Forecasting Using R

10. Forecast reconciliation

tsibble feasts fable

Instructors

Mitchell O'Hara-Wild

mitchelloharawild.com

@mitchelloharawild

Mitch.OHara-Wild@monash.edu

Assumptions

- This is not an introduction to R. We assume you are broadly comfortable with R code, the RStudio environment and the tidyverse.
- This is not a statistics course. We assume you are familiar with concepts such as the mean, standard deviation, quantiles, regression, normal distribution, likelihood, etc.
- This is not a theory course. We are not going to derive anything. We will teach you time series and forecasting tools, when to use them, and how to use them most effectively.

Key reference

Hyndman, R. J. & Athanasopoulos, G. (2021)
***Forecasting: principles and practice*, 3rd ed.**

Key reference

Hyndman, R. J. & Athanasopoulos, G. (2021)
***Forecasting: principles and practice*, 3rd ed.**

OTexts.org/fpp3/

Key reference

Hyndman, R. J. & Athanasopoulos, G. (2021)
***Forecasting: principles and practice*, 3rd ed.**

OTexts.org/fpp3/

- Free and online
- Data sets in associated R package
- R code for examples

Poll: How experienced are you in forecasting

1. Guru: I wrote the software, teach forecasting, and do the conference circuit.
2. Expert: It has been my full time job for more than a decade.
3. Skilled: I have been doing it for years.
4. Comfortable: I understand it and have done it.
5. Learner: I am still learning.
6. Beginner: I have heard of it and would like to learn more.
7. Unknown: What is forecasting? Is that what the weather people do?

Poll: How proficient are you in using R?

1. Guru: The R core team come to me for advice.
2. Expert: I have written several packages on CRAN.
3. Skilled: I use it regularly and it is an important part of my job.
4. Comfortable: I use it often and am comfortable with the tool.
5. User: I use it sometimes, but I am often searching around for the right function.
6. Learner: I have used it a few times.
7. Beginner: I’ve managed to download and install it.
8. Unknown: Why are you speaking like a pirate?

Install required packages

```
install.packages(c(
  "tidyverse",
  "fpp3",
  "readabs",
  "GGally",
  "sugrrants"
))
```

Approximate outline

Day	Topic	Chapter
1	1. Introduction to tsibbles	2
1	2. Time series graphics	2
1	3. Transformations	3
1	4. Decomposing patterns	3
1	5. Introduction to forecasting	5,7
2	6. Exponential smoothing	8
2	7. ARIMA models	9,10
2	8. Accuracy evaluation	5
2	9. Multivariate modelling	12
2	10. Forecast reconciliation	11

workshop.nectric.com.au/tidyfc2024

Outline

Outline

Tidyverts packages

tidyverts.org

tsibble

tsibbledata

feasts

fable

Tidyverts developers

Earo Wang

Mitchell O'Hara-Wild

Time series data

- Four-yearly Olympic winning times
- Annual Google profits
- Quarterly Australian beer production
- Monthly rainfall
- Weekly retail sales
- Daily IBM stock prices
- Hourly electricity demand
- 5-minute freeway traffic counts
- Time-stamped stock transaction data

Class packages

```r
# Data manipulation
library(dplyr)
# Plotting functions
library(ggplot2)
# Time and date manipulation
library(lubridate)
# Time series class
library(tsibble)
# Tidy time series data
library(tsibbledata)
# Time series graphics and statistics
library(feasts)
# Forecasting functions
library(fable)
```

Class packages

```r
# Data manipulation
library(dplyr)
# Plotting functions
library(ggplot2)
# Time and date manipulation
library(lubridate)
# Time series class
library(tsibble)
# Tidy time series data
library(tsibbledata)
# Time series graphics and statistics
library(feasts)
# Forecasting functions
library(fable)
```

```r
# All of the above
library(fpp3)
```

`tsibble` objects

```
{r, echo = TRUE} global_economy
```

`tsibble` objects

```
{r, echo = TRUE} global_economy
```

Index

`tsibble` objects

{r, echo = TRUE} global_economy

Index Key

`tsibble` objects

{r, echo = TRUE} global_economy

Index Key Measured variables

`tsibble` objects

```
{r, echo = TRUE} tourism
```

Domestic visitor nights in thousands by state/region and purpose.

`tsibble` objects

```
{r, echo = TRUE} tourism
```

Index

Domestic visitor nights in thousands by state/region and purpose.

`tsibble` objects

```
{r, echo = TRUE} tourism
```

Index

Keys

Domestic visitor nights in thousands by state/region and purpose.

`tsibble` objects

```
{r, echo = TRUE} tourism
```

Index

Keys

Measure

Domestic visitor nights in thousands by state/region and purpose.

`tsibble` objects

- A `tsibble` allows storage and manipulation of multiple time series in R.
- It contains:
  - An index: time information about the observation
  - Measured variable(s): numbers of interest
  - Key variable(s): optional unique identifiers for each series
- It works with tidyverse functions.

The `tsibble` index

Example

```
mydata <- tsibble(
  year = 2012:2016,
  y = c(123, 39, 78, 52, 110),
  index = year
)
mydata
```

The `tsibble` index

For observations more frequent than once per year, we need to use a time class function on the index.

```
{r tstablemonth, echo=FALSE} z <- tibble(Month = paste(2019,
month.abb[1:5]), Observation = c(50, 23, 34, 30, 25)) #
knitr::kable(z, booktabs=TRUE)
```

```
z
```

The `tsibble` index

For observations more frequent than once per year, we need to use a time class function on the index.

```
z |>
  mutate(Month = yearmonth(Month)) |>
  as_tsibble(index = Month)
```

The `tsibble` index

Common time index variables can be created with these functions:

```
{r tstable2, echo=FALSE} tribble(   ~`Frequency`,
~Function,   "Annual", "`start:end`",   "Quarterly",
"`yearquarter()`",   "Monthly", "`yearmonth()`",
"Weekly", "`yearweek()`",   "Daily", "`as_date()`,
`ymd()`",   "Sub-daily", "`as_datetime()`" ) |>
knitr::kable(booktabs = TRUE)
```

Outline

Australian prison population

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv")
```

`{r prison2a, dependson="prison", echo=FALSE} prison` ## Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") |>
  mutate(Quarter = yearquarter(date))
```

`{r prison3a, dependson="prison3", echo=FALSE} prison`

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") |>
  mutate(Quarter = yearquarter(date)) |>
  select(-date)
```

```
{r prison4a, dependson="prison4", echo=FALSE} prison
```

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") |>
  mutate(Quarter = yearquarter(date)) |>
  select(-date) |>
  as_tsibble(
    index = Quarter,
    key = c(state, gender, legal, indigenous)
  )
```

```
{r prison5a, dependson="prison5", echo=FALSE} prison
```

Outline

Australian Pharmaceutical Benefits Scheme

Australian Pharmaceutical Benefits Scheme

The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy scheme.

Australian Pharmaceutical Benefits Scheme

The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy scheme.

- Many drugs bought from pharmacies are subsidised to allow more equitable access to modern drugs.
- The cost to government is determined by the number and types of drugs purchased. Currently nearly 1% of GDP.
- The total cost is budgeted based on forecasts of drug usage.
- Costs are disaggregated by drug type (ATC1 `xr length(unique(PBS$ATC1))` / ATC2 `r length(unique(PBS$ATC2))`), concession category (`xr length(unique(PBS$Concession))`) and patient type (`xr length(unique(PBS$Type))`), giving $84 \times 2 \times 2 =$ '$r84 * 2 * 2$'

Working with `tsibble` objects

```
{r wide, include=FALSE} options(width = 78)
```

```
{r pbs1, dependson='wide'} PBS
```

Working with `tsibble` objects

We can use the `filter()` function to select rows.

```
PBS |>
  filter(ATC2 == "A10")
```

Working with `tsibble` objects

We can use the `select()` function to select columns.

```
PBS |>
  filter(ATC2 == "A10") |>
  select(Month, Concession, Type, Cost)
```

Working with `tsibble` objects

We can use the `summarise()` function to summarise over keys.

```
PBS |>
  filter(ATC2 == "A10") |>
  select(Month, Concession, Type, Cost) |>
  summarise(total_cost = sum(Cost))
```

Working with `tsibble` objects

We can use the `mutate()` function to create new variables.

```
PBS |>
  filter(ATC2 == "A10") |>
  select(Month, Concession, Type, Cost) |>
  summarise(total_cost = sum(Cost)) |>
  mutate(total_cost = total_cost / 1e6)
```

Working with `tsibble` objects

We can use the `mutate()` function to create new variables.

```
PBS |>
  filter(ATC2 == "A10") |>
  select(Month, Concession, Type, Cost) |>
  summarise(total_cost = sum(Cost)) |>
  mutate(total_cost = total_cost / 1e6) -> a10
```

```
{r a10, echo=FALSE, dependson="pbs6"} a10
```

```
{r narrow, include=FALSE} options(width = 60)
```

Outline

Lab Session 1a

1. Use the `readabs` package to download and tidy the time series data from table `"19"` of the `"6202.0"` catalogue.
2. Use `filter()` to remove time series relating to hours worked disaggregated by State/Territory. Also remove the aggregated time series (which specify `"Persons"`).
3. Split the series column into two columns, `workload` and `sex` using the `separate()` function.
4. Create a tsibble with the appropriate index and key variables, you will need to convert the date variable into a time variable with the appropriate granularity.

Lab Session 1b

Now let's look at this data:

5. Find which series contains the most worked hours.
6. Create a new tsibble which aggregates over `sex`, and just has monthly worked hours by type.
7. What is the typical difference in total working hours between full time and part time employees? (try to visualise this!)

Outline

Time plots

```
{r maxtemp, fig.height=2.4} maxtemp <- vic_elec |>   index_by(Day =
date(Time)) |>   summarise(Temperature = max(Temperature)) maxtemp |>
autoplot(Temperature) +   labs(y = "Max temperature")
```

Ansett airlines

Ansett airlines

```
{r, echo=TRUE, fig.height=3.1} ansett |>   autoplot(Passengers)
```

Ansett airlines

```
{r, echo=TRUE, fig.height=3.1} ansett |>   filter(Class == "Economy")
|>   autoplot(Passengers)
```

Ansett airlines

```
{r, echo=TRUE, fig.height=3.1} ansett |>   filter(Airports ==
"MEL-SYD") |>   autoplot(Passengers)
```

Ansett airlines

```
{r, echo=TRUE, fig.height=3.1} ansett |>   filter(Airports ==
"MEL-SYD") |>   autoplot(Passengers)
```

Not the real data! Or is it?

Outline

Lab Session 2

- Create time plots of the following four time series: `Bricks` from `aus_production`, `Lynx` from `pelt`, `Close` from `gafa_stock`, `Demand` from `vic_elec`.
- Use `help()` to find out about the data in each series.
- For the last plot, modify the axis labels and title.

Outline

Outline

Seasonal plots

- Data plotted against the individual "seasons" in which the data were observed. (In this case a "season" is a month.)
- Something like a time plot except that the data from each season are overlapped.
- Enables the underlying seasonal pattern to be seen more clearly, and also allows any substantial departures from the seasonal pattern to be easily identified.
- In R: `gg_season()`

Quarterly Australian Beer Production

```
{r, fig.height=2.7} beer <- aus_production |>
select(Quarter, Beer) |>   filter(year(Quarter) >=
1992) beer |> autoplot(Beer)
```

Quarterly Australian Beer Production

```
beer |> gg_season(Beer, labels = "right")
```

Multiple seasonal periods

```
vic_elec
```

Multiple seasonal periods

```
{r, dev = "ragg_png", dpi = 180} vic_elec |>
gg_season(Demand)
```

Multiple seasonal periods

```
{r, dev = "ragg_png", dpi = 180} vic_elec |>
gg_season(Demand, period = "week")
```

Multiple seasonal periods

```
{r, dev = "ragg_png", dpi = 180} vic_elec |>
gg_season(Demand, period = "day")
```

Seasonal subseries plots

- Data for each season collected together in time plot as separate time series.
- Enables the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.
- In R: `gg_subseries()`

Quarterly Australian Beer Production

```
beer |> gg_subseries(Beer)
```

Calendar plots

```
{r sugrrants, eval=FALSE} library(sugrrants) vic_elec
|>   filter(year(Date) == 2014) |>   mutate(Hour =
hour(Time)) |>   frame_calendar(x = Hour, y = Demand,
date = Date, nrow = 4) |>   ggplot(aes(x = .Hour, y =
.Demand, group = Date)) +   geom_line() -> p1
prettify(p1,   size = 3,   label.padding = unit(0.15,
"lines") )
```

- `frame_calendar()` makes a compact calendar plot
- `facet_calendar()` provides an easier ggplot2 integration.

Calendar plots

{r sugrrants2, ref.label="sugrrants", echo=FALSE, out.height="90%",

fig.height=5.4, fig.width=9}

Outline

Lab Session 3

1. Look at the quarterly tourism data for the Snowy Mountains

```
snowy <- tourism |>
  filter(Region == "Snowy Mountains")
```

   - Use `autoplot()`, `gg_season()` and `gg_subseries()` to explore the data.
   - What do you learn?

2. Produce a calendar plot for the `pedestrian` data from one location and one year.

Outline

Time series patterns

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

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).

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

Time series components

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

Time series patterns

```
{r, fig.height=2.7} aus_production |>   filter(year(Quarter) >= 1980) |>
autoplot(Electricity) +   labs(y = "GWh", title = "Australian electricity
production")
```

Time series patterns

```
{r, warning=FALSE, fig.height=2.7} aus_production |>   autoplot(Bricks) +
labs(title = "Australian clay brick production",        x = "Year", y =
"million units")
```

Time series patterns

{r, fig.height=2.7} us_employment |>   filter(Title == "Retail Trade", year(Month) >= 1980) |>   autoplot(Employed / 1e3) +   labs(title = "Retail employment, USA", y = "Million people")

Time series patterns

```
{r, fig.height=2.7} gafa_stock |>   filter(Symbol == "AMZN", year(Date) >= 2018) |>   autoplot(Close) +   labs(title = "Amazon closing stock price", x = "Day", y = "$")
```

Time series patterns

```
{r, fig.height=2.7} pelt |>   autoplot(Lynx) +   labs(title = "Annual Canadian
Lynx Trappings",        x = "Year", y = "Number trapped")
```

Seasonal or cyclic?

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

Seasonal or cyclic?

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

The timing of peaks and troughs is predictable with seasonal data, but unpredictable in the long term with cyclic data.

Outline

Example: Beer production

```r
new_production <- aus_production |>
  filter(year(Quarter) >= 1992)
new_production
```

Example: Beer production

```
{r, fig.height=6.5, fig.width=6.5, out.width="7cm"}
new_production |> gg_lag(Beer)
```

Example: Beer production

```
{r, fig.height=6.5, fig.width=6.5, out.width="7cm"}
new_production |> gg_lag(Beer, geom = "point")
```

Lagged scatterplots

- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.
- ACF (autocorrelation function):
  - $r_1 = \text{Correlation}(y_t, y_{t-1})$
  - $r_2 = \text{Correlation}(y_t, y_{t-2})$
  - $r_3 = \text{Correlation}(y_t, y_{t-3})$
  - etc.
- If there is **seasonality**, the ACF at the seasonal lag (e.g., 12 for monthly data) will be **large and positive**.

Autocorrelation

Results for first 9 lags for beer data:

```
{r, echo=TRUE} new_production |> ACF(Beer, lag_max = 9)
```

Autocorrelation

Results for first 9 lags for beer data:

```
{r beeracf, fig.height=2.5} new_production |>   ACF(Beer, lag_max = 9) |>

autoplot()
```

ACF

```
{r, fig.height=3, echo=TRUE} new_production |>
ACF(Beer) |>   autoplot()
```

Australian student enrolments

```
students <- readxl::read_excel("../data/schools/Table 42b Number of Full-time and Pa
  group_by(Year, State = substring(`State/Territory`, 3)) |>
  summarise(Count = sum(`All Full-time and Part-time Student count`), .groups = "drc
  as_tsibble(index = Year, key = State)
```

{r, echo=FALSE} students

Australian student enrolments

```
{r holidays-plot, echo=TRUE, dependson="holidays", fig.height=3.1} students |>

autoplot(Count) +   labs(y = "Student Count", title = "Australian students") +

scale_y_log10()
```

Australian holidays

```
students |> ACF(Count)
```

Australian holidays

```
{r tourismacf2, fig.height=5, fig.width=5,
out.width="49%"} students |> ACF(Count) |> autoplot()
```

Trend and seasonality in ACF plots

- When data have a trend, the autocorrelations for small lags tend to be large and positive.
- When data are seasonal, the autocorrelations will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency)
- When data are trended and seasonal, you see a combination of these effects.

US retail trade employment

```
retail <- us_employment |>
  filter(Title == "Retail Trade", year(Month) >= 1980)
retail |> autoplot(Employed)
```

US retail trade employment

```
retail |>
  ACF(Employed, lag_max = 48) |>
  autoplot()
```

Google stock price

```
google_2015 <- gafa_stock |>
  filter(Symbol == "GOOG", year(Date) == 2015) |>
  select(Date, Close)
google_2015
```

Google stock price

```
google_2015 |> autoplot(Close)
```

Google stock price

```
google_2015 |>
  ACF(Close, lag_max = 100) |>
  autoplot()
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. **Lab Session 4**
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Lab Session 4

We have introduced the following functions: `gg_lag` and `ACF`. Use these functions to explore the four time series: `Bricks` from `aus_production`, `Lynx` from `pelt`, `Close` price of Amazon from `gafa_stock`, `Demand` from `vic_elec`. Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

Which is which?

{r, fig.height=6, fig.width=12, echo=FALSE, warning=FALSE, out.width="100%"} cowtemp <- as_tsibble(fma::cowtemp) USAccDeaths <- as_tsibble(USAccDeaths) AirPassengers <- as_tsibble(AirPassengers) mink <- as_tsibble(fma::mink) tp1 <- autoplot(cowtemp, value) + labs(x = "") + labs(y = "chirps per minute") + labs(title = "1. Daily temperature of cow") tp2 <- autoplot(USAccDeaths, value) + labs(x = "") + labs(y = "thousands") +   labs(title = "2. Monthly accidental deaths") tp3 <- autoplot(AirPassengers, value) +

Outline

Example: White noise

{r, fig.height=3.3} wn <- tsibble(t = seq(36), y = rnorm(36), index = t) wn |> autoplot(y)

Example: White noise

```
{r, fig.height=3.3} wn <- tsibble(t = seq(36), y = rnorm(36), index =
t) wn |> autoplot(y)
```

White noise data is uncorrelated across time with zero mean and constant variance.
(Technically, we require independence as well.)

Example: White noise

```
wn |> ACF(y)
```

```
{r wnacf, echo=FALSE, dependson="wn"} wn |>   ACF(y, lag_max = 10) |>
as_tibble() |>   mutate(lag = as.numeric(lag)) |>   pivot_wider(names_from =
lag, values_from = acf) |>   rename_all(function(x) {     paste("$r_{", x,
"}$", sep = "")   }) |>   knitr::kable(     booktabs = TRUE,     escape =
FALSE, align = "c", digits = 3,     format.args = list(nsmall = 3)   )
```

```
{r, echo=FALSE, fig.height=1.5} wn |>   ACF(y) |>   autoplot()
```

Example: White noise

```r
wn |> ACF(y)
```

```
{r wnacf, echo=FALSE, dependson="wn"} wn |>   ACF(y, lag_max = 10) |>
as_tibble() |>   mutate(lag = as.numeric(lag)) |>   pivot_wider(names_from =
lag, values_from = acf) |>   rename_all(function(x) {     paste("$r_{", x,
"}$", sep = "")   }) |>   knitr::kable(     booktabs = TRUE,     escape =
FALSE, align = "c", digits = 3,     format.args = list(nsmall = 3)   )
```

```
{r, echo=FALSE, fig.height=1.5} wn |>   ACF(y) |>   autoplot()
```

- Sample autocorrelations for white noise series.
- Expect each autocorrelation to be close to zero.
- Blue lines show 95% critical values.

Example: Pigs slaughtered

```
{r, fig.height=2.5} pigs <- aus_livestock |>   filter(State ==
"Victoria", Animal == "Pigs", year(Month) >= 2014) pigs |>
autoplot(Count / 1e3) +   labs(x = "Year", y = "Thousands",
title = "Number of pigs slaughtered in Victoria")
```

Example: Pigs slaughtered

```
pigs |>
  ACF(Count) |>
  autoplot()
```

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows significant autocorrelation for lag 2 and 12.
- Indicate some slight seasonality.

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows significant autocorrelation for lag 2 and 12.
- Indicate some slight seasonality.

These show the series is **not a white noise series**.

Outline

Lab Session 5

Calculate the difference in ACT student enrolments, it can be done as follows:

```
{r, eval = FALSE} students |>   filter(State == "ACT") |>   mutate(diff = difference(Count))
```

Does `diff` look like white noise?

Outline

Outline

Per capita adjustments

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

Per capita adjustments

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

Outline

Lab Session 6

Consider the GDP information in `global_economy`. Plot the GDP per capita for each country over time. Which country has the highest GDP per capita? How has this changed over time?

Outline

Inflation adjustments

{r, eval=FALSE} print_retail <- aus_retail |>   filter(Industry == "Newspaper and book retailing") |>   group_by(Industry) |>   index_by(Year = year(Month)) |>   summarise(Turnover = sum(Turnover)) aus_economy <- filter(global_economy, Country == "Australia") print_retail |>   left_join(aus_economy, by = "Year") |>   mutate(Adj_turnover = Turnover / CPI) |>   pivot_longer(c(Turnover, Adj_turnover),   names_to = "Type", values_to = "Turnover"   ) |> ggplot(aes(x = Year, y = Turnover)) +   geom_line() +   facet_grid(vars(Type), scales = "free_y") +   labs(x = "Years", y = NULL,   title = "Turnover: Australian print media industry")

Inflation adjustments

```
{r, message=FALSE, warning=FALSE, echo=FALSE, fig.height=5, out.height="90%"}

print_retail <- aus_retail |>   filter(Industry == "Newspaper and book

retailing") |>   group_by(Industry) |>   index_by(Year = year(Month)) |>

summarise(Turnover = sum(Turnover)) aus_economy <-

filter(tsibbledata::global_economy, Country == "Australia") print_retail |>

left_join(aus_economy, by = "Year") |>   mutate(Adj_turnover = Turnover / CPI)

|>   pivot_longer(c(Turnover, Adj_turnover),     names_to = "Type", values_to

= "Turnover"   ) |>   ggplot(aes(x = Year, y = Turnover)) +   geom_line() +

facet_grid(vars(Type), scales = "free_y") +   labs(     x = "Years", y = NULL,

title = "Turnover: Australian print media industry"   )
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. **Mathematical transformations**
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Variance stabilization

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

Variance stabilization

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

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

Variance stabilization

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

Variance stabilization

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

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

Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	$\downarrow$
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**

Variance stabilization

```
{r food, echo=TRUE} food <- aus_retail |>   filter(Industry ==
"Food retailing") |>   summarise(Turnover = sum(Turnover))
```

```
{r food-plot, echo = FALSE, fig.height=3.8} food |>
autoplot(Turnover) +   labs(y = "Turnover ($AUD)")
```

Variance stabilization

```
{r food-sqrt1, echo=TRUE, fig.height=3.8} food |>
autoplot(sqrt(Turnover)) +   labs(y = "Square root turnover")
```

Variance stabilization

```
{r food-cbrt, echo=TRUE, fig.height=3.8} food |>
autoplot(Turnoverˆ(1 / 3)) +    labs(y = "Cube root turnover")
```

Variance stabilization

```
{r food-log, echo=TRUE, fig.height=3.8} food |>
autoplot(log(Turnover)) +   labs(y = "Log turnover")
```

Variance stabilization

```
{r food-inverse, echo=TRUE, fig.height=3.8} food |> autoplot(-1 /
Turnover) +   labs(y = "Inverse turnover")
```

Box-Cox transformations

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

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

Box-Cox transformations

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

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

- 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)

Box-Cox transformations

```
{r food-anim, echo=FALSE, fig.show='animate',
interval=1/10, message=FALSE, fig.height=5,
fig.width=8,
aniopts='controls,buttonsize=0.3cm,width=11.5cm'}
library(rlang) library(gganimate) library(latex2exp)
food |>   mutate(!!!set_names(map(seq(0, 1, 0.01), ~
expr(fabletools::box_cox(Turnover, !!.x))), seq(0, 1,
0.01))) |>   select(-Turnover) |>
pivot_longer(-Month, names_to = "lambda", values_to =
"Turnover") |>   mutate(lambda = as.numeric(lambda))
|>   ggplot(aes(x = Month, y = Turnover)) +
```

Box-Cox transformations

```
{r food-lambda, echo=TRUE} food |>   features(Turnover,
features = guerrero)
```

Box-Cox transformations

```
{r food-lambda, echo=TRUE} food |>   features(Turnover,
features = guerrero)
```

- 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

```
{r food-bc, echo=TRUE,fig.height=3.8} food |>
autoplot(box_cox(Turnover, 0.0895)) +   labs(y = "Box-Cox
transformed turnover")
```

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

Lab Session 7

1. For the following series, find an appropriate transformation in order to stabilise the variance.
   - United States GDP from `global_economy`
   - Slaughter of Victorian "Bulls, bullocks and steers" in `aus_livestock`
   - Victorian Electricity Demand from `vic_elec`.
   - Gas production from `aus_production`
2. Why is a Box-Cox transformation unhelpful for the `canadian_gas` data?

Outline

Outline

Time series decomposition

Trend-Cycle aperiodic changes in level over time.

Seasonal (almost) periodic changes in level due to seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Additive decomposition

$$y_t = S_t + T_t + R_t$$

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

STL decomposition

- STL: “Seasonal and Trend decomposition using Loess”
- Very versatile and robust.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Optionally robust to outliers
- No trading day or calendar adjustments.
- Only additive.
- Take logs to get multiplicative decomposition.
- Use Box-Cox transformations to get other decompositions.

US Retail Employment

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

US Retail Employment

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

US Retail Employment

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

US Retail Employment

```
components(dcmp)
```

US Retail Employment

```
{r usretail-stl, fig.width=8, fig.height=4}
components(dcmp) |> autoplot()
```

US Retail Employment

```
{r dable4, fig.height=2.7} us_retail_employment |>
autoplot(Employed, color = "gray") +
autolayer(components(dcmp), trend, color = "#D55E00")
+   labs(y = "Persons (thousands)", title = "Total
employment in US retail")
```

US Retail Employment

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

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

{r usretail-sa, fig.height=2.7} us_retail_employment |>   autoplot(Employed, color = "gray") + autolayer(components(dcmp), season_adjust, color = "#0072B2") +   labs(y = "Persons (thousands)", title = "Total employment in US retail")

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.

STL decomposition

“‘{r stlwindowanim, echo=FALSE, warning=FALSE, message=FALSE, fig.show=‘animate’, interval=1/10, fig.height=5.35, fig.width=8, aniopts=‘controls,buttonsize=0.3cm,width=11.5cm’, eval=TRUE} s_windows <- seq(5, 55, by = 2) stl_defs <- purrr::map(s_windows, function(s_window) { STL(Employed ~ season(window = s_window), robust = TRUE) }) names(stl_defs) <- sprintf(“season(window=%02d)”, s_windows)

us_retail_employment |> model(!!!stl_defs) |> components() |> as_tibble() |> pivot_longer(Employed:remainder, names_to = “component”, names_ptypes = list(component = factor(levels =

STL decomposition

```
{r mstl, fig.width=8.5, fig.height=3.4}
us_retail_employment |>   model(STL(Employed)) |>
components() |>   autoplot()
```

STL decomposition

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

```
{r mstl, fig.width=8.5, fig
us_retail_employment |>   model(STL(Employed)) |>
components() |>   autoplot()
```

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.
- trend window controls loess bandwidth on deasonalised values.
- season window controls loess bandwidth on detrended subseries.
- Robustness weights based on remainder.
- Default season: `window = 13`
- Default trend:

`window = nextodd(ceiling((1.5*period)/(1-(1.5/s.window))))`

Australian holidays

```
{r holidays, include=FALSE} holidays <- tourism |>
filter(Purpose == "Holiday") |>   group_by(State) |>
summarise(Trips = sum(Trips))

{r holidays-plot2, echo=TRUE, dependson="holidays",
fig.height=2.8} holidays |> autoplot(Trips) +   labs(y
= "thousands of trips", x = "Year",        title =
"Australian domestic holiday nights")
```

Australian holidays

```
{r stlagain2, echo=TRUE, warning=FALSE, fig.width=8,
fig.height=3} holidays |>   model(stl = STL(Trips)) |>
components() |>   autoplot()
```

Holidays decomposition

```
dcmp <- holidays |>
  model(stl = STL(Trips)) |>
  components()
dcmp
```

Holidays decomposition

```
{r holidays3, fig.height=4.6} dcmp |> gg_subseries(season_year)
```

Holidays decomposition

```
{r holidays-trend, message=FALSE, warning=FALSE, fig.height=3.5, out.height="70%"}

autoplot(dcmp, trend, scale_bars = FALSE) +   autolayer(holidays, alpha = 0.4)
```

Outline

Lab Session 8

1. Produce the following decomposition

```
canadian_gas |>
  model(STL(Volume ~ season(window=7) + trend(window=11))) |>
  components() |>
  autoplot()
```

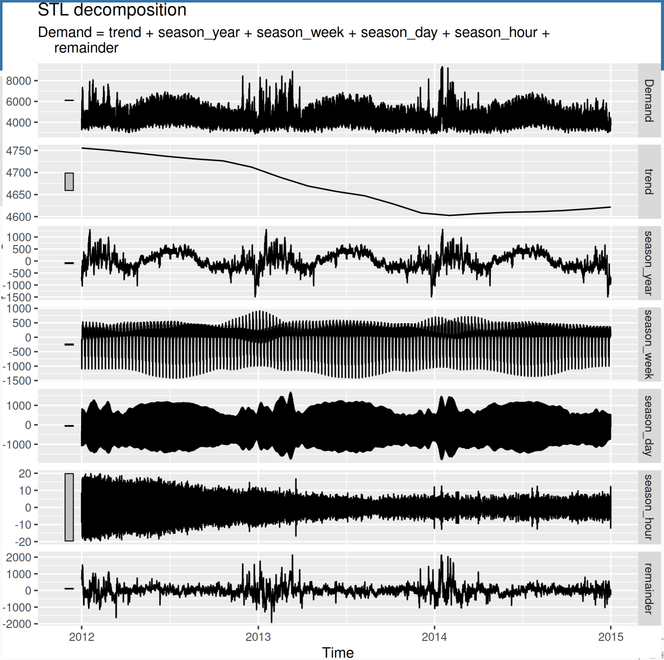
2. What happens as you change the values of the two `window` arguments?
3. How does the seasonal shape change over time? [Hint: Try plotting the seasonal component using `gg_season`.]
4. Can you produce a plausible seasonally adjusted series? [Hint: `season_adjust` is one of the variables returned by `STL`.]

Outline

Multiple seasonality

```
vic_elec |>
  model(STL(Demand)) |>
  components() |>
  autoplot()
```

{r vic_elec, echo=FALSE} png("figs/vi
300, units = "cm", type = "cairo-png"
components() |>   autoplot() crop::de



Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. **A cautionary tale**
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

A cautionary tale

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

A cautionary tale

NEWS

LOCATION:
Clayton, Vic Change

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

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

A cautionary tale

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:

A cautionary tale

```
{r abs1, echo=FALSE} employed <- tsibble(   Time =
yearmonth("1978 Feb") + 0:439,   Employed = c(
5985.7, 6040.6, 6054.2, 6038.3, 6031.3, 6036.1,
6005.4, 6024.3, 6045.9, 6033.8, 6125.4, 5971.3,
6050.7, 6096.2, 6087.7, 6075.6, 6095.7, 6103.9,
6078.5, 6157.8, 6164.0, 6188.8, 6257.2, 6112.9,
6207.2, 6278.7, 6224.9, 6273.4, 6269.9, 6314.1,
6281.4, 6360.0, 6320.2, 6342.0, 6426.6, 6253.0,
6356.5, 6428.1, 6426.3, 6412.4, 6413.9, 6425.3,
6393.7, 6502.7, 6445.3, 6433.3, 6506.9, 6355.5,
6432.4, 6497.4, 6431.6, 6440.9, 6414.3, 6425.9,
```

A cautionary tale

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

A cautionary tale

```
{r abs4, fig.height=2.8} employed |>   filter(Year >= 2005) |>   autoplot(Employed) +   labs(title = "Total employed", y = "Thousands")
```

A cautionary tale

```
{r abs5, fig.height=2.8} employed |>   filter(Year >=
2005) |>   gg_season(Employed, labels = "right") +
labs(title = "Total employed", y = "Thousands")
```

A cautionary tale

```
{r abs6, fig.height=2} 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)
```

A cautionary tale

```
{r abs7, fig.height=2.8} dcmp <- employed |>
filter(Year >= 2005) |>   model(stl = STL(Employed ~
season(window = 11), robust = TRUE)) components(dcmp)
|> autoplot()
```

A cautionary tale

```
{r abs8, fig.height=2.8} components(dcmp) |>
filter(year(Time) == 2013) |>   gg_season(season_year)
+   labs(title = "Seasonal component") + guides(colour
= "none")
```

A cautionary tale

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

A cautionary tale

- 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. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Outline

Notice: Material planned to change

This material is planned to be updated to better align with the training needs of the Department of Education.

In particular, this section will be removed to make time for:

- econometric concepts
- multivariate models.

Outline

Strength of seasonality and trend

STL decomposition

$$y_t = T_t + S_t + R_t$$

Seasonal strength

$$\max\left(0, 1 - \frac{\text{Var}(R_t)}{\text{Var}(S_t + R_t)}\right)$$

Trend strength

$$\max\left(0, 1 - \frac{\text{Var}(R_t)}{\text{Var}(T_t + R_t)}\right)$$

Feature extraction and statistics

```
tourism |> features(Trips, feat_stl)
```

Feature extraction and statistics

```
#| label: features-plot
#| fig-height: 3.6
tourism |>
  features(Trips, feat_stl) |>
  ggplot(aes(x = trend_strength, y = seasonal_strength_year, col = Purpose)) +
  geom_point() + facet_wrap(vars(State))
```

Feature extraction and statistics

```
#| label: features-plot
#| fig-height: 3.6
tourism |>
  features(Trips, feat_stl) |>
  ggplot(aes(x = trend_strength, y = seasonal_strength_year, col = Purpose)) +
  geom_point() + facet_wrap(vars(State))
```

- Holidays more seasonal than other travel.
- WA has strongest trends.

Feature extraction and statistics

Find the most seasonal time series:

```
most_seasonal <- tourism |>
  features(Trips, feat_stl) |>
  filter(seasonal_strength_year == max(seasonal_strength_year))
```

Feature extraction and statistics

Find the most seasonal time series:

```
most_seasonal <- tourism |>
  features(Trips, feat_stl) |>
  filter(seasonal_strength_year == max(seasonal_strength_year))
```

`{r extreme2, fig.height=1.8} tourism |>   right_join(most_seasonal, by = c("State", "Region", "Purpose")) |>   ggplot(aes(x = Quarter, y = Trips)) +   geom_line() +   facet_grid(vars(State, Region, Purpose))`

Feature extraction and statistics

Find the most trended time series:

```
most_trended <- tourism |>
  features(Trips, feat_stl) |>
  filter(trend_strength == max(trend_strength))
```

Feature extraction and statistics

Find the most trended time series:

```
most_trended <- tourism |>
  features(Trips, feat_stl) |>
  filter(trend_strength == max(trend_strength))
```

```
{r extreme4, fig.height=1.8} tourism |>   right_join(most_trended, by = c("State",
"Region", "Purpose")) |>   ggplot(aes(x = Quarter, y = Trips)) +   geom_line() +
facet_grid(vars(State, Region, Purpose))
```

Outline

Lab Session 9

- Use `GGally::ggpairs()` to look at the relationships between the STL-based features. You might wish to change `seasonal_peak_year` and `seasonal_trough_year` to factors.
- Which is the peak quarter for holidays in each state?

Feature extraction and statistics

```
tourism |> features(Trips, feat_acf)
```

Outline

Feature extraction and statistics

All features from the feasts package

```
#| echo: false
#| include: false
# Save pdf figures
savepdf <- function(file, width = 16, height = 10) {
  fname <<- paste("figs/", file, ".pdf", sep = "")
  Cairo::CairoPDF(fname, width = width / 2.54, height = height / 2.54, pointsize = 1
  par(mgp = c(2.2, 0.45, 0), tcl = -0.4, mar = c(3.3, 3.6, 1.1, 1.1))
}
endpdf <- function() {
  crop::dev.off.crop(fname)
}
# Compute features
tourism_features <- tourism |>
  features(Trips, feature_set(pkgs = "feasts"))
# Compute PCs
pcs <- tourism_features |>
  select(-State, -Region, -Purpose) |>
  prcomp(scale = TRUE) |>
  broom::augment(tourism_features)
```

Feature extraction and statistics

```
pcs <- tourism_features |>
  select(-State, -Region, -Purpose) |>
  prcomp(scale = TRUE) |>
  broom::augment(tourism_features)
```

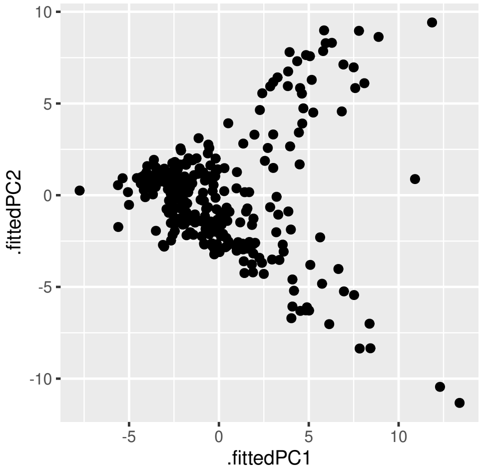
Principal components based on all features from the feasts package

```
{r echo=FALSE} pcs
```

Feature extraction and statistics

```r
pcs |> ggplot(aes(x=.fittedPC1, y=.fittedPC2)) +
  geom_point() + theme(aspect.ratio=1)
```

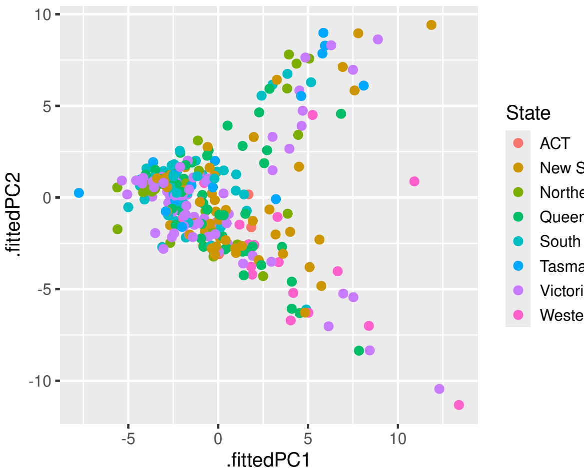
Principal components based on all features from the feasts package



Feature extraction and statistics

```
pcs |> ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=State)) +
  geom_point() + theme(aspect.ratio=1)
```

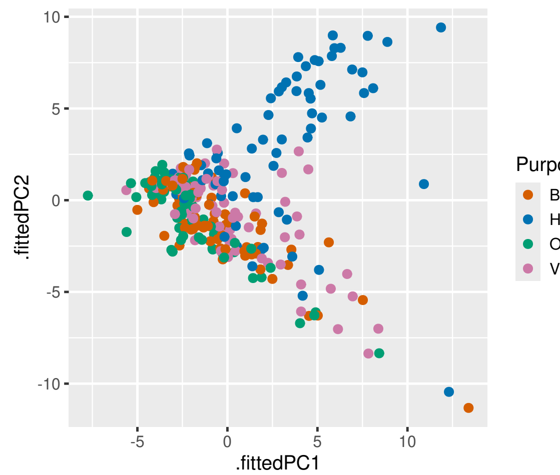
Principal components based on all features from the feasts package



Feature extraction and statistics

```
pcs |> ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=Purpose)) +
  geom_point() + theme(aspect.ratio=1)
```

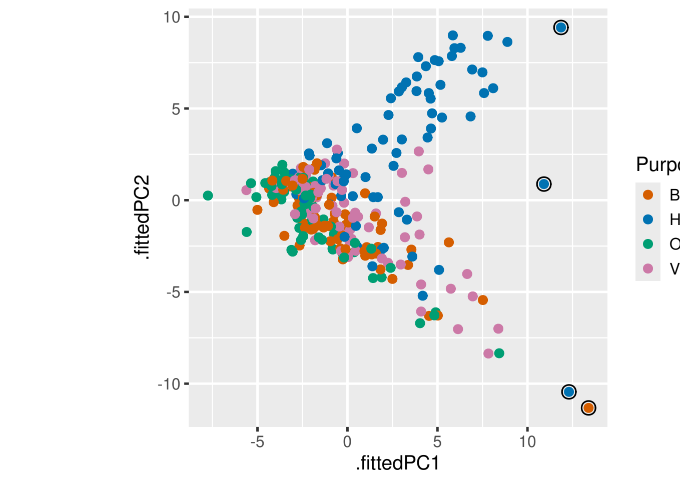
Principal components based on all features from the feasts package



Feature extraction and statistics

```
pcs |> ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=Purpose)) +
  geom_point() + theme(aspect.ratio=1)
```

Principal components based on all features from the feasts package



Feature extraction and statistics

```
{r outliers2, fig.height=4.5, fig.width=12, out.height="55%"} outliers |>   left_join(tourism, by
= c("State", "Region", "Purpose")) |>   mutate(Series = glue("{State}", "{Region}", "{Purpose}",
.sep = "\n\n")) |>   ggplot(aes(x = Quarter, y = Trips)) +   geom_line() + facet_grid(Series ~ .)
+   labs(title = "Outlying time series in PC space")
```

Outline

Lab Session 10

- Use a feature-based approach to look for outlying series in PBS.
- What is unusual about the series you identify as outliers?

Outline

Outline

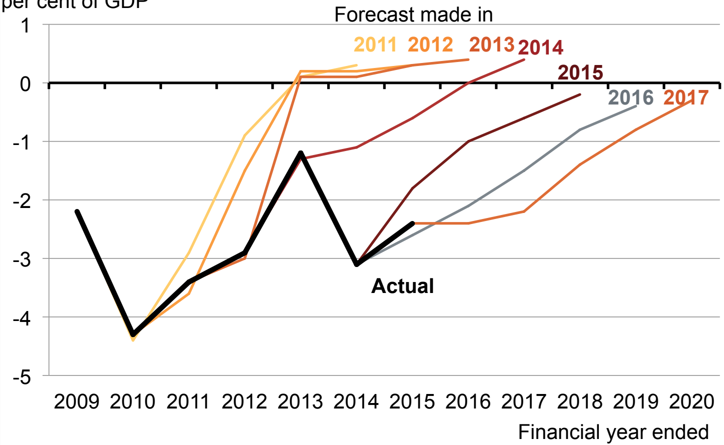
Forecasting is difficult

Commonwealth plans to drift back to surplus show the triumph of experience over hope

GRATTAN Institute

Actual and forecast Commonwealth underlying cash balance

per cent of GDP



Grattan analysis of Commonwealth Budget Papers

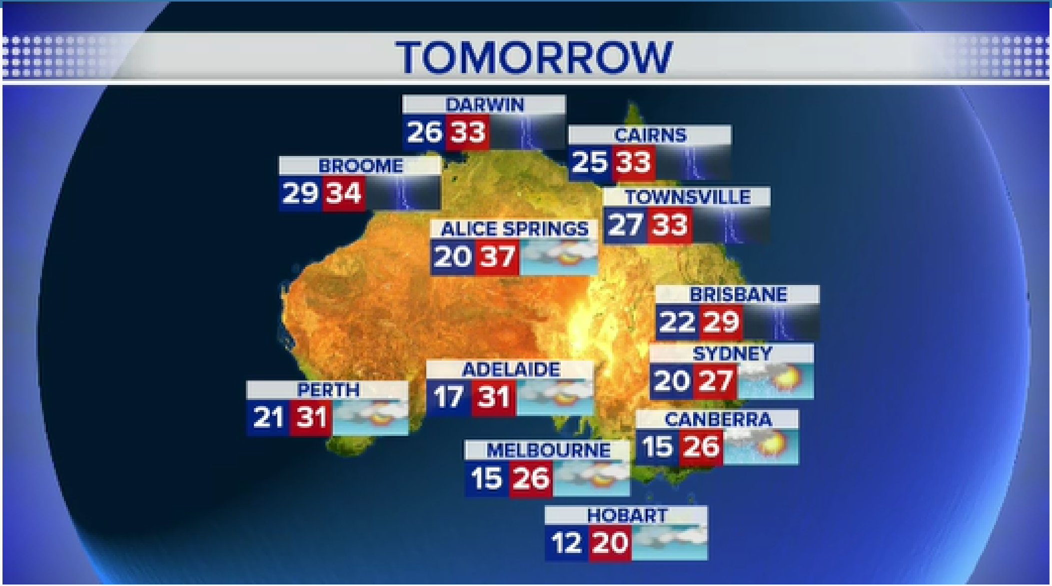
What can we forecast?

What can we forecast?

What can we forecast?

What can we forecast?

What can we forecast?



What can we forecast?

What can we forecast?

Which is easiest to forecast?

1. daily electricity demand in 3 days time
2. timing of next Halley's comet appearance
3. time of sunrise this day next year
4. Google stock price tomorrow
5. Google stock price in 6 months time
6. maximum temperature tomorrow
7. exchange rate of $US/AUS next week
8. total sales of drugs in Australian pharmacies next month

Which is easiest to forecast?

1. daily electricity demand in 3 days time
2. timing of next Halley's comet appearance
3. time of sunrise this day next year
4. Google stock price tomorrow
5. Google stock price in 6 months time
6. maximum temperature tomorrow
7. exchange rate of $US/AUS next week
8. total sales of drugs in Australian pharmacies next month

- how do we measure "easiest"?
- what makes something easy/difficult to forecast?

Factors affecting forecastability

Something is easier to forecast if:

- we have a good understanding of the factors that contribute to it
- there is lots of data available;
- the forecasts cannot affect the thing we are trying to forecast.
- there is relatively low natural/unexplainable random variation.
- the future is somewhat similar to the past

Random futures

```
{r austa, echo=FALSE} # Grab ABS data austa <-
readxl::read_excel("data/340101.xlsx", sheet =
"Data1", skip = 9) |>   rename(date = `Series ID`,
value = A85375847A) |>   select(date, value) |>
transmute(     Month = yearmonth(date),     Visitors =
value / 1e3   ) |>   bind_rows(tibble(     Month =
yearmonth(seq(as.Date("2021-11-01"), by = "1 month",
length = 2)),     Visitors = NA_real_   )) |>
as_tsibble(index = Month) |>   filter(Month >=
yearmonth("2000 Jan")) # Fit ETS model fit <- austa |>
filter(Month < yearmonth("2018 Jan")) |>
```

Random futures

```
{r austa, echo=FALSE} # Grab ABS data austa <-
readxl::read_excel("data/340101.xlsx", sheet =
"Data1", skip = 9) |>   rename(date = `Series ID`,
value = A85375847A) |>   select(date, value) |>
transmute(     Month = yearmonth(date),     Visitors =
value / 1e3   ) |>   bind_rows(tibble(     Month =
yearmonth(seq(as.Date("2021-11-01"), by = "1 month",
length = 2)),     Visitors = NA_real_   )) |>
as_tsibble(index = Month) |>   filter(Month >=
yearmonth("2000 Jan")) # Fit ETS model fit <- austa |>
filter(Month < yearmonth("2018 Jan")) |>
```

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa2, dependson='austa', echo=FALSE}
aligned_plots[[2]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa3, dependson='austa', echo=FALSE}
aligned_plots[[3]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa4, dependson='austa', echo=FALSE}
aligned_plots[[4]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa5, dependson='austa', echo=FALSE}
aligned_plots[[5]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa6, dependson='austa', echo=FALSE}
aligned_plots[[6]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa7, dependson='austa', echo=FALSE}
aligned_plots[[7]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa8, dependson='austa', echo=FALSE}
aligned_plots[[8]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa9, dependson='austa', echo=FALSE}
aligned_plots[[9]]
```

Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa9b, dependson='austa', echo=FALSE}
aligned_plots[[9]] +   geom_line(     colour =
"black",      data = austa |> filter(Month >=
yearmonth("2018 Jan")) |> mutate(Month =
as.Date(Month))    )
```

Simulated futures
from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.

```
{r austa9b, dependson='austa', echo=FALSE}
aligned_plots[[9]] +   geom_line(      colour =
"black",      data = austa |> filter(Month >=
yearmonth("2018 Jan")) |> mutate(Month =
as.Date(Month))    )
```

"He who sees the past as surprise-free is bound to have a future full of surprises."
(Amos Tversky)

Simulated futures from an ETS model

Statistical forecasting

- Thing to be forecast: y_{T+h}.
- What we know: $y_1, \dots, y_T$.
- Forecast distribution: $y_{T+h|t} = y_{T+h} \mid \{y_1, y_2, \dots, y_T\}$.
- Point forecast: $\hat{y}_{T+h|T} = \text{E}[y_{T+h} \mid y_1, \dots, y_T]$.
- Forecast variance: $\text{Var}[y_t \mid y_1, \dots, y_T]$
- Prediction interval is a range of values of y_{T+h} with high probability.

Outline

Some simple forecasting methods

```
{r ausbeer, fig.height=4.6, echo=FALSE} new_production
<- aus_production |>   filter(year(Quarter) >= 1992)
new_production |> autoplot(Beer) +   labs(     x =
"Year", y = "megalitres",     title = "Australian
quarterly beer production"   )
```

How would you forecast these series?

Some simple forecasting methods

```
{r pigs, fig.height=4.6, echo=FALSE} aus_livestock |>
filter(     between(year(Month), 1992, 1996),
Animal == "Pigs", State == "Victoria"    ) |>
autoplot(Count) +   labs(     x = "Year", y =
"thousands",      title = "Number of pigs slaughtered
in Victoria, 1990-1995"    )
```

How would you forecast these series?

Some simple forecasting methods

```
{r dj, fig.height=4.6, echo=FALSE} gafa_stock |>
filter(Symbol == "FB", Date >= ymd("2018-01-01")) |>
autoplot(Close) +   labs(     title = "Facebook
closing stock price in 2018",     x = "Date", y =
"Closing price ($USD)"   )
```

How would you forecast these series?

Some simple forecasting methods

MEAN(y): Average method

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

"'{r mean-method-explained, echo=FALSE, message=FALSE, warning=FALSE, fig.height = 3.4} bricks <- aus_production |> filter(!is.na(Bricks)) |> mutate(average = mean(Bricks))
fc <- bricks |> model(MEAN(Bricks)) |> forecast(h = "5 years")
bricks |> ggplot(aes(x = Quarter, y = Bricks)) + geom_line() + geom_line(aes(y = average), colour = "blue", linetype = "dashed") + geom_line(aes(y = .mean), data = fc, colour = "blue") + labs(title = "Clay brick production in Australia")

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 $(h-1)/m$.

```
{r snaive-method-explained, echo = FALSE, warning = FALSE,
fig.height = 3.4} bricks |>   model(SNAIVE(Bricks ~
lag("year"))) |>   forecast(h = "5 years") |>
autoplot(filter(bricks, year(Quarter) > 1990), level = NULL)
+   geom_point(aes(y = Bricks), data = slice(bricks, (n() -
3):n()), colour = "blue") +   labs(title = "Clay brick
production in Australia")
```

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

```
{r drift-method-explained, echo = FALSE, warning =
FALSE} aus_production |>   filter(!is.na(Bricks)) |>
model(RW(Bricks ~ drift())) |>   forecast(h = "5
years") |>   autoplot(aus_production, level = NULL) +
geom_line(     aes(y = Bricks),     data =
slice(aus_production, range(cumsum(!is.na(Bricks)))),
linetype = "dashed", colour = "blue"    ) +
labs(title = "Clay brick production in Australia")
```

Model fitting

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

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

```
{r brick-model2, echo=FALSE, dependson='brick-model'} brick_fit
```

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

Producing forecasts

```
{r brick-fc, echo = TRUE, dependson='brick-model'} brick_fc <- brick_fit |>
forecast(h = "5 years")
```

```
{r brick-fbl, echo = FALSE, dependson='brick-fc'} print(brick_fc, n = 4)
```

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

Visualising forecasts

```
{r brick-fc-plot, warning=FALSE, message=FALSE, fig.height=3.4,
dependson='brick-fc'} brick_fc |>   autoplot(aus_production, level = NULL) +
labs(title = "Forecasts for quarterly clay brick production",         x =
"Year", y = "Millions of bricks") +   guides(colour = guide_legend(title =
"Forecast"))
```

Prediction intervals

```
{r brick-fc-interval, dependson='brick-fc'} brick_fc |>   hilo(level = c(50, 75))
```

Prediction intervals

{r brick-fc-interval2, dependson='brick-fc'} brick_fc |>   hilo(level = c(50, 75)) |>   mutate(lower = `50%`$lower, upper = `50%`$upper)

Outline

Lab Session 11

- Produce forecasts using an appropriate benchmark method for student enrolments in Australia (as shown yesterday). Plot the results using `autoplot()`.
- Produce forecasts using an appropriate benchmark method for total Australian retail turnover (aggregate `aus_retail`). Plot the results using `autoplot()`.

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Fitted values

- $\hat{y}_{t|t-1}$ is the forecast of y_t based on observations $y_1, \dots, y_t$.
- 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: $e_t = y_t - \hat{y}_{t|t-1}$.

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.

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 prediction intervals)

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

Facebook closing stock price

```
fb_stock <- gafa_stock |>
  filter(Symbol == "FB")
fb_stock |> autoplot(Close)
```

Facebook closing stock price

```
fb_stock <- fb_stock |>
  mutate(trading_day = row_number()) |>
  update_tsibble(index = trading_day, regular = TRUE)
fit <- fb_stock |> model(NAIVE(Close))
augment(fit)
```

Facebook closing stock price

```
{r dj4, echo=TRUE, warning=FALSE, fig.height=3.4, dependson="augment"}

augment(fit) |>   ggplot(aes(x = trading_day)) +   geom_line(aes(y = Close,

colour = "Data")) +   geom_line(aes(y = .fitted, colour = "Fitted"))
```

Facebook closing stock price

```
{r dj4a, echo=TRUE, warning=FALSE, fig.height=3.4, dependson="augment"}

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"))
```

Facebook closing stock price

{r dj5, echo=TRUE, warning = FALSE, dependson="augment"} augment(fit) |> autoplot(.resid) +   labs(x = "Day", y = "", title = "Residuals from naïve method")

Facebook closing stock price

```
{r dj6, warning=FALSE, fig.height=3.4, dependson="augment"}
augment(fit) |>   ggplot(aes(x = .resid)) +   geom_histogram(bins =
150) +   labs(title = "Histogram of residuals")
```

Facebook closing stock price

```
{r dj7, dependson="augment"} augment(fit) |>   ACF(.resid) |>

autoplot() + labs(title = "ACF of residuals")
```

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.

Combined diagnostic graph

```
{r dj8, dependson="augment"} fit |> gg_tsresiduals()
```

Ljung-Box test

Test whether *whole set* of r_k values is significantly different from zero set.

$$Q = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2 \qquad \text{where } \ell = \text{max lag and } T = \text{\# observations}$$

- If each r_k close to zero, Q will be **small**.
- If some r_k values large (+ or −), Q will be **large**.
- My preferences: $h = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- If data are WN and T large, $Q \sim \chi^2$ with ℓ degrees of freedom.

Ljung-Box test

$$Q = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2$$ where ℓ = max lag and T = # observations.

```
{r dj9extra, echo=FALSE, fig.height=1.65} augment(fit) |>   ACF(.resid,
lag_max = 10) |>   autoplot() + labs(title = "ACF of residuals")
```

```
{r dj9, echo=TRUE, dependson="augment"} # lag = h augment(fit) |>
```

```
features(.resid, ljung_box, lag = 10)
```

Outline

Lab Session 12

- Compute RW w/ drift forecasts for total student enrolments in Australia.
- Test if the residuals are white noise. What do you conclude?

Outline

Training and test sets

```
{r traintest, fig.height=1, echo=FALSE} train <- 1:18
test <- 19:24 par(mar = c(0, 0, 0, 0)) plot(0, 0, xlim
= c(0, 26), ylim = c(0, 2), xaxt = "n", yaxt = "n",
bty = "n", xlab = "", ylab = "", type = "n") arrows(0,
0.5, 25, 0.5, 0.05) points(train, train * 0 + 0.5, pch
= 19, col = "blue") points(test, test * 0 + 0.5, pch =
19, col = "red") text(26, 0.5, "time") text(10, 1,
"Training data", col = "blue") text(21, 1, "Test
data", col = "red")
```

- A model which fits the training data well will not necessarily forecast well.

Measures of forecast accuracy

```
beer_fit <- aus_production |>
  filter(between(year(Quarter), 1992, 2007)) |>
  model(
    snaive = SNAIVE(Beer),
    mean = MEAN(Beer)
  )
beer_fit |>
  forecast(h = "3 years") |>
  autoplot(aus_production, level = NULL) +
  labs(title ="Forecasts for quarterly beer production",
       x ="Year", y ="Megalitres") +
  guides(colour = guide_legend(title = "Forecast"))
```

Measures of forecast accuracy

```
{r beer-fc-1, echo=FALSE, fig.height=4} beer_fit <-
aus_production |>   filter(between(year(Quarter),
1992, 2007)) |>   model(      snaive = SNAIVE(Beer),
mean = MEAN(Beer)   ) beer_fit |>   forecast(h = "3
years") |>   autoplot(aus_production, level = NULL) +
labs(     title = "Forecasts for quarterly beer
production",      x = "Year", y = "Megalitres"   ) +
guides(colour = guide_legend(title = "Forecast"))
```

Measures of forecast accuracy

$$
\begin{aligned}
y_{T+h} &= (T+h)\text{th observation, } h = 1, \dots, H \\
\hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\
e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T}
\end{aligned}
$$

$$
\begin{aligned}
\text{MAE} &= \text{mean}(|e_{T+h}|) \\
\text{MSE} &= \text{mean}(e_{T+h}^2) \qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)} \\
\text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|)
\end{aligned}
$$

Measures of forecast accuracy

$$
\begin{aligned}
y_{T+h} &= (T+h)\text{th observation, } h = 1,\dots,H \\
\hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\
e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T}
\end{aligned}
$$

$$
\begin{aligned}
\text{MAE} &= \text{mean}(|e_{T+h}|) \\
\text{MSE} &= \text{mean}(e_{T+h}^2) \qquad\qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)} \\
\text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|)
\end{aligned}
$$

- 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

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

Measures of forecast accuracy

Mean Absolute Scaled Error

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

- For non-seasonal series, scale uses naïve forecasts:
$$Q = \frac{1}{T-1}\sum_{t=2}^{T} |y_t - y_{t-1}|$$
- For seasonal series, scale uses seasonal naïve forecasts:
$$Q = \frac{1}{T-m}\sum_{t=m+1}^{T} |y_t - y_{t-m}|$$

where m is the seasonal frequency

Measures of forecast accuracy

Mean Absolute Scaled Error

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

- For non-seasonal series, scale uses naïve forecasts:

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

- For seasonal series, scale uses seasonal naïve forecasts:

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

where m is the seasonal frequency

Proposed by Hyndman and Koehler (IJF, 2006).

Measures of forecast accuracy

Root Mean Squared Scaled Error

$$\text{RMSSE} = \sqrt{\text{mean}(e_{T+h}^2/Q)}$$

- For non-seasonal series, scale uses naïve forecasts:
$$Q = \frac{1}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})^2$$
- For seasonal series, scale uses seasonal naïve forecasts:
$$Q = \frac{1}{T-m}\sum_{t=m+1}^{T}(y_t - y_{t-m})^2$$
where m is the seasonal frequencyq

Proposed by Hyndman and Koehler (IJF, 2006).

Measures of forecast accuracy

```
{r beer-test-accuracy, dependson='beer-fc-1'} beer_fc <- forecast(beer_fit, h =
"3 years") accuracy(beer_fc, aus_production)
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Lab Session 13

- Create a training set for Australian student enrolments by withholding the last four years as a test set.
- Fit all the appropriate benchmark methods to the training set and forecast the periods covered by the test set.
- Compute the accuracy of your forecasts. Which method does best?
- Repeat the exercise using the Australian takeaway food turnover data (`aus_retail`) with a test set of four years.

Outline

Outline

Pharmaceutical Benefits Scheme

Pharmaceutical Benefits Scheme

The Pharmaceutical Benefits Scheme (PBS) is the Australian government drugs subsidy scheme.

- Many drugs bought from pharmacies are subsidised to allow more equitable access to modern drugs.
- The cost to government is determined by the number and types of drugs purchased. Currently nearly 1% of GDP.
- The total cost is budgeted based on forecasts of drug usage.

Pharmaceutical Benefits Scheme

ABC News Online

AUSTRALIAN BROADCASTING CORPORATION

NewsRadio

Streaming audio news

LISTEN: WMP | Real

Select a Topic from the list below

Click "Refresh" or "Reload" on your browser for the latest edition.

This Bulletin: **Wed, May 30 2001 6:22 PM AEST**

- Top Stories
- Just In
- World
- Asia-Pacific
- Business
- Sport
- Arts
- Sci Tech
- Indigenous
- Weather
- Rural
- Local News
- Broadband

Search

SPECIALS

- Federal Election

POLITICS

Opp demands drug price restriction after PBS budget blow-out

The Federal Opposition has called for tighter controls on drug prices after the Pharmaceutical Benefits Scheme (PBS) budget blew out by almost $800 million.

The money was spent on two new drugs including the controversial anti-smoking aid Zyban, which dropped in price from $220 to $22 after it was listed on the PBS.

the Public Record

For full election coverage

FEATURES

the Public Record

Federal Election 2001

For a fresh perspective on the federal election, reach into ABC Online's campaign weblog, The Poll Vault.

Audio News Online

Pharmaceutical Benefits Scheme

- In 2001: $4.5 billion budget, under-forecasted by $800 million.
- Thousands of products. Seasonal demand.
- Subject to covert marketing, volatile products, uncontrollable expenditure.
- Although monthly data available for 10 years, data are aggregated to annual values, and only the first three years are used in estimating the forecasts.
- All forecasts being done with the FORECAST function in MS-Excel!

Historical perspective

- Developed in the 1950s and 1960s as methods (algorithms) to produce point forecasts.
- Combine a “level”, “trend” (slope) and “seasonal” component to describe a time series.
- The rate of change of the components are controlled by “smoothing parameters”: α, β and γ respectively.
- Need to choose best values for the smoothing parameters (and initial states).
- Equivalent ETS state space models developed in the 1990s and 2000s.

A model for levels, trends, and seasonalities

We want a model that captures the level (ℓ_t), trend (b_t) and seasonality (s_t).

How do we combine these elements?

A model for levels, trends, and seasonalities

We want a model that captures the level (ℓ_t), trend (b_t) and seasonality (s_t).

How do we combine these elements?

Additively?

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$$

A model for levels, trends, and seasonalities

We want a model that captures the level (ℓ_t), trend (b_t) and seasonality (s_t).

How do we combine these elements?

Additively?

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$$

Multiplicatively?

$$y_t = \ell_{t-1} b_{t-1} s_{t-m} (1 + \varepsilon_t)$$

A model for levels, trends, and seasonalities

We want a model that captures the level (ℓ_t), trend (b_t) and seasonality (s_t).

How do we combine these elements?

Additively?

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$$

Multiplicatively?

$$y_t = \ell_{t-1}b_{t-1}s_{t-m}(1 + \varepsilon_t)$$

Perhaps a mix of both?

$$y_t = (\ell_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t$$

A model for levels, trends, and seasonalities

We want a model that captures the level (ℓ_t), trend (b_t) and seasonality (s_t).

How do we combine these elements?

Additively?

$$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$$

Multiplicatively?

$$y_t = \ell_{t-1} b_{t-1} s_{t-m}(1 + \varepsilon_t)$$

Perhaps a mix of both?

$$y_t = (\ell_{t-1} + b_{t-1}) s_{t-m} + \varepsilon_t$$

How do the level, trend and seasonal components evolve over time?

ETS models

General notation E T S : **E**xponen**T**ial **S**moothing

Error **T**rend **S**eason

Error: Additive (`"A"`) or multiplicative (`"M"`)

ETS models

General notation   E T S : **E**xponen**T**ial **S**moothing

Error **T**rend **S**eason

Error: Additive (`"A"`) or multiplicative (`"M"`)

Trend: None (`"N"`), additive (`"A"`), multiplicative (`"M"`), or damped (`"Ad"` or `"Md"`).

ETS models

General notation E T S : **E**xponen**T**ial **S**moothing

Error **T**rend **S**eason

Error: Additive (`"A"`) or multiplicative (`"M"`)

Trend: None (`"N"`), additive (`"A"`), multiplicative (`"M"`), or damped (`"Ad"` or `"Md"`).

Seasonality: None (`"N"`), additive (`"A"`) or multiplicative (`"M"`)

ETS(A,N,N): SES with additive errors

Forecast equation	$\hat{y}_{T+h\|T} = \ell_T$
Measurement equation	$y_t = \ell_{t-1} + \varepsilon_t$
State equation	$\ell_t = \ell_{t-1} + \alpha\varepsilon_t$

where $\varepsilon_t \sim \text{NID}(0, \sigma^2)$.

ETS(A,N,N): SES with additive errors

Forecast equation	$\hat{y}_{T+h\|T} = \ell_T$
Measurement equation	$y_t = \ell_{t-1} + \varepsilon_t$
State equation	$\ell_t = \ell_{t-1} + \alpha\varepsilon_t$

where $\varepsilon_t \sim \text{NID}(0, \sigma^2)$.

- "innovations" or "single source of error" because equations have the same error process, ε_t.
- Measurement equation: relationship between observations and states.
- Transition/state equation(s): evolution of state(s) over time.

ETS(M,N,N): SES with multiplicative errors

$$
\begin{aligned}
\text{Forecast equation} \qquad \hat{y}_{T+h|T} &= \ell_T \\
\text{Measurement equation} \qquad y_t &= \ell_{t-1}(1 + \varepsilon_t) \\
\text{State equation} \qquad \ell_t &= \ell_{t-1}(1 + \alpha\varepsilon_t)
\end{aligned}
$$

where $\varepsilon_t \sim \text{NID}(0, \sigma^2)$.

ETS(M,N,N): SES with multiplicative errors

Forecast equation	$\hat{y}_{T+h\|T} = \ell_T$
Measurement equation	$y_t = \ell_{t-1}(1 + \varepsilon_t)$
State equation	$\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$

where $\varepsilon_t \sim \text{NID}(0, \sigma^2)$.

- Models with additive and multiplicative errors with the same parameters generate the same point forecasts but different prediction intervals.

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Holt's linear trend

Additive errors: ETS(A,A,N)

$$
\begin{aligned}
\text{Forecast equation} \quad && \hat{y}_{T+h|T} &= \ell_T + hb_T \\
\text{Measurement equation} \quad && y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\
\text{State equations} \quad && \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
&& b_t &= b_{t-1} + \beta\varepsilon_t
\end{aligned}
$$

Holt's linear trend

Additive errors: ETS(A,A,N)

$$
\begin{aligned}
\text{Forecast equation} \qquad \hat{y}_{T+h|T} &= \ell_T + hb_T \\
\text{Measurement equation} \qquad y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\
\text{State equations} \qquad \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t
\end{aligned}
$$

Multiplicative errors: ETS(M,A,N)

$$
\begin{aligned}
\text{Forecast equation} \qquad \hat{y}_{T+h|T} &= \ell_T + hb_T \\
\text{Measurement equation} \qquad y_t &= (\ell_{t-1} + b_{t-1})(1 + \varepsilon_t) \\
\text{State equations} \qquad \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t) \\
b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t
\end{aligned}
$$

Example: Australian population

```
{r holt-fit, echo=TRUE} aus_economy <- global_economy |>   filter(Country ==
"Australia") |>   mutate(Pop = Population / 1e6) fit <- aus_economy |> model(AAN =
ETS(Pop)) report(fit)
```

Example: Australian population

```
{r holt-cmp, echo=TRUE, dependson='holt-fit'} components(fit)
```

Example: Australian population

```
{r holt-cmp-plot, echo=TRUE, dependson='holt-fit', fig.height=4.5}
```

```
components(fit) |> autoplot()
```

Example: Australian population

{r holt-fc, echo=TRUE, dependson='holt-fit', fig.height=3.4} fit |>   forecast(h = 20) |>   autoplot(aus_economy) +   labs(y = "Population", x = "Year")

ETS(A,Ad,N): Damped trend method

Additive errors

$$
\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{T+h|T} = \ell_T + (\phi + \cdots + \phi^{h-1})b_T \\
\text{Measurement equation} \quad & y_t = (\ell_{t-1} + \phi b_{t-1}) + \varepsilon_t \\
\text{State equations} \quad & \ell_t = (\ell_{t-1} + \phi b_{t-1}) + \alpha\varepsilon_t \\
& b_t = \phi b_{t-1} + \beta\varepsilon_t
\end{aligned}
$$

ETS(A,Ad,N): Damped trend method

Additive errors

$$
\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{T+h|T} = \ell_T + (\phi + \cdots + \phi^{h-1})b_T \\
\text{Measurement equation} \quad & y_t = (\ell_{t-1} + \phi b_{t-1}) + \varepsilon_t \\
\text{State equations} \quad & \ell_t = (\ell_{t-1} + \phi b_{t-1}) + \alpha\varepsilon_t \\
& b_t = \phi b_{t-1} + \beta\varepsilon_t
\end{aligned}
$$

- Damping parameter $0 < \phi < 1$.
- If $\phi = 1$, identical to Holt's linear trend.
- As $h \to \infty$, $\hat{y}_{T+h|T} \to \ell_T + \phi b_T/(1-\phi)$.
- Short-run forecasts trended, long-run forecasts constant.

Example: Australian population

```
{r, echo=TRUE, fig.height=3.4} aus_economy |>   model(holt =
ETS(Pop ~ trend("Ad"))) |>   forecast(h = 20) |>
autoplot(aus_economy)
```

Example: National populations

```
{r popfit, echo=TRUE} fit <- global_economy |>   mutate(Pop = Population / 1e6) |>

model(ets = ETS(Pop)) fit
```

Example: National populations

```
{r popfc, echo=TRUE, dependson="popfit"} fit |>   forecast(h = 5)
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Lab Session 14

Try forecasting the Australian GDP from the `global_economy` data set using an ETS model.

Experiment with the various options in the `ETS()` function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to develop an intuition of what each is doing to the forecasts.

[Hint: use `h=20` when forecasting, so you can clearly see the differences between the various options when plotting the forecasts.]

Outline

ETS(A,A,A): Holt-Winters additive method

$$
\begin{aligned}
\text{Forecast equation} \quad \hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\text{Observation equation} \quad y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\
\text{State equations} \quad \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t \\
s_t &= s_{t-m} + \gamma\varepsilon_t
\end{aligned}
$$

- k = integer part of $(h-1)/m$.
- $\sum_i s_i \approx 0$.
- Parameters: $0 \le \alpha \le 1,\ 0 \le \beta^* \le 1,\ 0 \le \gamma \le 1-\alpha$ and m = period of seasonality (e.g. $m = 4$ for quarterly data).

ETS(M,A,M): Holt-Winters multiplicative method

$$\begin{aligned}
\text{Forecast equation} \quad \hat{y}_{t+h|t} &= (\ell_t + hb_t)s_{t+h-m(k+1)} \\
\text{Observation equation} \quad y_t &= (\ell_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t) \\
\text{State equations} \quad \ell_t &= (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t) \\
b_t &= b_{t-1}(1 + \beta\varepsilon_t) \\
s_t &= s_{t-m}(1 + \gamma\varepsilon_t)
\end{aligned}$$

- k is integer part of $(h-1)/m$.
- $\sum_i s_i \approx m$.
- Parameters: $0 \le \alpha \le 1$, $0 \le \beta^* \le 1$, $0 \le \gamma \le 1-\alpha$ and m = period of seasonality (e.g. $m = 4$ for quarterly data).

Example: Australian holiday tourism

```
{r ausholidays-fit, echo=TRUE} holidays <- tourism |>   filter(Purpose == "Holiday")

fit <- holidays |> model(ets = ETS(Trips)) fit
```

Example: Australian holiday tourism

```
fit |>
  filter(Region == "Snowy Mountains") |>
  report()
```

Example: Australian holiday tourism

```
fit |>
  filter(Region == "Snowy Mountains") |>
  components(fit)
```

Example: Australian holiday tourism

```
#| fig-height: 3.6
fit |>
  filter(Region == "Snowy Mountains") |>
  components(fit) |>
  autoplot()
```

Example: Australian holiday tourism

```
fit |> forecast()
```

Example: Australian holiday tourism

```
{r ausholidays-forecast-plot, fig.height=3.4} fit |>   forecast() |>   filter(Region
== "Snowy Mountains") |>   autoplot(holidays) +   labs(x = "Year", y = "Overnight
trips (thousands)")
```

Outline

Exponential smoothing models

Additive Error

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	A,N,N	A,N,A	~~A,N,M~~
A	(Additive)	A,A,N	A,A,A	~~A,A,M~~
A_d	(Additive damped)	A,A_d,N	A,A_d,A	~~A,A_d,M~~

Multiplicative Error

Trend Component		Seasonal Component N (None)	A (Additive)	M (Multiplicative)
N	(None)	M,N,N	M,N,A	M,N,M
A	(Additive)	M,A,N	M,A,A	M,A,M
A_d	(Additive damped)	M,A_d,N	M,A_d,A	M,A_d,M

Estimating ETS models

- Smoothing parameters α, β, γ and ϕ, and the initial states ℓ_0, b_0, $s_0, s_{-1}, \dots, s_{-m+1}$ are estimated by maximising the "likelihood" = the probability of the data arising from the specified model.
- For models with additive errors equivalent to minimising SSE.
- For models with multiplicative errors, **not** equivalent to minimising SSE.

Model selection

Akaike's Information Criterion

$$\text{AIC} = -2\log(\text{L}) + 2k$$

where L is the likelihood and k is the number of parameters & initial states estimated in the model.

Model selection

Akaike's Information Criterion

$$\text{AIC} = -2\log(\text{L}) + 2k$$

where L is the likelihood and k is the number of parameters & initial states estimated in the model.

Corrected AIC

$$\text{AIC}_\text{c} = \text{AIC} + \frac{2k(k+1)}{T-k-1}$$

which is the AIC corrected (for small sample bias).

Model selection

Akaike's Information Criterion

$$\text{AIC} = -2\log(\text{L}) + 2k$$

where L is the likelihood and k is the number of parameters & initial states estimated in the model.

Corrected AIC

$$\text{AIC}_{\text{c}} = \text{AIC} + \frac{2k(k+1)}{T-k-1}$$

which is the AIC corrected (for small sample bias).

Bayesian Information Criterion

$$\text{BIC} = \text{AIC} + k(\log(T) - 2).$$

AIC and cross-validation

Minimizing the AIC assuming Gaussian residuals is asymptotically equivalent to minimizing one-step time series cross validation MSE.

Automatic forecasting

From Hyndman et al. (IJF, 2002):

1. Apply each model that is appropriate to the data. Optimize parameters and initial values using MLE.
2. Select best method using AICc.
3. Produce forecasts using best method.
4. Obtain forecast intervals using underlying state space model.

- Method performed very well in M3 competition.
- Used as a benchmark in the M4 competition.

Outline

Lab Session 15

Find an ETS model for the Gas data from `aus_production`.

- Why is multiplicative seasonality necessary here?
- Experiment with making the trend damped. Does it improve the forecasts?

Outline

Non-Gaussian forecast distributions

```
{r cafe, fig.height=3.1} vic_cafe <- tsibbledata::aus_retail |>   filter(State ==
"Victoria",          Industry == "Cafes, restaurants and catering services") |>
select(Month, Turnover) vic_cafe |>   autoplot(Turnover) + labs(title = "Monthly
turnover of Victorian cafes")
```

Forecasting with transformations

```
vic_cafe |> autoplot(box_cox(Turnover, lambda = 0.2))
```

Forecasting with transformations

```r
fit <- vic_cafe |>
  model(ets = ETS(box_cox(Turnover, 0.2)))
fit
```

{r include=FALSE} if (!identical(fabletools::model_sum(fit$ets[[1]]), "ETS(A,A,A)")) { stop("Model not ETS(A,A,A)") }

```r
(fc <- fit |> forecast(h = "3 years"))
```

Forecasting with transformations

```r
fit <- vic_cafe |>
  model(ets = ETS(box_cox(Turnover, 0.2)))
fit
```

{r include=FALSE} if (!identical(fabletools::model_sum(fit$ets[[1]]), "ETS(A,A,A)")) { stop("Model not ETS(A,A,A)") }

```r
(fc <- fit |> forecast(h = "3 years"))
```

- `t(N)` denotes a transformed normal distribution.
- back-transformation and bias adjustment is done automatically.

Forecasting with transformations

```
fc |> autoplot(vic_cafe)
```

Bootstrapped forecast distributions

```
sim <- fit |> generate(h = "3 years", times = 5, bootstrap = TRUE)
sim
```

Bootstrapped forecast distributions

```
{r, fig.height=2.7} vic_cafe |>   filter(year(Month) >= 2008) |>   ggplot(aes(x =
Month)) +   geom_line(aes(y = Turnover)) +   geom_line(aes(y = .sim, colour =
as.factor(.rep)), data = sim) +   labs(title = "Monthly turnover of Victorian cafes") +
guides(col = FALSE)
```

Bootstrapped forecast distributions

```
fc <- fit |> forecast(h = "3 years", bootstrap = TRUE)
fc
```

Bootstrapped forecast distributions

```
fc |> autoplot(vic_cafe) +
  labs(title = "Monthly turnover of Victorian cafes")
```

Outline

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

ARIMA models

AR: autoregressive (lagged observations as inputs)
I: integrated (differencing to make series stationary)
MA: moving average (lagged errors as inputs)

ARIMA models

AR: autoregressive (lagged observations as inputs)

I: integrated (differencing to make series stationary)

MA: moving average (lagged errors as inputs)

An ARIMA model is rarely interpretable in terms of visible data structures like trend and seasonality. But it can capture a huge range of time series patterns.

Stationarity

Definition

If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \dots, y_{t+s})$ does not depend on t.

Stationarity

Definition

If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \dots, y_{t+s})$ does not depend on t.

A **stationary series** is:

- roughly horizontal
- constant variance
- no patterns predictable in the long-term

Stationary?

```
{r, fig.height=2.7} gafa_stock |>   filter(Symbol == "GOOG", year(Date)
== 2018) |>   autoplot(Close) +   labs(y = "Google closing stock price
($US)")
```

Stationary?

```
{r, fig.height=2.7} gafa_stock |>   filter(Symbol == "GOOG", year(Date)
== 2018) |>   autoplot(difference(Close)) +   labs(y = "Daily change in
Google closing stock price")
```

Differencing

- Differencing helps to **stabilize the mean**.
- The differenced series is the *change* between each observation in the original series.
- Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time.
- In practice, it is almost never necessary to go beyond second-order differences.

Autoregressive models

Autoregressive (AR) models:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

where ε_t is white noise. A multiple regression with **lagged values** of y_t as predictors.

```
{r arp, echo=FALSE, fig.height=2.3} set.seed(1) p1 <-
tsibble(idx = seq_len(100), sim = 10 +
arima.sim(list(ar = -0.8), n = 100), index = idx) |>
autoplot(sim) + labs(x = "time", y = "", title =
"AR(1)") p2 <- tsibble(idx = seq_len(100), sim = 20 +
arima.sim(list(ar = c(1.3, -0.7)), n = 100), index =
```

Moving Average (MA) models

Moving Average (MA) models:

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q},$$

where ε_t is white noise. A multiple regression with **lagged *errors*** as predictors. *Don't confuse with moving average smoothing!*

"'{r maq, fig.height=2.3, echo=FALSE} set.seed(2) p1 <- tsibble(idx = seq_len(100), sim = 20 + arima.sim(list(ma = 0.8), n = 100), index = idx) |> autoplot(sim) + labs(x = "time", y = " ", title ="MA(1)") p2 <- tsibble(idx = seq_len(100), sim = arima.sim(list(ma = c(-1, +0.8)), n = 100), index = idx) |>

Example: National populations

```
{r popfit3, echo=TRUE} fit |>   filter(Country == "Australia") |>
report()
```

Example: National populations

```
{r popfit3, echo=TRUE} fit |>   filter(Country == "Australia") |>
report()
```

$$
\begin{aligned}
y_t &= 2y_{t-1} - y_{t-2} - 0.7\varepsilon_{t-1} + \varepsilon_t \\
\varepsilon_t &\sim \text{NID}(0, 4 \times 10^9)
\end{aligned}
$$

Understanding ARIMA models

- If $c = 0$ and $d = 0$, the long-term forecasts will go to zero.
- If $c = 0$ and $d = 1$, the long-term forecasts will go to a non-zero constant.
- If $c = 0$ and $d = 2$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 0$, the long-term forecasts will go to the mean of the data.
- If $c \neq 0$ and $d = 1$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 2$, the long-term forecasts will follow a quadratic trend.

Understanding ARIMA models

Forecast variance and d

- The higher the value of d, the more rapidly the prediction intervals increase in size.
- For $d = 0$, the long-term forecast standard deviation will go to the standard deviation of the historical data.

Example: National populations

```
{r popfc2, echo=TRUE, fig.height=3.4} fit |>   forecast(h = 10) |>   filter(Country ==
"Australia") |>   autoplot(global_economy)
```

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

$$\text{AICc} = -2\log(L) + 2(p+q+k+1)\left[1 + \frac{(p+q+k+2)}{T-p-q-k-2}\right]$$

where L is the maximised likelihood fitted to the *differenced* data, $k=1$ if $c \neq 0$ and $k=0$ otherwise.

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

$$\text{AICc} = -2\log(L) + 2(p+q+k+1)\left[1 + \frac{(p+q+k+2)}{T-p-q-k-2}\right]$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Note: Can't compare AICc for different values of d.

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:

ARIMA($2, d, 2$)
ARIMA($0, d, 0$)
ARIMA($1, d, 0$)
ARIMA($0, d, 1$)

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:

ARIMA$(2, d, 2)$
ARIMA$(0, d, 0)$
ARIMA$(1, d, 0)$
ARIMA$(0, d, 1)$

Step 2: Consider variations of current model:

- vary one of p, q, from current model by ± 1;
- p, q both vary from current model by ± 1;
- Include/exclude c from current model.

Model with lowest AICc becomes current model.

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:
ARIMA$(2, d, 2)$
ARIMA$(0, d, 0)$
ARIMA$(1, d, 0)$
ARIMA$(0, d, 1)$

Step 2: Consider variations of current model:

- vary one of $p, q,$ from current model by ± 1;
- p, q both vary from current model by ± 1;
- Include/exclude c from current model.

Model with lowest AICc becomes current model.

Repeat Step 2 until no lower AICc can be found.

How does ARIMA() work?

```
{r ARMAgridsearch, echo=FALSE, message=FALSE,
warning=FALSE, fig.asp=1, out.width="60%",
fig.width=4, fig.height=4} start <- tribble(   ~p, ~q,
0, 0,   1, 0,   0, 1,   2, 2 ) selected <- tribble(
~p, ~q,   2, 2,   3, 3,   4, 2 ) griddf <-
expand.grid(p = 0:6, q = 0:6) |>   as_tibble() |>
left_join(start |> mutate(start = TRUE)) |>
left_join(selected |> mutate(chosen = TRUE)) |>
replace_na(list(start = FALSE, chosen = FALSE)) |>
mutate(     step = case_when(       start ~ 1,
(p - selected$p[1])^2 + (q - selected$q[1])^2 <= 2 ~
```

How does ARIMA() work?

```
{r ARMAgridsearch2, echo=FALSE, message=FALSE,
warning=FALSE, fig.asp=1, out.width="60%",
fig.width=4, fig.height=4} griddf |>   ggplot(aes(x =
q, y = p)) +   geom_point(aes(alpha = 0.2), colour =
"gray", size = 5, shape = 19) +   geom_segment(aes(x =
fromq, y = fromp, xend = q, yend = p, col = step),
data = griddf |> filter(step == "2"),     arrow =
arrow(length = unit(0.15, "inches"), type = "open"),
size = 1, lineend = "butt"    ) +   geom_point(aes(col
= step),      size = 5, shape = 19,     data = griddf
|> filter(step %in% c("1", "2"))    ) +
```

How does ARIMA() work?

```
{r ARMAgridsearch3, echo=FALSE, message=FALSE,
warning=FALSE, fig.asp=1, out.width="60%",
fig.width=4, fig.height=4} griddf |>   ggplot(aes(x =
q, y = p)) +   geom_point(aes(alpha = 0.2), colour =
"gray", size = 5, shape = 19) +   geom_segment(aes(x =
fromq, y = fromp, xend = q, yend = p, col = step),
data = griddf |> filter(step %in% "3"),     arrow =
arrow(length = unit(0.15, "inches"), type = "open"),
size = 1, lineend = "butt"    ) +   geom_point(aes(col
= step),     size = 5, shape = 19,     data = griddf
|> filter(step %in% c("1", "2", "3"))   ) +
```

How does ARIMA() work?

```
{r ARMAgridsearch4, echo=FALSE, message=FALSE,
warning=FALSE, fig.asp=1, out.width="60%",
fig.width=4, fig.height=4} griddf |>   ggplot(aes(x =
q, y = p)) +   geom_point(aes(alpha = 0.2), colour =
"gray", size = 5, shape = 19) +   geom_segment(aes(x =
fromq, y = fromp, xend = q, yend = p, col = step),
data = griddf |> filter(step %in% "4"),     arrow =
arrow(length = unit(0.15, "inches"), type = "open"),
size = 1, lineend = "butt"   ) +   geom_point(aes(col
= step),     size = 5, shape = 19,     data = griddf
|> filter(step %in% c("1", "2", "3", "4"))   ) +
```

Outline

Lab Session 16

For the Australian job vacancies:

```
{r, eval = FALSE} skill_vacancies <-
readxl::read_excel("data/Internet Vacancies, ANZSCO
Skill Level, States and Territories - May 2024.xlsx",
sheet = 2) |>    # Tidy into a long form
pivot_longer(matches("\\d{5}"), names_to = "month",
values_to = "vacancies",
names_transform = list(month = ~
yearmonth(as.Date(as.integer(.), origin =
"1900-01-01")))) |>    # Remove aggregate vacancies
filter(State != "AUST", Skill level != 0) |>    #
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Seasonal ARIMA models

ARIMA	$\underbrace{(p, d, q)}$	$\underbrace{(P, D, Q)_m}$
	$\uparrow$ Non-seasonal part of the model	$\uparrow$ Seasonal part of of the model

- m = number of observations per year.
- d first differences, D seasonal differences
- p AR lags, q MA lags
- P seasonal AR lags, Q seasonal MA lags

Seasonal and non-seasonal terms combine multiplicatively

Cortecosteroid drug sales

```
{r, echo=TRUE} h02 <- PBS |>   filter(ATC2 == "H02")
|>   summarise(Cost = sum(Cost) / 1e6)
```

Cortecosteroid drug sales

```
{r, echo=TRUE} h02 |> autoplot(   Cost )
```

Cortecosteroid drug sales

```
{r, echo=TRUE} h02 |> autoplot(   log(Cost) )
```

Cortecosteroid drug sales

```
{r, echo=TRUE} h02 |> autoplot(   log(Cost) |>
difference(12) )
```

Cortecosteroid drug sales

```
{r, echo=TRUE} h02 |> autoplot(   log(Cost) |>
difference(12) |> difference(1) )
```

Example: US electricity production

```
{r h02fit, echo=TRUE, fig.height=2.8} h02 |>   model(arima =
ARIMA(log(Cost))) |>   report()
```

Example: US electricity production

```
{r h02fcst, echo=TRUE, fig.height=2.8} h02 |>   model(arima =
ARIMA(log(Cost))) |>   forecast(h = "3 years") |>   autoplot(h02)
```

Cortecosteroid drug sales

```
{r h02tryharder, echo=TRUE, fig.height=3.6} fit <- h02 |>   model(best =

ARIMA(log(Cost),     stepwise = FALSE,     approximation = FALSE,     order_constraint

= p + q + P + Q <= 9   )) report(fit)
```

Cortecosteroid drug sales

```
{r h02f, echo=TRUE, fig.height=2.8} fit |>   forecast() |>
autoplot(h02) +   labs(y = "H02 Expenditure ($AUD)", x = "Year")
```

Outline

Lab Session 17

For the Australian tourism data (from `tourism`):

- Fit a suitable ARIMA model for all data.
- Produce forecasts of your fitted models.
- Check the forecasts for the "Snowy Mountains" and "Melbourne" regions. Do they look reasonable?

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Forecast ensembles

```
{r trainall, echo=TRUE, dependson='tourism'} train <- tourism |>
filter(year(Quarter) <= 2014) fit <- train |>   model(     ets = ETS(Trips),
arima = ARIMA(Trips),      snaive = SNAIVE(Trips)    ) |>   mutate(mixed = (ets
+ arima + snaive) / 3)
```

- Ensemble forecast `mixed` is a simple average of the three fitted models.
- `forecast()` will produce distributional forecasts taking into account the correlations between the forecast errors of the component models.

Forecast ensembles

```
{r trainfc, dependson='trainall'} #| fig-height: 3.6 fc <- fit |> forecast(h =
"3 years") fc |>   filter(Region == "Snowy Mountains", Purpose == "Holiday")
|>   autoplot(tourism, level = NULL)
```

Forecast ensembles

```
{r snowy-test-accuracy, dependson='trainfc'} accuracy(fc, tourism) |>
group_by(.model) |>   summarise(     RMSE = mean(RMSE),     MAE = mean(MAE),
MASE = mean(MASE)   ) |>   arrange(RMSE)
```

Forecast ensembles

Can we do better than equal weights?

Forecast ensembles

Can we do better than equal weights?

- Hard to find weights that improve forecast accuracy.
- Known as the "forecast combination puzzle".
- Solution: FFORMA

Forecast ensembles

Can we do better than equal weights?

- Hard to find weights that improve forecast accuracy.
- Known as the "forecast combination puzzle".
- Solution: FFORMA

FFORMA (Feature-based FORecast Model Averaging)

- Vector of time series features used to predict best weights.
- A modification of xgboost is used.
- Method came 2nd in the 2018 M4 international forecasting competition.
- Main author: Pablo Montero-Manso (now Uni Sydney)
- Not (yet) available for fable.

Outline

Outline

Notice: Material planned to change

This material is planned to be updated to better align with the training needs of the Department of Education.

In particular, the new material will be more focused on:

- the use of policy in models,
- forecasting with different scenarios.

Outline

Regression with ARIMA errors

Regression models

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t,$$

- y_t modeled as function of k explanatory variables
- In regression, we assume that ε_t is white noise.

Regression with ARIMA errors

Regression models

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t,$$

- y_t modeled as function of k explanatory variables
- In regression, we assume that ε_t is white noise.

RegARIMA model

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t,$$
$$\eta_t \sim \text{ARIMA}$$

- Residuals are from ARIMA model.
- Estimate model in one step using MLE
- Select model with lowest AICc value.

US personal consumption and income

```
{r usconsumpdata, echo=TRUE} us_change
```

US personal consumption and income

```
{r usconsump, fig.height=4, fig.width=6,
out.height="55%"} us_change |>
pivot_longer(-Quarter, names_to = "variable",
values_to = "value") |>   ggplot(aes(y = value, x =
Quarter, group = variable)) +   geom_line() +
facet_grid(variable ~ ., scales = "free_y") +   labs(x
= "Year", y = "",         title = "Quarterly changes in
US consumption and personal income")
```

US personal consumption and income

```
{r usconsump_pairs, fig.height=6, fig.width=8.5, echo=TRUE} #| out-width: 70%

us_change |> as_tibble() |> select(-Quarter) |> GGally::ggpairs()
```

US personal consumption and income

- No need for transformations or further differencing.
- Increase in income does not necessarily translate into instant increase in consumption (e.g., after the loss of a job, it may take a few months for expenses to be reduced to allow for the new circumstances). We will ignore this for now.

US personal consumption and income

```
{r usconsump2, echo=TRUE, fig.height=3} fit <- us_change |>   model(regarima =
ARIMA(Consumption ~ Income + Production + Savings +
Unemployment)) report(fit)
```

US personal consumption and income

```
{r usconsump2, echo=TRUE, fig.height=3} fit <- us_change |>   model(regarima =
ARIMA(Consumption ~ Income + Production + Savings +
Unemployment)) report(fit)
```

Write down the equations for the fitted model.

US personal consumption and income

```
{r , echo=TRUE, fig.height=3.7,
dependson='usconsump2'} gg_tsresiduals(fit)
```

US personal consumption and income

```
{r , echo=TRUE, fig.height=3.7, dependson='usconsump2'} augment(fit) |>
features(.resid, ljung_box, dof = 2, lag = 12)
```

US personal consumption and income

```
{r usconsump3, echo=TRUE, fig.height=2.4,
dependson='usconsump2'} us_change_future <-
new_data(us_change, 8) |>   mutate(Income =
tail(us_change$Income, 1),          Production =
tail(us_change$Production, 1),          Savings =
tail(us_change$Savings, 1),          Unemployment =
tail(us_change$Unemployment, 1)) forecast(fit,
new_data = us_change_future) |>   autoplot(us_change)
+   labs(x = "Year", y = "Percentage change",
title = "Forecasts from dynamic regression")
```

Forecasting

- To forecast a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model and combine the results.
- Some predictors are known into the future (e.g., time, dummies).
- Separate forecasting models may be needed for other predictors.
- Forecast intervals ignore the uncertainty in forecasting the predictors.

Daily electricity demand

Model daily electricity demand as a function of temperature using quadratic regression with ARMA errors.

```
{r, echo=TRUE, fig.height=2.7} vic_elec_daily |>   ggplot(aes(x = Temperature, y = Demand, colour
= Day_Type)) +   geom_point() +   labs(x = "Maximum temperature", y = "Electricity demand (GW)")
```

Daily electricity demand

```
{r, echo=TRUE, fig.height=3.2} vic_elec_daily |>   pivot_longer(c(Demand, Temperature)) |>

ggplot(aes(x = Date, y = value)) +   geom_line() +   facet_grid(vars(name), scales = "free_y")
```

Daily electricity demand

{r dailymodel, echo=TRUE} fit <- vic_elec_daily |>   model(fit = ARIMA(Demand ~ Temperature +

I(Temperatureˆ2) +   (Day_Type == "Weekday"))) report(fit)

Daily electricity demand

```
{r, echo=TRUE, dependson='dailymodel'} augment(fit) |>
gg_tsdisplay(.resid, plot_type = "histogram")
```

Daily electricity demand

```
{r, echo=TRUE, dependson='dailymodel'} augment(fit) |>
features(.resid, ljung_box, dof = 9, lag = 14)
```

Daily electricity demand

```
{r, echo=TRUE, dependson='dailymodel'} # Forecast one day ahead

vic_next_day <- new_data(vic_elec_daily, 1) |>   mutate(Temperature =

26, Day_Type = "Holiday") forecast(fit, vic_next_day)
```

Daily electricity demand

```
{r, echo=TRUE} vic_elec_future <-
new_data(vic_elec_daily, 14) |>   mutate(
Temperature = 26,     Holiday = c(TRUE, rep(FALSE,
13)),     Day_Type = case_when(       Holiday ~
"Holiday",        wday(Date) %in% 2:6 ~ "Weekday",
TRUE ~ "Weekend"     )   )
```

Daily electricity demand

```
{r, echo = TRUE, dependson='dailymodel'} forecast(fit, vic_elec_future) |>

autoplot(vic_elec_daily) + labs(y = "Electricity demand (GW)")
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Lab Session 18

Repeat the daily electricity example, but instead of using a quadratic function of temperature, use a piecewise linear function with the “knot” around 20 degrees Celsius (use predictors `Temperature` & `Temp2`). How can you optimize the choice of knot?

```
{r echo=TRUE, eval=FALSE} vic_elec_daily <- vic_elec |>   filter(year(Time) == 2014) |>

index_by(Date = date(Time)) |>   summarise(Demand = sum(Demand) / 1e3,             Temperature =

max(Temperature),             Holiday = any(Holiday)   ) |>   mutate(Temp2 = I(pmax(Temperature -

20, 0)),         Day_Type = case_when(            Holiday ~ "Holiday",            wday(Date)

%in% 2:6 ~ "Weekday",            TRUE ~ "Weekend")   )
```

Outline

Dynamic harmonic regression

Combine Fourier terms with ARIMA errors

Advantages

- it allows any length seasonality;
- for data with more than one seasonal period, you can include Fourier terms of different frequencies;
- the seasonal pattern is smooth for small values of K (but more wiggly seasonality can be handled by increasing K);
- the short-term dynamics are easily handled with a simple ARMA error.

Disadvantages

- seasonality is assumed to be fixed

Eating-out expenditure

```
{r cafe, echo=TRUE, fig.height=2.3, fig.width=8} aus_cafe <- aus_retail |>   filter(     Industry
== "Cafes, restaurants and takeaway food services",     year(Month) %in% 2004:2018   ) |>
summarise(Turnover = sum(Turnover)) aus_cafe |> autoplot(Turnover)
```

Eating-out expenditure

```
{r cafefit, dependson='cafe', fig.height=5, echo=TRUE, results='hide'} fit <- aus_cafe
|> model(   `K = 1` = ARIMA(log(Turnover) ~ fourier(K = 1) + PDQ(0, 0, 0)),   `K = 2`
= ARIMA(log(Turnover) ~ fourier(K = 2) + PDQ(0, 0, 0)),   `K = 3` =
ARIMA(log(Turnover) ~ fourier(K = 3) + PDQ(0, 0, 0)),   `K = 4` = ARIMA(log(Turnover)
~ fourier(K = 4) + PDQ(0, 0, 0)),   `K = 5` = ARIMA(log(Turnover) ~ fourier(K = 5) +
PDQ(0, 0, 0)),   `K = 6` = ARIMA(log(Turnover) ~ fourier(K = 6) + PDQ(0, 0, 0)) )
glance(fit) {r, echo = FALSE} glance(fit) |>   select(.model, sigma2, log_lik, AIC,
AICc, BIC) |>   knitr::kable()
```

Eating-out expenditure

{r, include=FALSE} cafe_plot <- function(...) {   fit |>   select(...) |>   forecast() |>   autoplot(aus_cafe) +   labs(title = sprintf("Log transformed %s, fourier(K = %s)", model_sum(select(fit, ...)[[1]][[1]]), deparse(..1))) +   geom_label(aes(x = yearmonth("2007 Jan"), y = 4250, label = paste0("AICc = ", format(AICc))),   data = glance(select(fit, ...))   ) +   geom_line(aes(y = .fitted), colour = "red", augment(select(fit, ...))) +   ylim(c(1500, 5100)) }

{r cafe1, dependson='cafe', fig.height=5, echo=FALSE}

Eating-out expenditure

```
{r cafe2, dependson='cafe', fig.height=5, echo=FALSE}
cafe_plot("K = 2")
```

Eating-out expenditure

```
{r cafe3, dependson='cafe', fig.height=5, echo=FALSE}
cafe_plot("K = 3")
```

Eating-out expenditure

```
{r cafe4, dependson='cafe', fig.height=5, echo=FALSE}
cafe_plot("K = 4")
```

Eating-out expenditure

```
{r cafe5, dependson='cafe', fig.height=5, echo=FALSE}
cafe_plot("K = 5")
```

Eating-out expenditure

```
{r cafe6, dependson='cafe', fig.height=5, echo=FALSE}
cafe_plot("K = 6")
```

Example: weekly gasoline products

{r, echo = FALSE} options(width = 70)

{r gasmodel, echo=TRUE} fit <- us_gasoline |> model(ARIMA(Barrels ~ fourier(K = 13) + PDQ(0, 0,

0))) report(fit)

Example: weekly gasoline products

```
{r gasf, echo=TRUE, fig.height=3} forecast(fit, h = "3 years") |>

autoplot(us_gasoline)
```

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Lab Session 19

Repeat Lab Session 18 but using all available data, and handling the annual seasonality using Fourier terms.

Outline

Lagged predictors

Sometimes a change in x_t does not affect y_t instantaneously

- y_t = sales, x_t = advertising.
- y_t = stream flow, x_t = rainfall.
- y_t = size of herd, x_t = breeding stock.

Lagged predictors

Sometimes a change in x_t does not affect y_t instantaneously

- y_t = sales, x_t = advertising.
- y_t = stream flow, x_t = rainfall.
- y_t = size of herd, x_t = breeding stock.

- These are dynamic systems with input (x_t) and output (y_t).
- x_t is often a leading indicator.
- There can be multiple predictors.

Lagged predictors

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \dots.$

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \dots + \nu_k x_{t-k} + \eta_t$$

where η_t is an ARIMA process.

Lagged predictors

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \dots$.

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \dots + \nu_k x_{t-k} + \eta_t$$

where η_t is an ARIMA process.

- x can influence y, but y is not allowed to influence x.

Example: Insurance quotes and TV adverts

```
insurance
```

Example: Insurance quotes and TV adverts

```
{r tvadvert, dependson='tvadvertdata', echo=FALSE}
insurance |>   pivot_longer(c(Quotes, TVadverts)) |>
ggplot(aes(x = Month, y = value)) +   geom_line() +
facet_grid(vars(name), scales = "free_y") +   labs(x =
"Year", y = NULL, title = "Insurance advertising and
quotations")
```

Example: Insurance quotes and TV adverts

```
{r tvadvertpairs, dependson='tvadvertdata',
echo=FALSE} insurance |>   mutate(      lag1 =
lag(TVadverts),      lag2 = lag(lag1)    ) |>
as_tibble() |>   select(-Month) |>   rename(lag0 =
TVadverts) |>   pivot_longer(-Quotes, names_to =
"Lag", values_to = "TV_advert") |>   ggplot(aes(x =
TV_advert, y = Quotes)) +   geom_point() +
facet_grid(. ~ Lag) +   labs(title = "Insurance
advertising and quotations")
```

Example: Insurance quotes and TV adverts

```
{r, echo=TRUE} fit <- insurance |>   # Restrict data so models use same
fitting period   mutate(Quotes = c(NA, NA, NA, Quotes[4:40])) |>   model(
ARIMA(Quotes ~ pdq(d = 0) + TVadverts),     ARIMA(Quotes ~ pdq(d = 0) +
TVadverts +       lag(TVadverts)),     ARIMA(Quotes ~ pdq(d = 0) + TVadverts +
lag(TVadverts) +       lag(TVadverts, 2)),     ARIMA(Quotes ~ pdq(d = 0) +
TVadverts +       lag(TVadverts) +       lag(TVadverts, 2) +
lag(TVadverts, 3))   )
```

Example: Insurance quotes and TV adverts

```
{r, echo=TRUE, results = 'hide'} glance(fit) {r, echo = FALSE} glance(fit) |>

transmute(`Lag order` = 0:3, sigma2, log_lik, AIC, AICc, BIC) |>

knitr::kable()
```

Example: Insurance quotes and TV adverts

```
{r tvadvertagain, echo=TRUE} # Re-fit to all data fit <- insurance |>   model(ARIMA(Quotes ~ TVadverts + lag(TVadverts) + pdq(d = 0))) report(fit)
```

Example: Insurance quotes and TV adverts

{r tvadvertagain, echo=TRUE} # Re-fit to all data fit <- insurance |>   model(ARIMA(Quotes ~ TVadverts + lag(TVadverts) + pdq(d = 0))) report(fit)

{r tvadvertparam, echo=FALSE, dependson='tvadvertagain'} # Store coefficients coef <- rlang::set_names(tidy(fit)$estimate, tidy(fit)$term) phi1 <- coef["ar1"] theta1 <- coef["ma1"] theta2 <- coef["ma2"] intercept <- coef["intercept"] gamma0 <- coef["TVadverts"] gamma1 <- coef["lag(TVadverts)"]

$$y_t = `rformat(intercept, digits = 3)` + `rformat(gamma0, digits = 3)`x_t + `rfor$$

$$\eta_t = `rformat(phi1, digits = 3)`\eta_{t-1} + \varepsilon_t + `rformat(theta1, digits = 2)`\varepsilon_{t-1} + `r$$

Example: Insurance quotes and TV adverts

```
{r, echo=TRUE, fig.height=3} advert_a <- new_data(insurance, 20) |>
mutate(TVadverts = 10) forecast(fit, advert_a) |> autoplot(insurance)
```

Example: Insurance quotes and TV adverts

```
{r, echo=TRUE, fig.height=3} advert_b <- new_data(insurance, 20) |>
mutate(TVadverts = 8) forecast(fit, advert_b) |> autoplot(insurance)
```

Example: Insurance quotes and TV adverts

{r, echo=TRUE, fig.height=3} advert_c <- new_data(insurance, 20) |>

mutate(TVadverts = 6) forecast(fit, advert_c) |> autoplot(insurance)

Outline

Outline

Notice: Material planned to change

This material is planned to be updated to better align with the training needs of the Department of Education.

In particular, this section will be reduced or removed to make time for multivariate econometric models.

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

Australian Pharmaceutical Benefits Scheme

PBS sales

```
{r setwidth, include=FALSE} fred <- options(width = 71)
```

```
{r pbs, dependson='setwidth'} PBS
```

```
{r setwidthback, include=FALSE} options(width = fred$width)
```

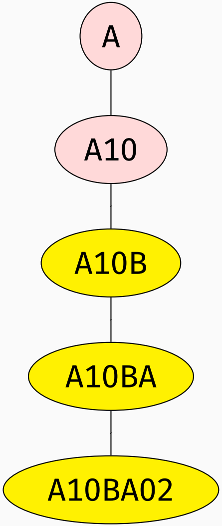
ATC drug classification

A	Alimentary tract and metabolism
B	Blood and blood forming organs
C	Cardiovascular system
D	Dermatologicals
G	Genito-urinary system and sex hormones
H	Systemic hormonal preparations, excluding sex hormones and insulins
J	Anti-infectives for systemic use
L	Antineoplastic and immunomodulating agents
M	Musculo-skeletal system
N	Nervous system
P	Antiparasitic products, insecticides and repellents
R	Respiratory system
S	Sensory organs
V	Various

ATC drug classification

ATC1: 14 classes

ATC2: 84 classes



Alimentary tract and metabolism

Drugs used in diabetes

Blood glucose lowering drugs

Biguanides

Metformin

Australian tourism

```
#| label: ausmap
#| fig-height: 3
#| echo: false
library(sf)
# Use Okabe-Ito color-blind friendly color palette
state_colors <- c(
  `New South Wales` = "#56b4e9",
  `Victoria` = "#0072b2",
  `Queensland` = "#009e73",
  `South Australia` = "#f0e442",
  `Northern Territory` = "#d55e00",
  `Western Australia` = "#e69f00",
  `Tasmania` = "#cc79a7",
  `Australian Capital Territory` = "#cccccc"
)
read_sf("tourism/Tourism_Regions_2020.shp") |>
  rename(State = "STE_NAME16") |>
  ggplot() +
  geom_sf(aes(fill = State), alpha = 0.8) +
```

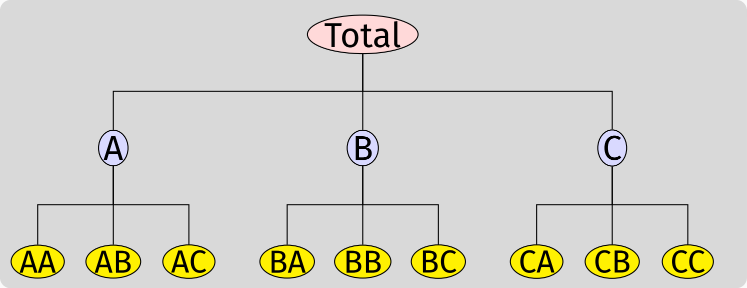
Australian tourism

```
tourism
```

- Quarterly data on visitor nights, 1998:Q1 – 2017:Q4
- From: *National Visitor Survey*, based on annual interviews of 120,000 Australians aged 15+, collected by Tourism Research Australia.
- Split by 8 states and 76 regions
- Split by purpose of travel
  - Holiday
  - Visiting friends and relatives (VFR)
  - Business
  - Other
- 304 bottom-level series

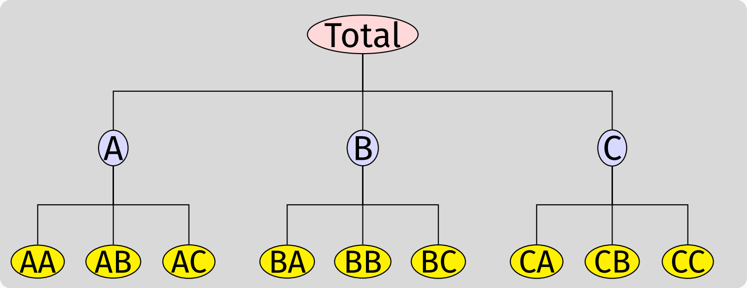
Hierarchical time series

A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.



Hierarchical time series

A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.

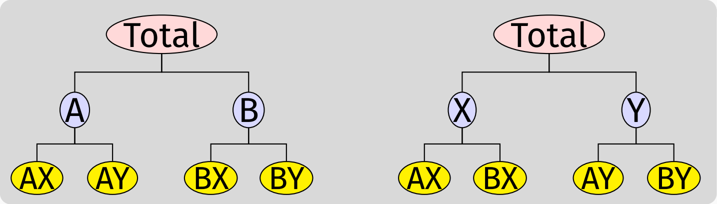


Examples

- PBS sales by ATC groups
- Tourism demand by states, regions

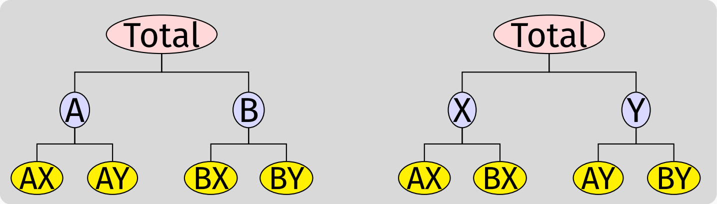
Grouped time series

A **grouped time series** is a collection of time series that can be grouped together in a number of non-hierarchical ways.



Grouped time series

A **grouped time series** is a collection of time series that can be grouped together in a number of non-hierarchical ways.



Examples

- Tourism by state and purpose of travel
- Retail sales by product groups/sub groups, and by countries/regions

Creating aggregates

```
PBS |>
  aggregate_key(ATC1 / ATC2, Scripts = sum(Scripts)) |>
  filter(Month == yearmonth("1991 Jul")) |>
  print(n = 18)
```

Creating aggregates

```
tourism |>
  aggregate_key(Purpose * (State / Region), Trips = sum(Trips)) |>
  filter(Quarter == yearquarter("1998 Q1")) |>
  print(n = 15)
```

Creating aggregates

- Similar to `summarise()` but using the key structure
- A grouped structure is specified using `grp1 * grp2`
- A nested structure is specified via `parent / child`.
- Groups and nesting can be mixed:

```
(country/region/city) * (brand/product)
```

- All possible aggregates are produced.
- These are useful when forecasting at different levels of aggregation.

Outline

1. Time series data and tsibbles
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Lab Session 1
5. Time plots
6. Lab Session 2
7. Seasonal plots
8. Lab Session 3
9. Seasonal or cyclic?
10. Lag plots and autocorrelation
11. Lab Session 4
12. White noise
13. Lab Session 5
14. Per capita adjustments
15. Lab Session 6
16. Inflation adjustments
17. Mathematical transformations
18. Lab Session 7
19. Time series decompositions
20. Lab Session 8
21. Multiple seasonality
22. A cautionary tale
23. Notice: Material planned to change
24. STL Features
25. Lab Session 9
26. Dimension reduction for features
27. Lab Session 10

The problem

1. How to forecast time series at all nodes such that the forecasts add up in the same way as the original data?
2. Can we exploit relationships between the series to improve the forecasts?

The problem

1. How to forecast time series at all nodes such that the forecasts add up in the same way as the original data?
2. Can we exploit relationships between the series to improve the forecasts?

The solution

1. Forecast all series at all levels of aggregation using an automatic forecasting algorithm.
(e.g., `ETS`, `ARIMA`, …)
2. Reconcile the resulting forecasts so they add up correctly using least squares optimization (i.e., find closest reconciled forecasts to the original forecasts).
3. This is available using `reconcile()`.

Forecast reconciliation

```
{r tourismets_reconciled, message=FALSE} tourism |>   aggregate_key(Purpose * (State /
Region), Trips = sum(Trips)) |>   model(ets = ETS(Trips)) |>   reconcile(ets_adjusted
= min_trace(ets)) |>   forecast(h = 2)
```

Hierarchical and grouped time series

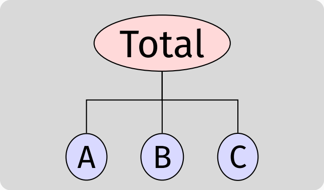
Every collection of time series with aggregation constraints can be written as

$$\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{b}_t$$

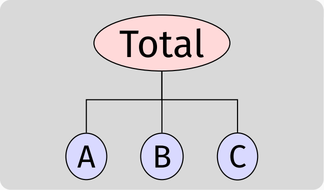
where

- $\boldsymbol{y}_t$ is a vector of all series at time t
- $\boldsymbol{b}_t$ is a vector of the most disaggregated series at time t
- $\boldsymbol{S}$ is a "summing matrix'' containing the aggregation constraints.

Hierarchical time series



Hierarchical time series

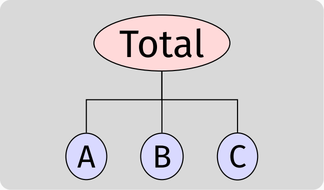


y_t : observed aggregate of all series at time t.

$y_{X,t}$: observation on series X at time t.

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

Hierarchical time series



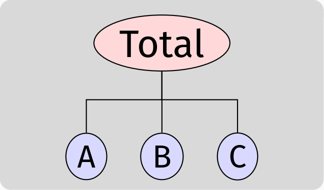
y_t : observed aggregate of all series at time t.

$y_{X,t}$: observation on series X at time t.

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

$$\boldsymbol{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix}$$

Hierarchical time series



y_t : observed aggregate of all series at time t.

$y_{X,t}$: observation on series X at time t.

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

$$\boldsymbol{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix} y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix}}_{\boldsymbol{b}_t}$$

$$\boxed{\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{b}_t}$$

Forecasting notation

Let $\hat{\boldsymbol{y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{y}_t$.

Forecasting notation

Let $\hat{\boldsymbol{y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{y}_t$.
(In general, they will not "add up".)

Forecasting notation

Let $\hat{\boldsymbol{y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{y}_t$.
(In general, they will not "add up".)

Reconciled forecasts must be of the form:

$$\tilde{\boldsymbol{y}}_n(h) = \boldsymbol{S}\boldsymbol{G}\hat{\boldsymbol{y}}_n(h)$$

for some matrix $\boldsymbol{G}$.

Forecasting notation

Let $\hat{\boldsymbol{y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{y}_t$.
(In general, they will not "add up''.)

Reconciled forecasts must be of the form:

$$\tilde{\boldsymbol{y}}_n(h) = \boldsymbol{S}\boldsymbol{G}\hat{\boldsymbol{y}}_n(h)$$

for some matrix $\boldsymbol{G}$.

- $\boldsymbol{G}$ extracts and combines base forecasts $\hat{\boldsymbol{y}}_n(h)$ to get bottom-level forecasts.
- $\boldsymbol{S}$ adds them up

Optimal combination forecasts

Main result

The best (minimum sum of variances) unbiased forecasts are obtained when $\boldsymbol{G} = (\boldsymbol{S}'\Sigma_h^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}'\Sigma_h^{-1}$, where Σ_h is the h-step base forecast error covariance matrix.

Optimal combination forecasts

Main result

The best (minimum sum of variances) unbiased forecasts are obtained when $\boldsymbol{G} = (\boldsymbol{S}'\Sigma_h^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}'\Sigma_h^{-1}$, where Σ_h is the h-step base forecast error covariance matrix.

$$\tilde{\boldsymbol{y}}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Sigma_h^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}'\Sigma_h^{-1}\hat{\boldsymbol{y}}_n(h)$$

Problem: Σ_h hard to estimate, especially for $h > 1$.

Solutions:

- Ignore Σ_h (OLS) [`min_trace(method='ols')`]
- Assume $\Sigma_h = k_h\Sigma_1$ is diagonal (WLS) [`min_trace(method='wls')`]
- Assume $\Sigma_h = k_h\Sigma_1$ and estimate it (GLS) [`min_trace(method='shrink')` (the default)]

Features

- Covariates can be included in initial forecasts.
- Adjustments can be made to initial forecasts at any level.
- Very simple and flexible method. Can work with *any* hierarchical or grouped time series.
- Conceptually easy to implement: regression of base forecasts on structure matrix.

Outline

- Time series data and tsibbles
- Example: Australian prison population
- Example: Australian pharmaceutical sales
- Lab Session 1
- Time plots
- Lab Session 2
- Seasonal plots
- Lab Session 3
- Seasonal or cyclic?
- Lag plots and autocorrelation
- Lab Session 4
- White noise
- Lab Session 5
- Per capita adjustments
- Lab Session 6
- Inflation adjustments
- Mathematical transformations
- Lab Session 7
- Time series decompositions
- Lab Session 8
- Multiple seasonality
- A cautionary tale
- Notice: Material planned to change
- STL Features
- Lab Session 9
- Dimension reduction for features
- Lab Session 10

Example: Australian tourism

```
tourism_agg <- tourism |>
  aggregate_key(Purpose * (State / Region),
    Trips = sum(Trips)
  )
fc <- tourism_agg |>
  filter_index(. ~ "2015 Q4") |>
  model(ets = ETS(Trips)) |>
  reconcile(ets_adjusted = min_trace(ets)) |>
  forecast(h = "2 years")
```

Example: Australian tourism

```
{r fctourism2, dependson='fctourism'} fc |>   filter(is_aggregated(Purpose) &
is_aggregated(State)) |>   autoplot(tourism_agg, level = 95)
```

Example: Australian tourism

```
{r fctourism3, dependson='fctourism'} fc |>   filter(is_aggregated(Purpose) &
State == "VIC" & is_aggregated(Region)) |>   autoplot(tourism_agg, level = 95)
```

Example: Australian tourism

```
{r fctourism4, dependson='fctourism'} fc |>   filter(is_aggregated(Purpose) &
Region == "Melbourne") |>   autoplot(tourism_agg, level = 95)
```

Example: Australian tourism

```
{r fctourism5, dependson='fctourism'} fc |>   filter(is_aggregated(Purpose) &
Region == "Snowy Mountains") |>   autoplot(tourism_agg, level = 95)
```

Example: Australian tourism

{r fctourism6, dependson='fctourism'} fc |>   filter(Purpose == "Holiday" & Region == "Barossa") |>   autoplot(tourism_agg, level = 95)

Example: Australian tourism

```
{r fctourism7, dependson='fctourism'} fc |>   filter(is_aggregated(Purpose) &
Region == "MacDonnell") |>   autoplot(tourism_agg, level = 95)
```

Example: Australian tourism

```
fc <- tourism_agg |>
  filter_index(. ~ "2015 Q4") |>
  model(
    ets = ETS(Trips),
    arima = ARIMA(Trips)
  ) |>
  mutate(
    comb = (ets + arima) / 2
  ) |>
  reconcile(
    ets_adj = min_trace(ets),
    arima_adj = min_trace(arima),
    comb_adj = min_trace(comb)
  ) |>
  forecast(h = "2 years")
```

Forecast evaluation

```
{r fcaccuracy, dependson='fctourismcomb'} fc |> accuracy(tourism_agg)
```

Forecast evaluation

{r fcaccuracy2, dependson='fctourismcomb'} fc |> accuracy(tourism_agg) |> group_by(.model) |> summarise(MASE = mean(MASE)) |> arrange(MASE)

Outline

Lab Session 20

- Prepare aggregations of the PBS data by Concession, Type, and ATC1.
- Use forecast reconciliation with the PBS data, using ETS, ARIMA and SNAIVE models, applied to all but the last 3 years of data.
- Which type of model works best?
- Does the reconciliation improve the forecast accuracy?
- Why doesn't the reconcililation make any difference to the SNAIVE forecasts?