Time series graphics

Print-friendly PDF version

Before you start

This is an in-class lab: work through the material below at your own pace (in pairs or small groups is fine) — I'll walk around to help, and we'll stop along the way to discuss as a group.

The lecture covered the *theory* — trend, seasonality, cycles, autocorrelation, white noise, stationarity. Today you apply it: load real data, produce the graphics yourself, and interpret what you see. Don't just make the plots — for every plot, write down (in the text, or on paper) what it tells you.

```
library(fpp3)
library(tidyverse)
```

Cheat sheet: turning a date/time column into a tsibble index

Frequency	Function
Annual	`start:end`
Quarterly	`yearquarter()`
Monthly	`yearmonth()`
Weekly	`yearweek()`
Daily	`as_date()`, `ymd()`
Sub-daily	`as_datetime()`, `ymd_hms()`

A tsibble is created with `as_tsibble(index = <time column>, key = <grouping column(s)>)`. To re-aggregate an *existing* tsibble to a coarser time scale, use `group_by_key() %>% index_by(<new time column> = ~<function of the old one>) %>% summarise(...)`.

Cheat sheet: graphics functions

Plot	Function
Time plot	`autoplot(data, variable)`
Seasonal plot	`gg_season(data, variable, period = ...)`
Seasonal subseries plot	`gg_subseries(data, variable)`

Plot	Function
Scatter plot	`ggplot(data, aes(x, y)) + geom_point()`
Lag plot	`gg_lag(data, variable, lags = 1:k)`
Autocorrelation plot	`ACF(data, variable) %>% autoplot()`
Distribution	`geom_histogram()`, `geom_boxplot()`, `stat_density()`

Warm-up: city temperatures across Norway

A quick warm-up before the main case, using daily temperature data for four Norwegian cities — a first taste of the `key` argument on something familiar.

```
citytemp <- readr::read_csv2("https://raw.githubusercontent.com/holleland/BAN430/master/data/
                             show_col_types = FALSE)
```

```
i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
citytemp_ts <- citytemp %>%
  mutate(date = as_date(date, format = "%d.%m.%Y")) %>%
  filter(!is.na(date)) %>%
  as_tsibble(index = date, key = name) %>%
  fill_gaps()

citytemp_ts
```

```
# A tsibble: 35,068 x 4 [1D]
# Key:       name [4]
   name              station date       meanTemp
   <chr>             <chr>   <date>        <dbl>
 1 Bergen - Florida  SN50540 2000-01-01      6.3
 2 Bergen - Florida  SN50540 2000-01-02      6.3
 3 Bergen - Florida  SN50540 2000-01-03      6.7
 4 Bergen - Florida  SN50540 2000-01-04      4.6
 5 Bergen - Florida  SN50540 2000-01-05      4.6
 6 Bergen - Florida  SN50540 2000-01-06      6.5
 7 Bergen - Florida  SN50540 2000-01-07      6.3
```

```
 8 Bergen - Florida SN50540 2000-01-08      6.4
 9 Bergen - Florida SN50540 2000-01-09      4.1
10 Bergen - Florida SN50540 2000-01-10      5.3
# i 35,058 more rows
```

Q: How many series are in `citytemp_ts`? What's the key, and what's the index? (`fill_gaps()` is there because there are a few missing days per station; `gg_season()` below won't run without it.)

Now make these three plots:

```
# Time plot: one line per city
```

```
# Seasonal plot (period = "year")
```

```
# Seasonal subseries plot - hint: aggregate to monthly means first with
# group_by_key() %>% index_by() %>% summarise(), same as wp_month later
```

Now put a number on what you saw in the plots. For each city, compute the **average temperature per calendar month** (Jan, Feb, …, Dec, pooling across all years — not year-month), then the **minimum**, **maximum**, and **range** of those 12 monthly averages.

```
# Hint: mutate(month = month(date, label = TRUE)) %>% group_by(name, month)
# %>% summarise(...) to get the 12 monthly averages per city, then
# group_by(name) %>% summarise(min = ___, max = ___, range = ___)
```

Q: Do all four cities have roughly the same seasonal *shape*? Do they have the same seasonal *amplitude* (range)? Does the ranking from your table match what the seasonal plot suggested? Any guesses why the amplitudes differ, geographically?

The case: offshore wind power

You have five years (2015–2019) of **hourly** simulated power production data (in MW) for two proposed Norwegian offshore wind farm locations: **Utsira Nord** and **Sørlig Nordsjø 2**.

> You work as an analyst asked to give an initial, purely data-driven, recommendation on where to build the **first** offshore wind farm. You have no cost or grid-connection information — only five years of production data. What can the data alone tell you?

Keep that question in mind as you work through the tasks below — you'll return to it at the end.

Task 0: Load the data

The data lives in the `ban430data` package as `windpower` — install it if you haven't already.

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

```
library(ban430data)
data(windpower)

wp_hour <- windpower
wp_hour
```

```
# A tsibble: 87,648 x 3 [1h] <?>
# Key:       Place [2]
   Place             powerprod datetime
   <chr>                 <dbl> <dttm>
 1 Sørlig Nordsjø 2         15 2015-01-01 00:00:00
 2 Sørlig Nordsjø 2         15 2015-01-01 01:00:00
 3 Sørlig Nordsjø 2         15 2015-01-01 02:00:00
 4 Sørlig Nordsjø 2         15 2015-01-01 03:00:00
 5 Sørlig Nordsjø 2         15 2015-01-01 04:00:00
 6 Sørlig Nordsjø 2         15 2015-01-01 05:00:00
 7 Sørlig Nordsjø 2         15 2015-01-01 06:00:00
 8 Sørlig Nordsjø 2         15 2015-01-01 07:00:00
 9 Sørlig Nordsjø 2         15 2015-01-01 08:00:00
10 Sørlig Nordsjø 2         15 2015-01-01 09:00:00
# i 87,638 more rows
```

Q: What is the index variable, and what is the key? Why do we need a key here — for the same reason `citytemp` needed one above?

Task 1: Aggregate to other time scales

Hourly data is hard to read as a time plot over five years. Create three new tsibbles — `wp_day`, `wp_week` and `wp_month` — by aggregating `wp_hour` with the mean production. The daily version is done for you; use the cheat sheet above (and the fact that a `datetime` can be turned into a date with `as.Date()`) to complete the other two.

```r
wp_day <- wp_hour %>%
  group_by_key() %>%
  index_by(date = ~as.Date(.)) %>%
  summarise(powerprod = mean(powerprod, na.rm = TRUE))

wp_day
```

```
# A tsibble: 3,654 x 3 [1D]
# Key:       Place [2]
   Place             date       powerprod
   <chr>             <date>         <dbl>
 1 Sørlig Nordsjø 2 2014-12-31     15
 2 Sørlig Nordsjø 2 2015-01-01     13.1
 3 Sørlig Nordsjø 2 2015-01-02     12.5
 4 Sørlig Nordsjø 2 2015-01-03     15
 5 Sørlig Nordsjø 2 2015-01-04     14.8
 6 Sørlig Nordsjø 2 2015-01-05      4.81
 7 Sørlig Nordsjø 2 2015-01-06     14.2
 8 Sørlig Nordsjø 2 2015-01-07     14.4
 9 Sørlig Nordsjø 2 2015-01-08     15
10 Sørlig Nordsjø 2 2015-01-09      7.26
# i 3,644 more rows
```

```r
wp_week <- wp_hour %>%
  group_by_key() %>%
  index_by(yearweek = ___) %>%
  summarise(powerprod = ___)
```

```r
wp_month <- wp_hour %>%
  group_by_key() %>%
  index_by(yearmonth = ___) %>%
  summarise(powerprod = ___)
```

Task 2: Time plots at every scale

Produce `autoplot()` time plots for `wp_hour`, `wp_day`, `wp_week` and `wp_month`.

Q: At which scale(s) can you see a trend? A cycle? Seasonality? Does the picture change as you aggregate more?

Task 3: Seasonal plots

Use `gg_season()` on an appropriate aggregate (or several) to check for seasonality at different periods: within a day, within a week, within a year.

Q: Is there a daily pattern? A weekly pattern? An annual pattern? Is it the same at both locations?

Task 4: Seasonal subseries plots

Pick the time scale and period where you found the clearest seasonal pattern in Task 3, and use `gg_subseries()` to look at it more closely.

Q: Does `gg_subseries()` show you anything `gg_season()` didn't — e.g. is the seasonal pattern stable across the five years, or changing?

Task 5: Relationship between the two locations

So far you've looked at each location separately. Now compare them directly.

```
# Hint: pivot_wider(names_from = Place, values_from = powerprod) first,
# then ggplot() + geom_point() + geom_smooth()
```

Q: Do the two locations move together? Is the relationship linear? Does your answer change depending on which time scale (hourly/daily/weekly/monthly) you use? Why might that be?

Task 6: Lag plots and autocorrelation

Choose **one** location (`filter(Place == ...)`) and one time scale, and produce a lag plot and an ACF plot.

Q: At which lag(s) is the autocorrelation strongest? Does that match the seasonal period you found in Task 3? Would you say this series looks like white noise, or not — why?

Task 7 (if time permits): Distribution

Use `geom_histogram()`, `stat_density()` or `geom_boxplot()` to look at the *distribution* of production values (e.g. split by month, or by location).

Q: Is the distribution symmetric? Are there a lot of hours/days at or near zero production? What might that mean physically?

Wrap-up discussion

Come back to the opening question:

> If you were to decide where to build the **first** offshore wind farm based solely on the data you have explored today, which location would you choose, and why? What are you *not* able to say from this data alone? Does your recommendation depend on which time scale you looked at?

Discuss with your neighbors, then we'll compare answers across the room.