

# ETC5521: Diving Deeply into Data Exploration

*Exploring data having a space and time context Part I*

**Professor Di Cook**

*Department of Econometrics and Business Statistics*



# Outline

- What is temporal data?
- What is exploratory temporal data analysis?
- Using temporal objects in R: `tsibble`
- Data wrangling: aggregation, creating temporal components, missing values
- Plotting conventions: connect the dots; aspect ratio, landscape or portrait
- Calendar plots: arranging daily records into a calendar format
- Visual inference for temporal data
- tignostics: cognostics for temporal data
- Interactive graphics for temporal data
- Exploring longitudinal data, with the `broldgar` package

# Philosophy

Time series analysis is what you do after all the interesting stuff has been done!

Heike Hofmann, 2005

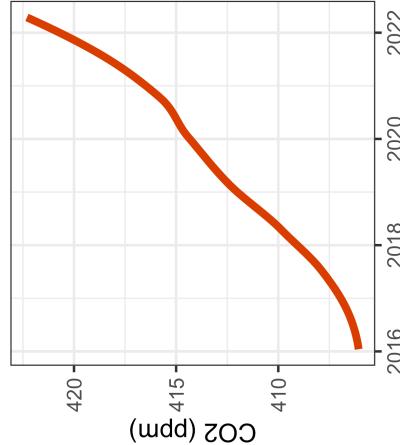
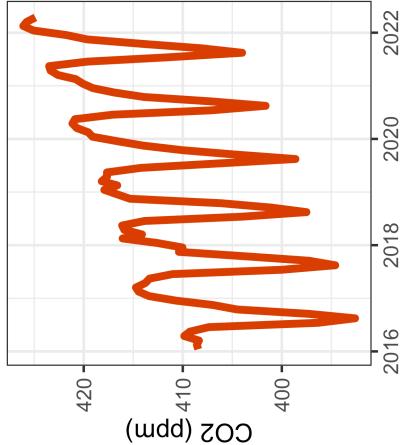
Time series analysis focuses on modeling the temporal dependence. Data needs to have trend, seasonality, anomalies removed first.



Exploratory temporal analysis involves exploring and discovering temporal trend, patterns related to seasons, and anomalies. And possibly also unusual temporal dependence.

# What is temporal data?

- Temporal data has date/time/ordering index variable, call it **time**.
- A time variable has special structure:
  - it can have *cyclical* patterns, eg *seasonality* (summer, winter), an over in cricket
  - the cyclical patterns can be *nested*, eg postcode within state, over within innings
- Measurements are also **NOT independent** - yesterday may influence today.
- It still likely has **non-cyclical patterns**, trends and associations with other variables, eg temperature increasing over time, over is bowled by Elise Perry or Sophie Molineaux



# **tsibble: R temporal object**



The `tsibble` package provides a data infrastructure for tidy temporal data with wrangling tools. Adapting the tidy data principles, `tsibble` is a data- and model-oriented object. In `tsibble`:

- **Index** is a variable with inherent ordering from past to present.
- **Key** is a set of variables that define observational units over time.
- Each observation should be **uniquely identified** by index and key.
- Each observational unit should be measured at a **common interval**, if regularly spaced.

# Regular vs irregular

The Melbourne pedestrian sensor data has a **regular** period. Counts are provided for every hour, at numerous locations.

	Date_Time	Date	Time	Count
1	<date>	<int>	<int>	<int>
1	Birrarun...	2015-01-01	00:00:00	2015-01-01
2	Birrarun...	2015-01-01	01:00:00	2015-01-01
3	Birrarun...	2015-01-01	02:00:00	2015-01-01
4	Birrarun...	2015-01-01	03:00:00	2015-01-01
5	Birrarun...	2015-01-01	04:00:00	2015-01-01
6	Birrarun...	2015-01-01	05:00:00	2015-01-01
7	Birrarun...	2015-01-01	06:00:00	2015-01-01
8	Birrarun...	2015-01-01	07:00:00	2015-01-01
9	Birrarun...	2015-01-01	08:00:00	2015-01-01
10	Birrarun...	2015-01-01	09:00:00	2015-01-01

In contrast, the US flights data, below, is irregular.

```
1 options(width=55)
2 library(nycflights13)
3 flights_ts <- flights |>
4   mutate(dt = ymd_hm(paste(paste(year, month, day, sep="",
5   paste(hour, minute, sep=":"),
6   as_tsibble(index = dt, key = c(origin, dest, carrier,
7   flights_ts
```

Is pedestrian traffic really regular?

ETC5521 Lecture 9 | ddde.numbat.space

# Getting started

Wrangling prior to analysing temporal data includes:

- aggregate by temporal unit.
- **construct temporal units to study seasonality, such as month, week, day of the week, quarter, ...**
- checking and imputing **missings**.

For the airlines data, you can aggregate by multiple quantities, eg number of arrivals, departures, average hourly arrival delay and departure delays.

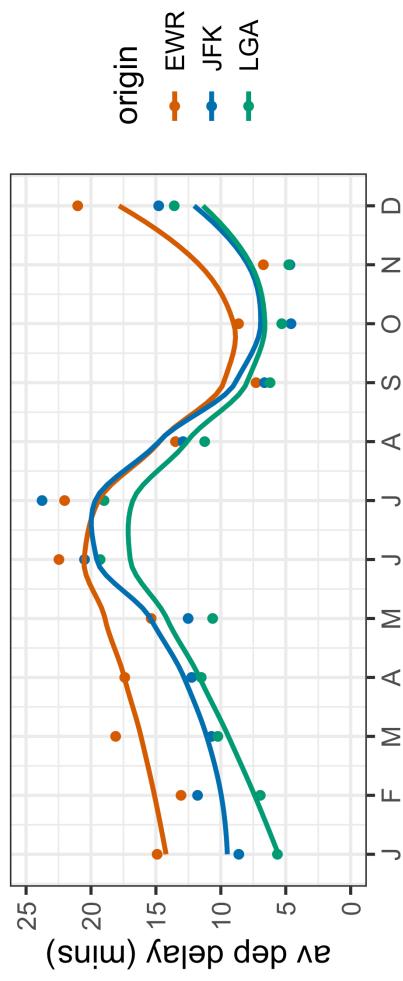
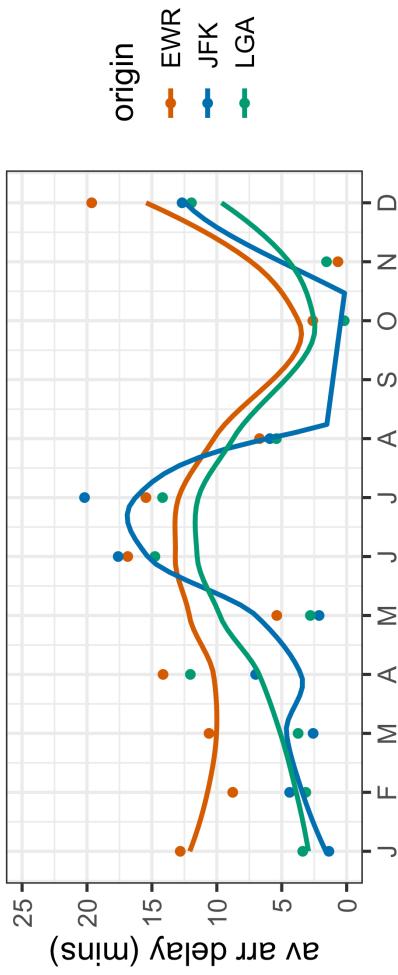
# Aggregating

The US flights data already has some temporal components created, so aggregating by these is easy. Here is departure delay.

► Code

Aggregate by month, but examine arrival delays.

► Code



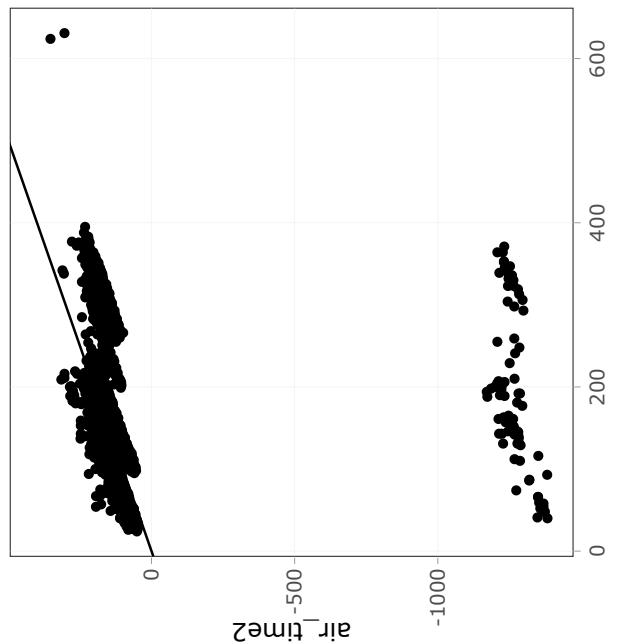
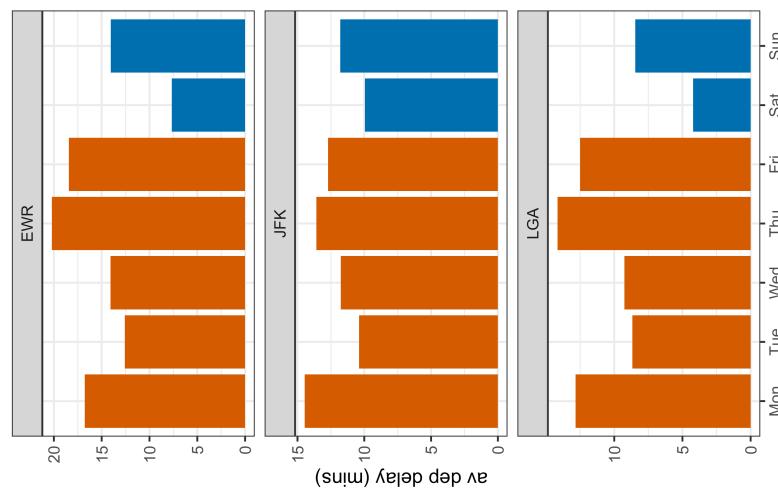
# Constructing temporal units

Week day vs weekend would be expected to have different patterns of delay, but this is not provided.

► Code

Be careful of times!

► Code



Why is this not what we expect?

# Checking and filling missings (1/4)

```
1 set.seed(328)
2 harvest <- tsibble(
3   year = c(2010, 2011, 2013, 2011,
4   2012, 2013),
5   fruit = rep(c("kiwi", "cherry"),
6   each = 3),
7   kilo = sample(1:10, size = 6),
8   key = fruit, index = year
9 )
10 harvest
```

```
# A tsibble: 6 x 3 [1Y]
# Key:   fruit    kilo
#>     year fruit    kilo
#>       <dbl> <chr> <int>
#> 1 2011 cherry      2
#> 2 2012 cherry      7
#> 3 2013 cherry      1
#> 4 2010 kiwi       6
#> 5 2011 kiwi       5
#> 6 2013 kiwi       8
```

Can you see the gaps in time?

Both levels of the key have missings.

# Checking and filling missings (2/4)

```
# A tibble: 6 x 3 [1Y]
# Key:   fruit    kilo
#> year fruit     kilo
#> <dbl> <chr> <int>
#> 1 2011 cherry     2
#> 2 2012 cherry     7
#> 3 2013 cherry     1
#> 4 2010 kiwi       6
#> 5 2011 kiwi       5
#> 6 2013 kiwi       8
```

1 count\_gaps(harvest, .full=TRUE)

```
# A tibble: 2 x 4
#> fruit   from   to   n
#> <chr> <dbl> <dbl> <int>
#> 1 cherry 2010 2010     1
#> 2 kiwi   2012 2012     1
```

One missing in each level, although it is a different year.

Notice how `tsibble` handles this summary so neatly.

# Checking and filling missings (3/4)

```
# A tsibble: 6 x 3 [1Y]
# Key:   fruit [2]
#       year   fruit   kilo
#<dbl> <chr> <int>
1 2011 cherry     2
2 2012 cherry     7
3 2013 cherry     1
4 2010 kiwi       6
5 2011 kiwi       5
6 2013 kiwi       8
```

Make the implicit missing values explicit.

```
1 harvest <- fill_gaps(harvest,
2   .full=TRUE)
3 harvest
```

```
# A tsibble: 8 x 3 [1Y]
# Key:   fruit [2]
#       year   fruit   kilo
#<dbl> <chr> <int>
1 2010 cherry    NA
2 2011 cherry     2
3 2012 cherry     7
4 2013 cherry     1
5 2010 kiwi       6
6 2011 kiwi       5
7 2012 kiwi    NA
8 2013 kiwi       8
```

# Checking and filling missings (4/4)

```
# A tibble: 6 x 3 [1Y]
# Key:   fruit [2]
#       year   fruit   kilo
#<dbl> <chr> <int>
# 1 2011 cherry     2
# 2 2012 cherry     7
# 3 2013 cherry     1
# 4 2010 kiwi      6
# 5 2011 kiwi      5
# 6 2013 kiwi      8
```

We have already seen `na_ma()` function, that imputes using a moving average. Alternatively, `na_interpolation()` uses the previous and next values to impute.

```
1 harvest_nomiss <- harvest |>
  2 group_by(fruit) |>
  3 mutate(kilo =
  4   na_interpolation(kilo)) |>
  5 ungroup()
  6 harvest_nomiss
```

```
# A tibble: 6 x 3 [1Y]
# Key:   fruit [2]
#       year   fruit   kilo
#<dbl> <chr> <int>
# 1 2011 cherry     2
# 2 2012 cherry     7
# 3 2013 cherry     1
# 4 2010 kiwi      6
# 5 2011 kiwi      5
# 6 2013 kiwi      8
```

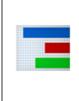
# Plotting conventions

# Conventions

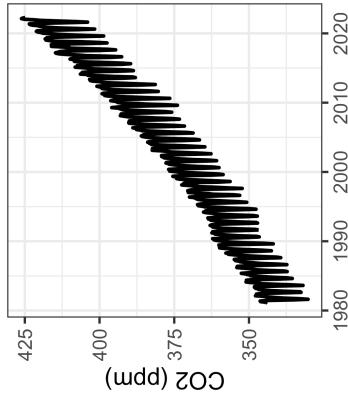
- **lines**: connecting sequential time points reminding the reader that the temporal dependence is important.
- **aspect ratio**: wide or tall? Cleveland, McGill, McGill (1988) argue the average line slope in a line chart should be 45 degrees, which is called banking to 45 degrees. But this is refuted in Talbot, Gerth, Hanrahan (2012) that the conclusion was based on a flawed study. Nevertheless, aspect ratio is an inescapable skill for designing effective plots. For time series, typically a wide aspect ratio is good.
- **conventions**:
  - time on the horizontal axis,
  - ordering of elements like week day, month. Most software organises by alphabetical order, so this needs to be controlled.

# Aspect ratio

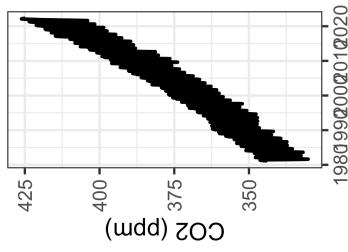
learn R



1 to 1 (may be useless)

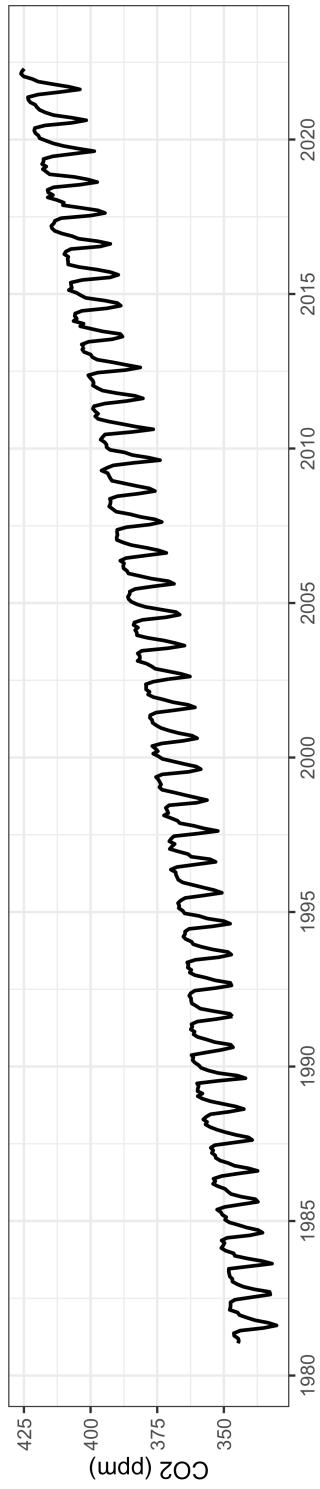


tall & skinny: trend



CO2 at  
Point Barrow,  
Alaska

short & wide: seasonality

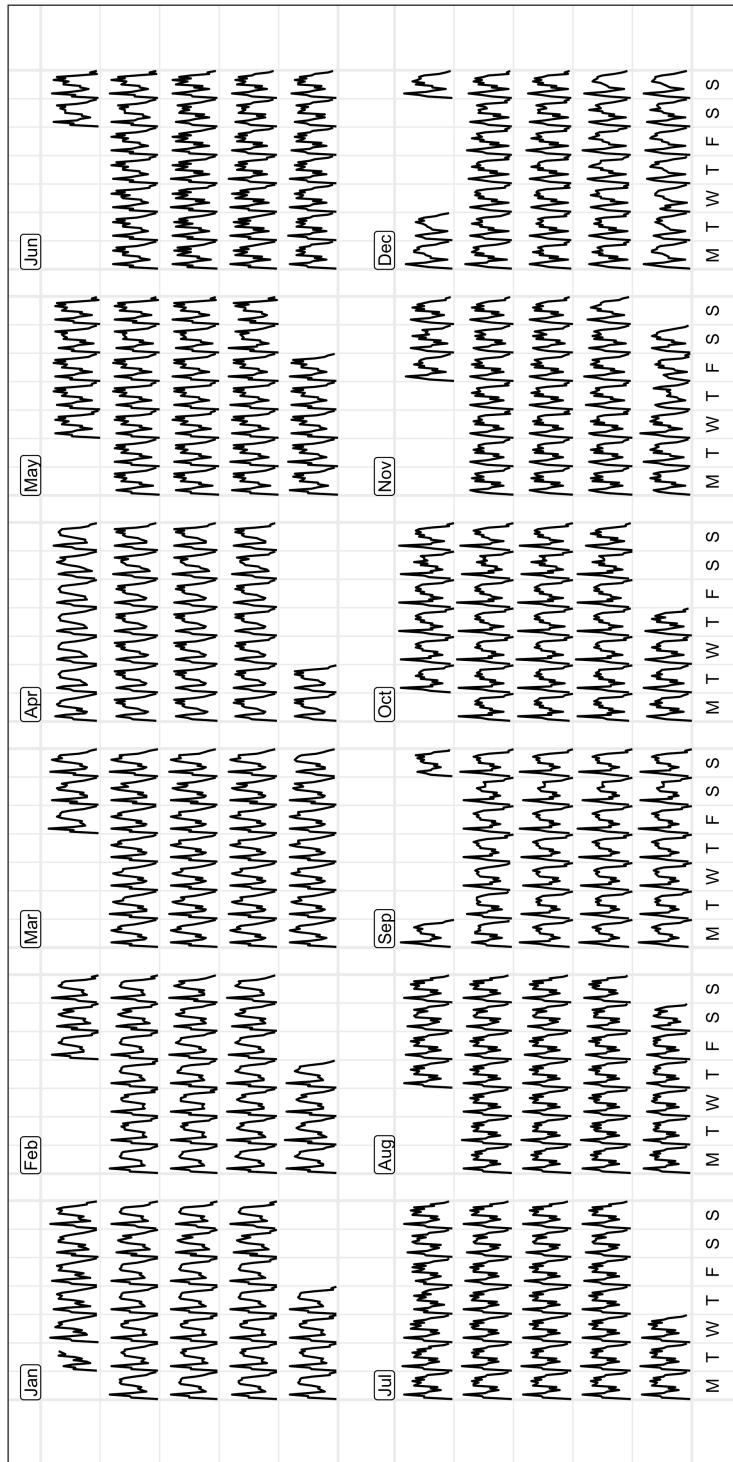




# Calendar plot

# Case study: NYC flights (1/2)

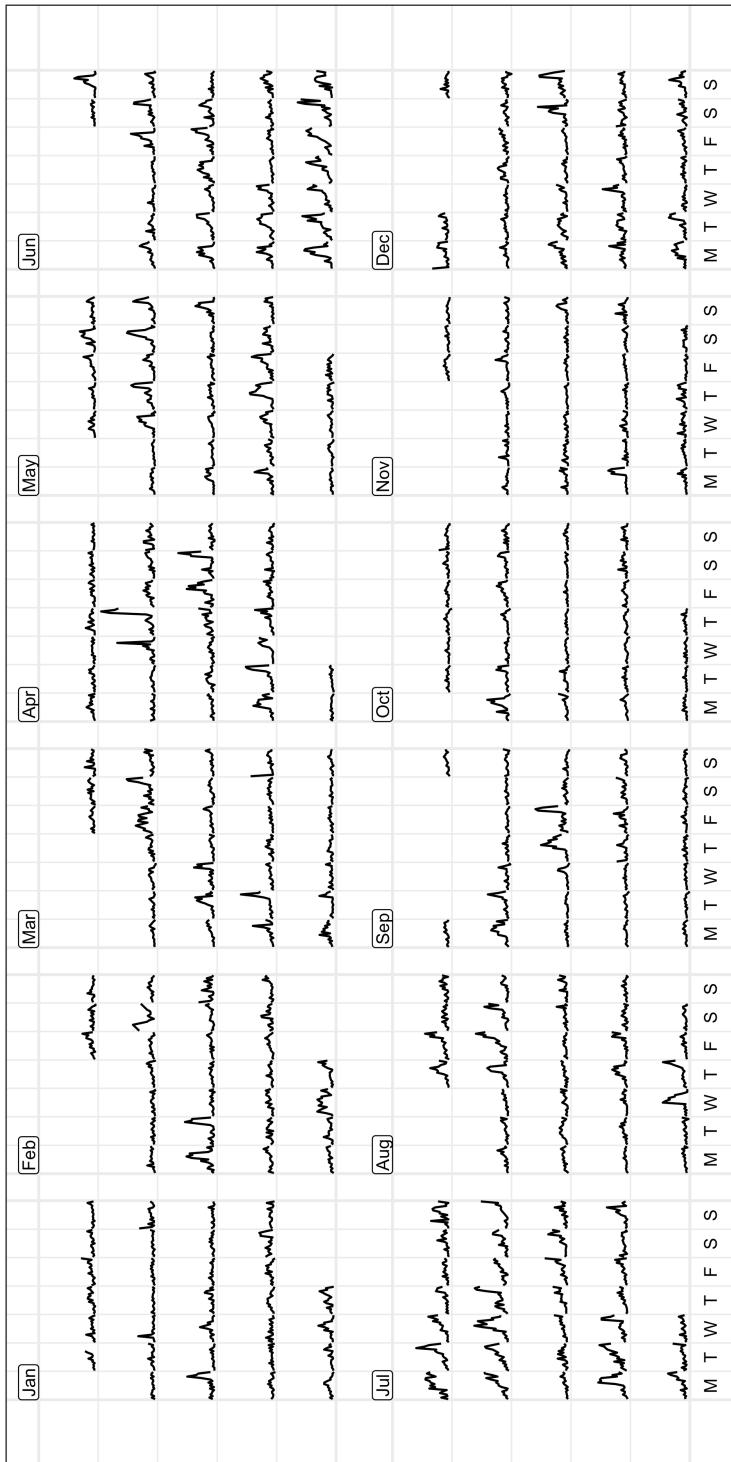
About calendars What do we see? R





# Case study: NYC flights (2/2)

What do we see? R



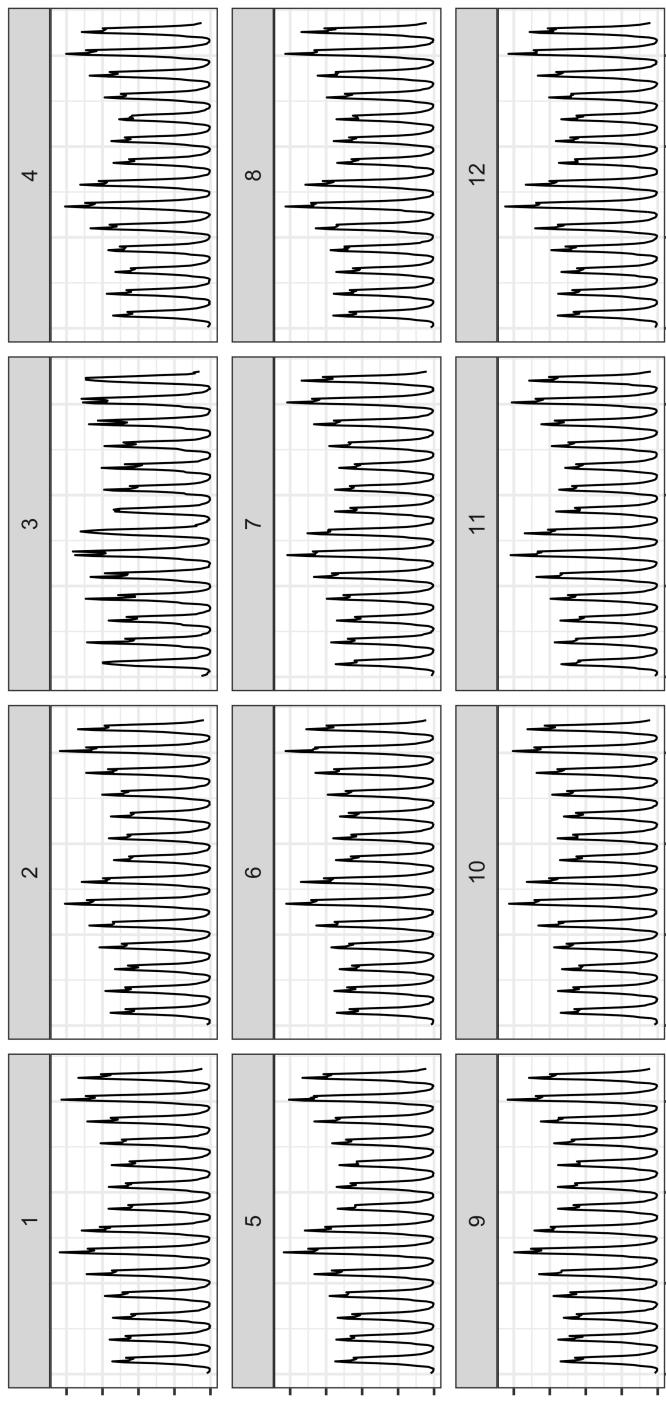


# Visual inference

# Temporal patterns: simulation

► Code

1. Decide on a model
2. Simulate from the model  
to generate nulls

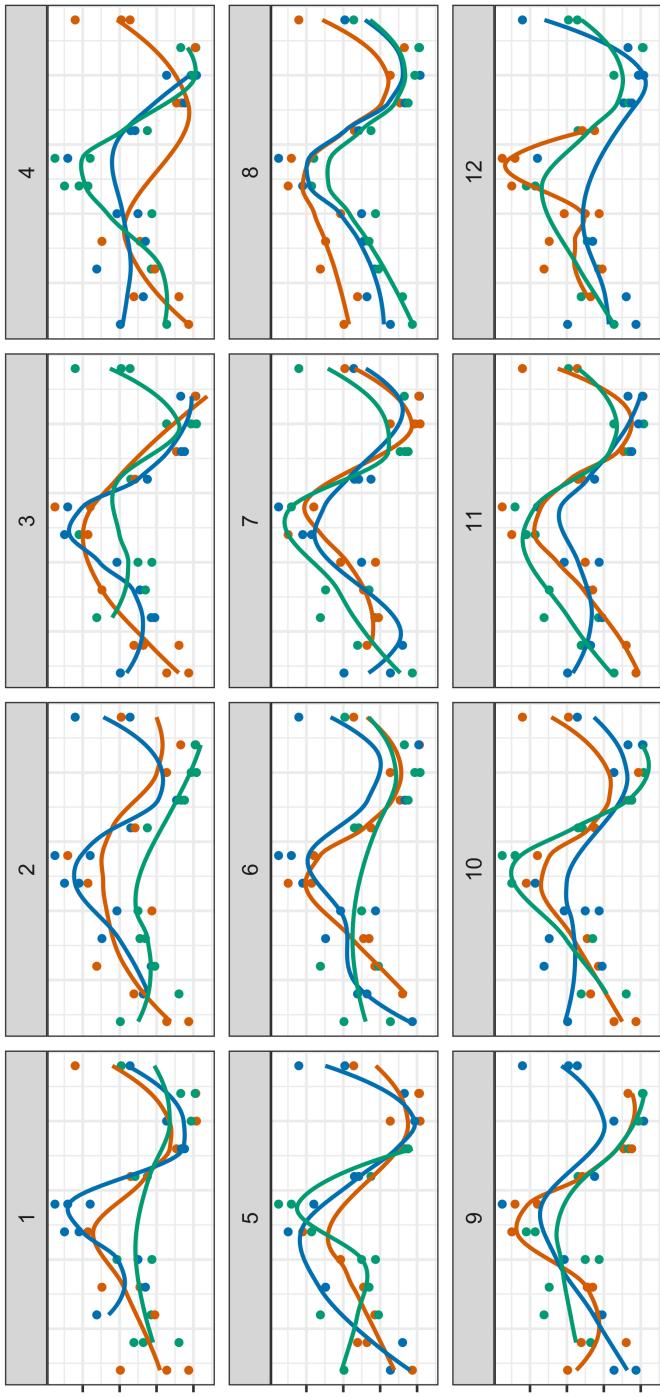


# Association: permutation

## ► Code

Break association between variables. Here `origin` is permuted which breaks association with `dep_delay`, while keeping `month` fixed.

*Which plot has the biggest difference between the three groups?*

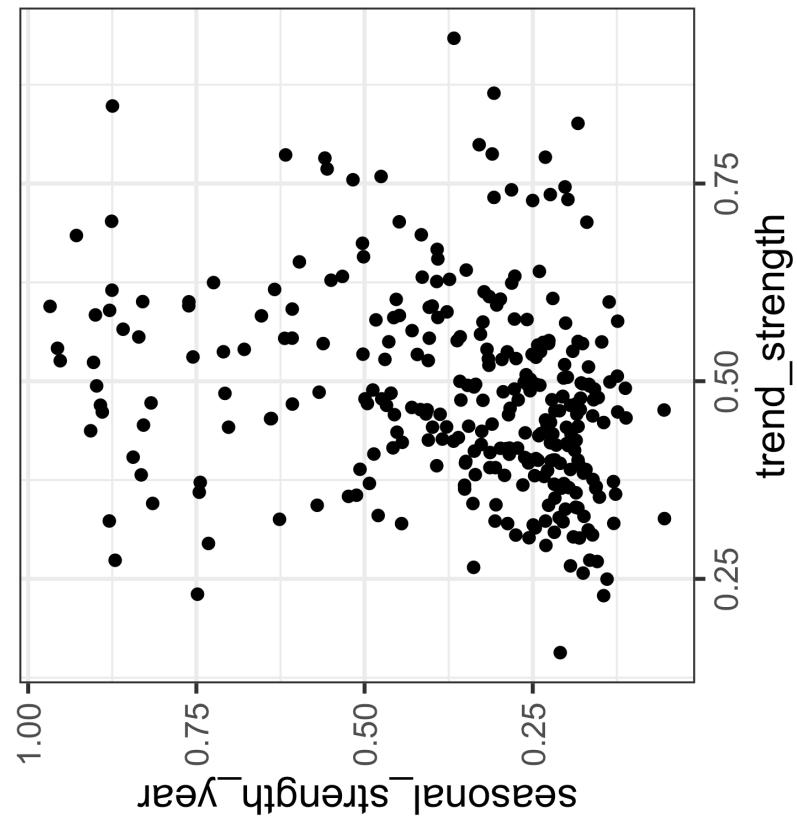


# Tignostics

# feasts: time series features



► Code



The [feasts](#) package provides functions to calculate tignostics for time series.

Remember scagnostics?

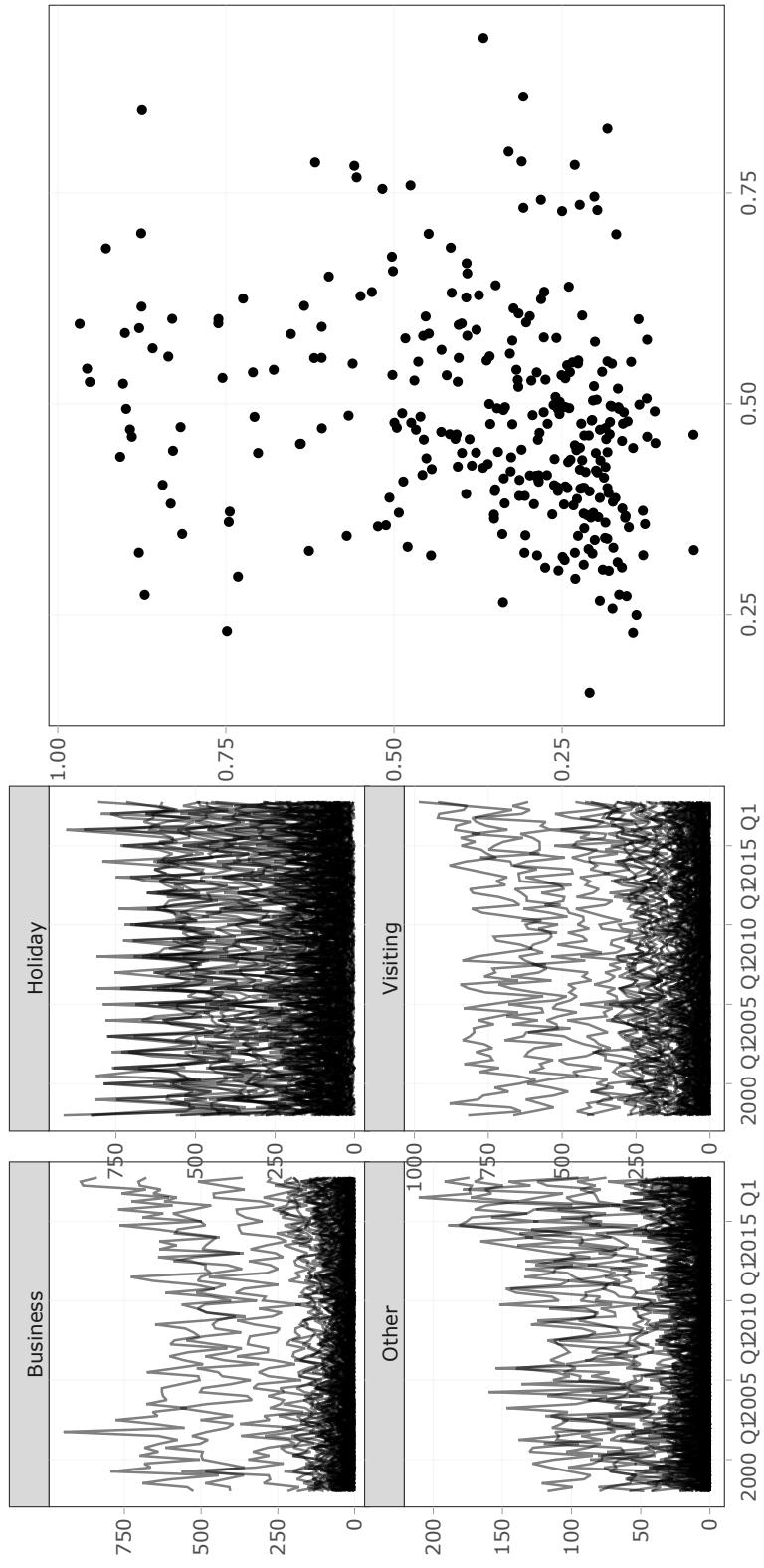
Compute tignostics for each series, for example,

- trend
- seasonality
- linearity
- spikiness
- peak
- trough

# Interactivity

# Interactive exploration with `tsibble`

## ► Code



# Wrapping series

Pedestrian counts at Bourke St Mall, has a daily seasonality.

## DEMO

```
1 pp <- p_bourke |>
2   as_tsibble(index = time) |>
3   ggplot(aes(x=time, y=count)) +
4     geom_line() +
5     theme(aspect.ratio=0.5)
```

## Famous data: Lynx

Annual numbers of lynx trappings for 1821–1934 in Canada. Almost 10 year cycle.

Explore periodicity by wrapping series on itself.

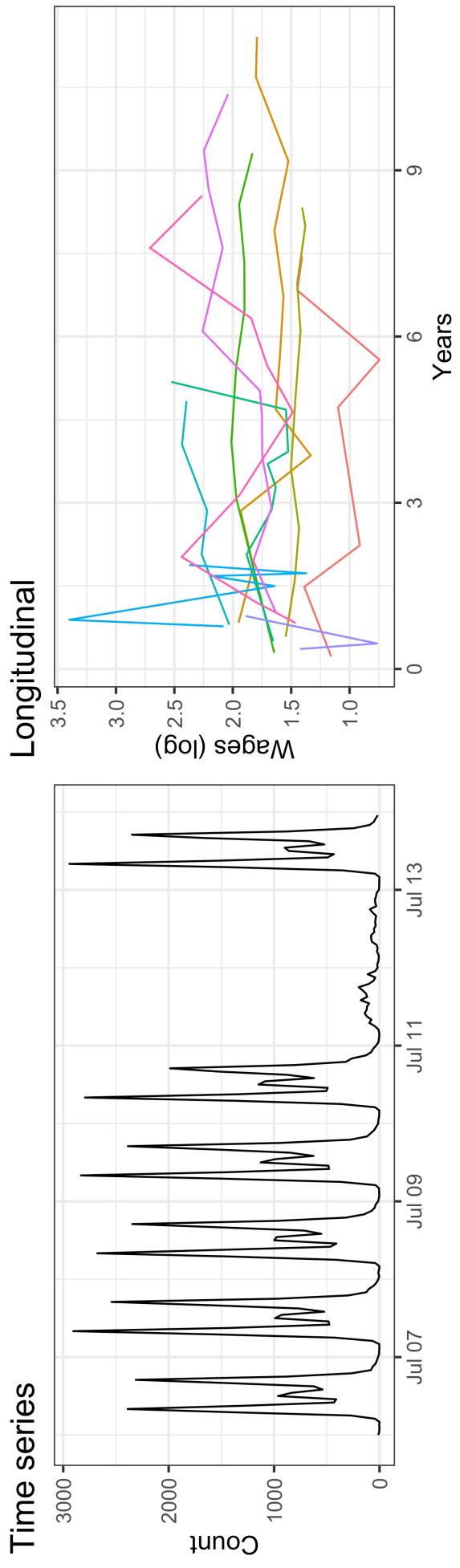
## DEMO

```
1 lynx_tsby <- as_tsibble(lynx) |>
2   rename(count = value)
3 pl <- ggplot(lynx_tsby,
4   aes(x = index, y = count)) +
5   geom_line(size = .2)
6
7 ui <- fluidPage(tsibbleWrapUI("tswrap"))
8 server <- function(input, output, session) {
9   tsibbleWrapServer("tswrap", pp, period = "1 day")
10 }
11
12 shinyApp(ui, server)
```

# Longitudinal data

# Longitudinal vs time series

Longitudinal data tracks the same sample of individuals at different points in time. It often has different lengths and different time points for each individual.

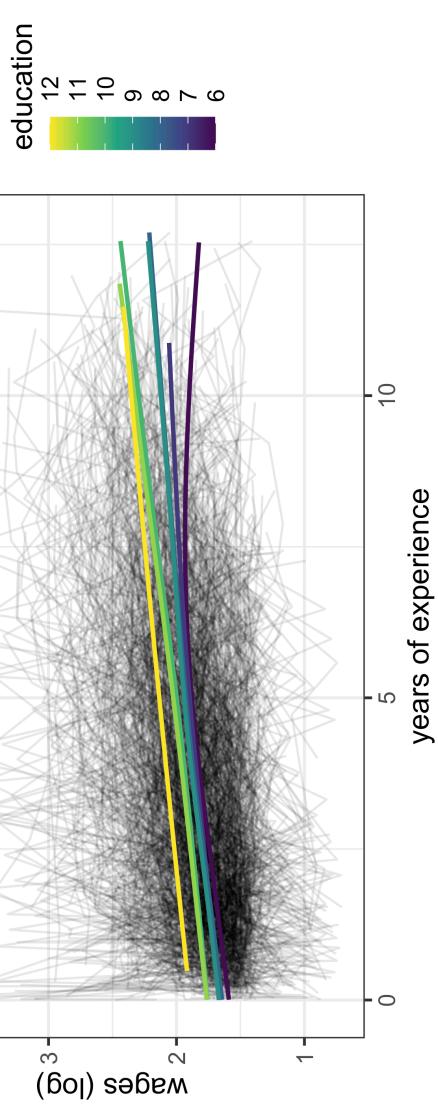


When the time points are the same for each individual, it is usually referred to as **panel data**. When different individuals are measured at each time point, it is called **cross-**

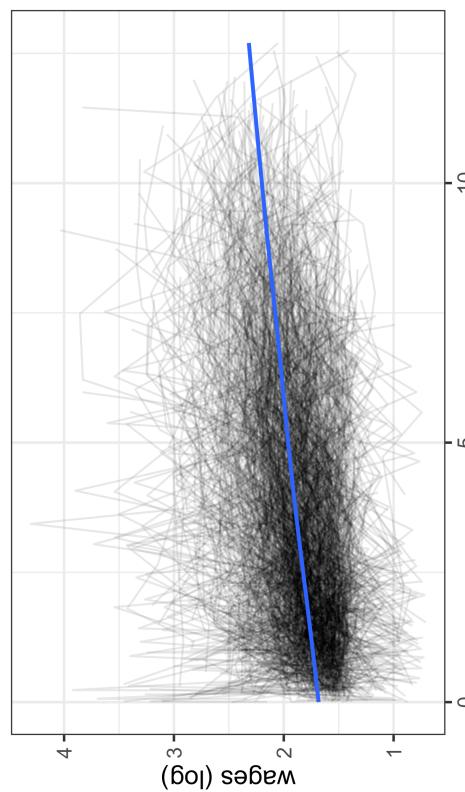
# Overall trend

Log(wages) of 888 individuals, measured at various times in their employment US National Longitudinal Survey of Youth.

► Code



The higher the education level achieved, the higher overall wage, on average.



Wages tend to increase as time in the workforce gets longer, on average.

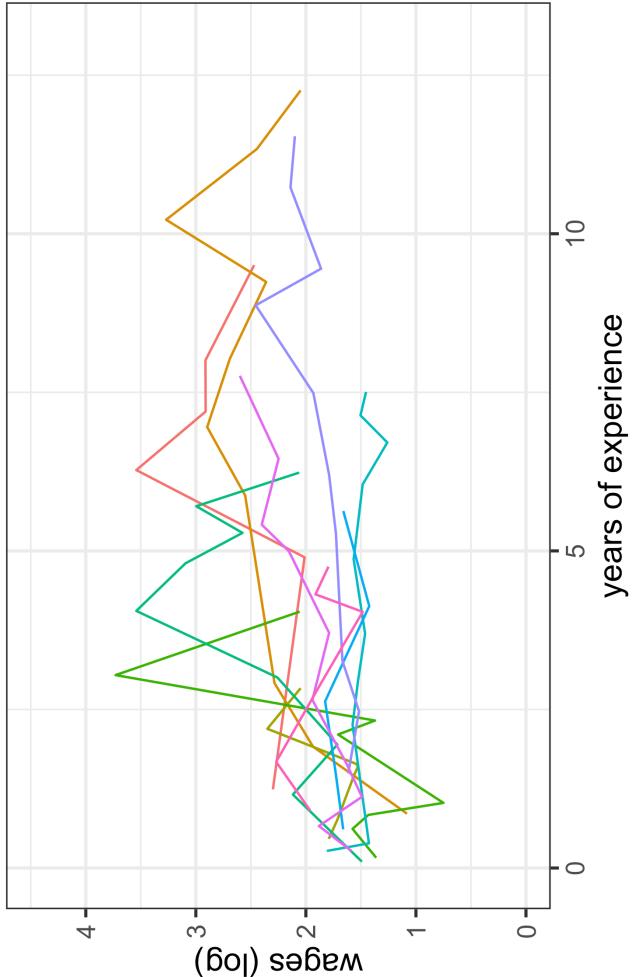
# Eating spaghetti

`brolgar` uses `tsibble` as the data object, and provides:

- sampling individuals
- longnistics for individuals
- diagnostics for statistical models

Sample 1    Sample 2    Sample 3

► Code



*Few individuals experience wages like the overall trend.*

# Individual patterns

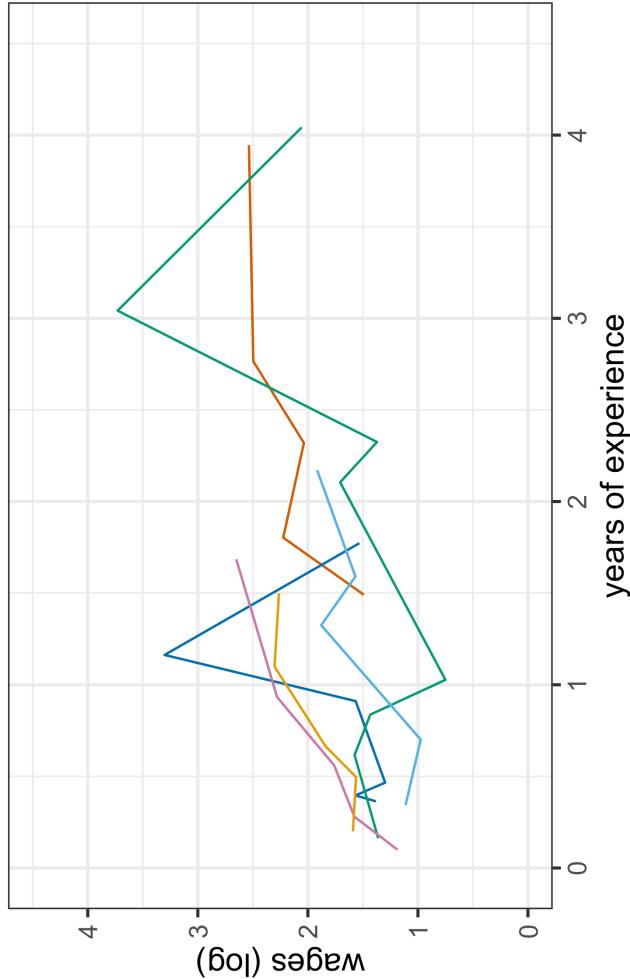
*Remember scagnostics?*

Compute **longnistics** for each subject, for example,

- Slope, intercept from simple linear model
- Variance, standard deviation
- Jumps, differences

► Code

Increasing   Decreasing   Consistent   Volatile



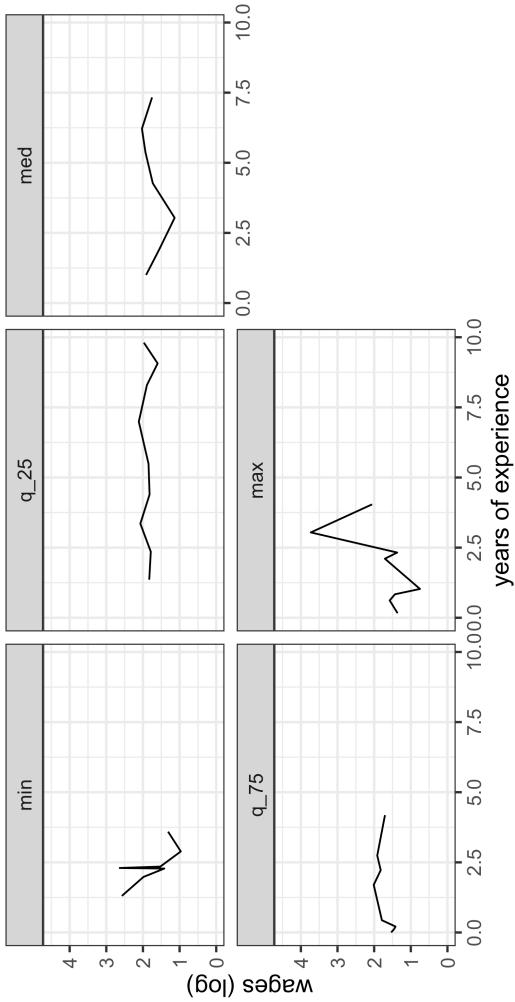
# Individual summaries

► A different style of five number summary: What does average look like? What do extremes look like?

Find those individuals who are **representative** of the min, median, maximum, etc of a particular feature, e.g. trend, using `keys_near()`. This reports the individual who is closest to a particular statistic.

`wages_threenum()` returns the three individuals: min, max and closest to the median value, for a particular feature.

`wages_fivenum()` returns the five individuals: min, max and closest to the median, Q1 and Q3 values, for a particular feature.



# Assessing model fits

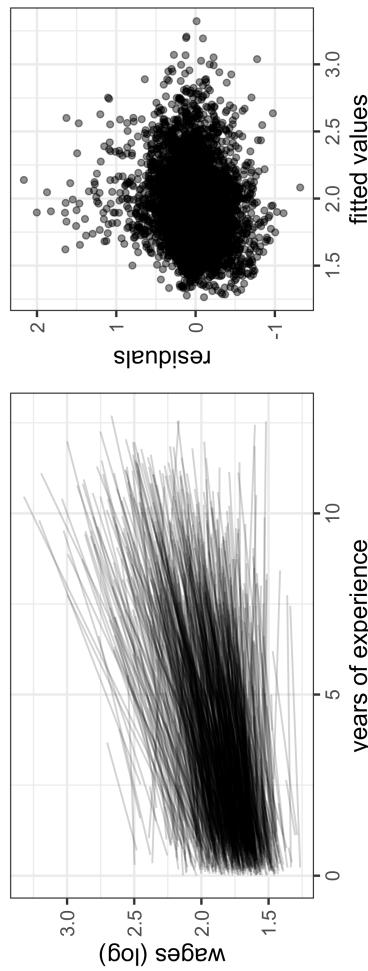
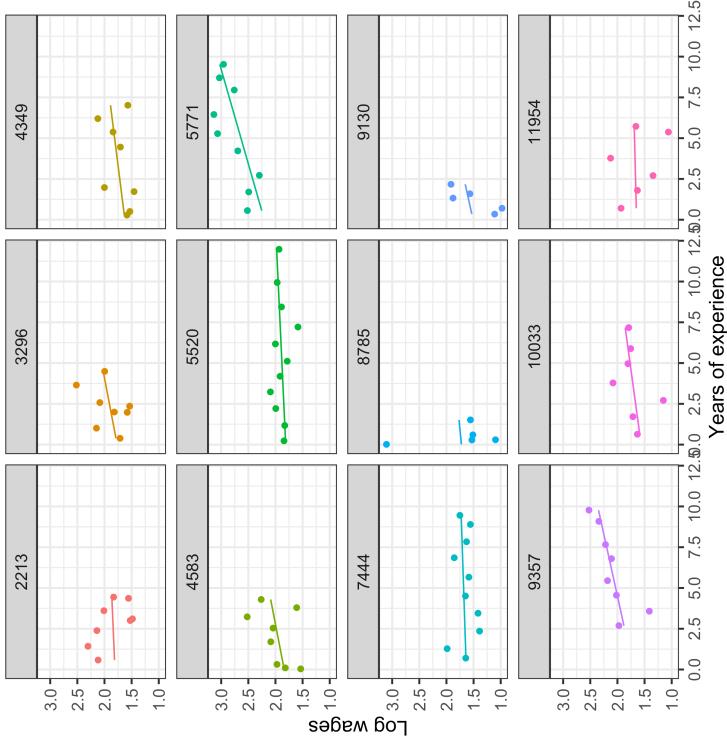
Fit a mixed effect model, education as fixed effect,  
subject random effect using slope.

Summary of the fit

► Code

Diagnosing the fit: each individual model

► Code



# Resources

- Wang, Cook, Hyndman (2019) [A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data](#)
- Wang, Cook, Hyndman, O'Hara-Wild (2019) [tsibble](#)
- O'Hara-Wild, Hyndman, Wang (2020). [fabletools: Core Tools for Packages in the ‘fable’ Framework](#)
- O'Hara-Wild, Hyndman, Wang (2024). [feasts: Feature Extraction and Statistics for Time Series](#)
- Tierney, Cook, Prvan (2020) [Browse Over Longitudinal Data Graphically and Analytically in R](#)