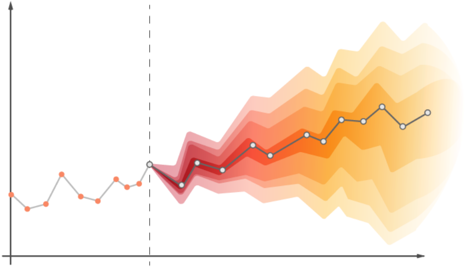
Decomposition and Features

DS-5740 Advanced Statistics



Overview: Week 2

Preliminaries

- None

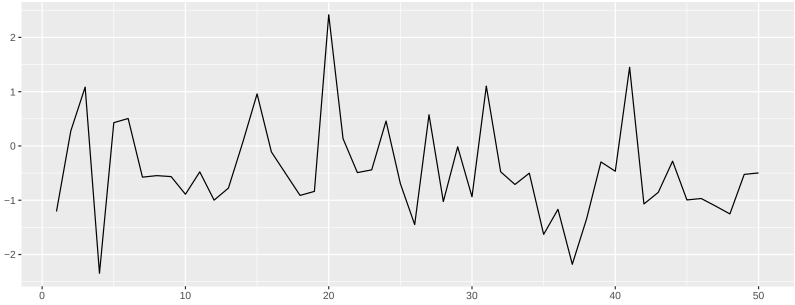
Goals for the Week

- Check and test model residuals
- TSLM Pipeline
- Identify trend and seasonal patterns
- Learn about decomposition to trend and seasonal patterns

Residuals

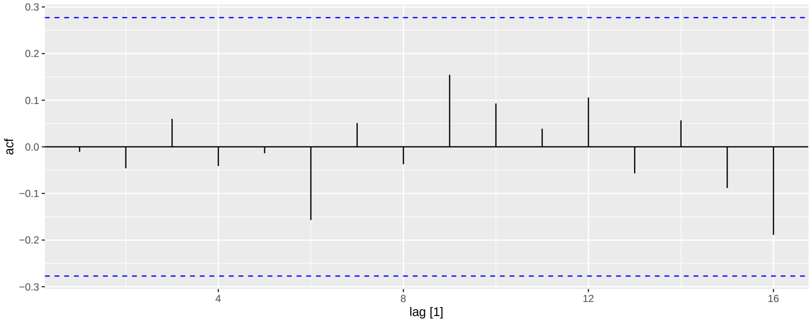
Assume residuals to be white noise

```r
# Set seed for reproducibility
set.seed(1234)

# Random noise
y_wn <- tsibble(sample = 1:50, wn = rnorm(50, 0, 1), index = sample)

# Plot
y_wn %>% autoplot(wn) + labs(x = "", y = "")
```



Decomposition | Residuals

```r
# Random noise
y_wn %>% ACF(wn) %>% autoplot()
```



Where are these autocorrelations coming from?

Where are these autocorrelations coming from?

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$$

Where are these autocorrelations coming from?

```
         [,1]  [,2]  [,3]   [,4]  [,5]  [,6]
sample  1.000 2.000 3.000  4.000 5.000 6.000
wn     -1.207 0.277 1.084 -2.346 0.429 0.506
```

Where are these autocorrelations coming from?

```
         [,1]  [,2]  [,3]   [,4]  [,5]  [,6]
sample  1.000 2.000 3.000  4.000 5.000 6.000
wn     -1.207 0.277 1.084 -2.346 0.429 0.506
```

```
          [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
sample   1.000  2.000  3.000  4.000  5.000  6.000
wn      -1.207  0.277  1.084 -2.346  0.429  0.506
wn_lag1     NA -1.207  0.277  1.084 -2.346  0.429
wn_lag2     NA     NA -1.207  0.277  1.084 -2.346
```

Where are these autocorrelations coming from?

```r
# Lag-1 correlation
sum(
  (wn - mean(wn)) *
    (wn_lag1 - mean(wn)),
  na.rm = TRUE
) / sum((wn - mean(wn))^2)
```

```
[1] -0.01080718
```

```r
# Lag-2 correlation
sum(
  (wn - mean(wn)) *
    (wn_lag2 - mean(wn)),
  na.rm = TRUE
) / sum((wn - mean(wn))^2)
```

```
[1] -0.04598648
```

```r
# `acf` from {tseries}
acf(wn, lag.max = 2, plot = FALSE)
```

```
Autocorrelations of series 'wn', by lag

     0      1      2 
 1.000 -0.011 -0.046 
```

Decomposition | Residuals

Box-Pierce Test

$$\overbrace{Q}^{\text{test statistic}} = T \underbrace{\sum_{k=1}^{\ell}}_{\text{lags}} \overbrace{r_k^2}^{\text{autocorrelations}}$$

Ljung-Box Test

$$\overbrace{Q^*}^{\text{test statistic}} = T(T+2) \underbrace{\sum_{k=1}^{\ell}}_{\text{lags}} (T-k)^{-1} \overbrace{r_k^2}^{\text{autocorrelations}}$$

Usually $\ell = 10$ for non-seasonal data and $h = 2m$ for seasonal data (m = seasonal period)

Statistical test with χ^2 distribution

Pipeline with TSLM

Pipeline with TSLM

Pipeline with TSLM

1. Prepare data
2. Visualize data
3. Model estimation
4. Forecast
5. Visualize (and quantify) forecast

1. Prepare data

Fried, E. I., Papanikolaou, F., & Epskamp, S. (2022). Mental health and social contact during the COVID-19 pandemic: an ecological momentary assessment study. *Clinical Psychological Science*, *10*(2), 340-354. https://doi.org/10.1177/21677026211017839

Download Dataset

- Emotions over two weeks, queried 4 times per day, between March 11-April 4, 2020
- Our goal: Forecast a person's level of feeling **worried**

```r
# Load data
emotions <- read.csv("../data/fried_mental_2022.csv")
```

```r
# Data variables
head(emotions)
```

```
            ID           Scheduled                  Issued
1 User #25290 2020-03-16 12:00:00 2020-03-16 12:00:00 CET
2 User #25290 2020-03-16 15:00:00 2020-03-16 15:00:00 CET
3 User #25290 2020-03-16 18:00:00 2020-03-16 18:00:00 CET
4 User #25290 2020-03-16 21:00:00 2020-03-16 21:00:00 CET
5 User #25290 2020-03-17 12:00:00 2020-03-17 12:00:00 CET
6 User #25290 2020-03-17 15:00:00 2020-03-17 15:00:00 CET
                 Response Duration Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13
1 2020-03-16 12:04:45 CET  285.734  1  1  2  1  1  1  2  3  1   2   3   3   2
2 2020-03-16 15:30:22 CET 1822.742  2  1  2  1  1  1  1  2  1   2   5   3   2
3 2020-03-16 18:03:58 CET  238.774  1  1  3  1  1  1  2  3  1   1   4   4   3
4 2020-03-16 21:16:07 CET  967.132  1  1  3  2  1  1  2  2  1   1   5   3   1
5 2020-03-17 12:33:15 CET  1995.54  1  1  2  1  1  1  2  3  1   1   2   3   1
6 2020-03-17 15:12:01 CET  721.237  3  2  2  1  1  1  1  2  1   1   4   4   2
  Q14 Q15 Q16 Q17 Q18                time        Day beepvar
1   1   1   3   2   5 2020-03-16 12:00:00 2020-03-16       1
2   2   2   2   2   5 2020-03-16 15:00:00 2020-03-16       2
3   3   1   3   1   5 2020-03-16 18:00:00 2020-03-16       3
4   3   1   3   2   5 2020-03-16 21:00:00 2020-03-16       4
5   1   1   2   1   5 2020-03-17 12:00:00 2020-03-17       1
6   2   2   2   1   4 2020-03-17 15:00:00 2020-03-17       2
```

Obtain data for the fourth participant

```
# Participant four
participant <- emotions[emotions$ID == unique(emotions$ID)[4],]

# Obtain time and question variables
questions <- data.frame(
  time = participant$time, # time
  participant[,grep(
    "Q", colnames(participant) # questions
  )]
)
```

Questions from Fried, Papanikolaou, and Epskamp (2022)

Table 1. Ecological Momentary Assessment Items, Queried Four Times per Day Over 2 Weeks

No.	Abbreviation	Item	Change	*p*
1	Relax	I found it difficult to relax	–0.11	.00
2	Irritable	I felt (very) irritable	–0.08	.00
3	Worry	I was worried about different things	–0.12	.00
4	Nervous	I felt nervous, anxious, or on edge	–0.13	.00
5	Future	I felt that I had nothing to look forward	–0.05	.00
6	Anhedonia	I couldn't seem to experience any positive feeling at all	–0.03	.07
7	Tired	I felt tired	–0.05	.00
8	Alone	I felt like I lack companionship, or that I am not close to people	–0.04	.02
9	Social_offline	I spent __ on meaningful, offline, social interaction	–0.02	.14
10	Social_online	I spent __ using social media to kill/pass the time	–0.06	.00
11	Outdoors	I spent __ outside (outdoors)	–0.03	.08
12	C19_occupied	I spent __ occupied with the coronavirus (e.g., watching news, thinking about it, talking to friends about it)	–0.18	.00
13	C19_worry	I spent __ thinking about my own health or that of my close friends and family members regarding the coronavirus	–0.16	.00
14	Home	I spent __ at home (including the home of parents/partner)	0.03	.03

Note: All items had five answer options. Items 1 through 8: 1 = *not at all*, 2 = *slightly*, 3 = *moderately*, 4 = *very*, 5 = *extremely*. Items 9 through 14: 1 = *0 min*, 2 = *1–15 min*, 3 = *15–60 min*, 4 = *1–2 hr*, 5 = *>2 hr*. The "Change" column displays standardized coefficients of change from univariate regression models over the 54 assessment points, followed by *p* values for these changes.

Obtain first 8 questions

```
# First eight questions
data <- questions[,c(
  1, # time
  2:9 # first eight questions
)]

# Relabel questions
colnames(data)[2:9] <- c(
  "relax", "irritable", "worry",
  "nervous", "future", "anhedonia",
  "tired", "alone"
)
```

Pipeline with TSLM | Prepare data

Convert to `tsibble` format

```r
# Remove missing data
data <- na.omit(data)

# Convert to `tsibble`
ts <- data %>%
  mutate(
    time = ymd_hms(time)
  ) %>%
  as_tsibble(
    index = time
  )

# Convert to `tsibble`
ts_fill <- ts %>%
  fill_gaps() # fill in time gaps
# for plotting residuals later
```

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

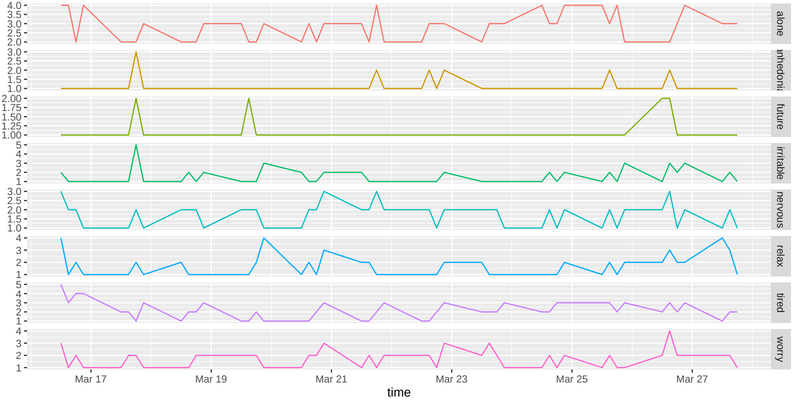
Pipeline with TSLM | Prepare data

```r
# Length of time series
ts_length <- nrow(ts)
ts_fill_length <- nrow(ts_fill)

# Remove last four time points (we'll make a prediction later)
prediction <- ts[
  -c((ts_length - 7):ts_length), # remove last 4 points
]

# For modeling residuals
prediction_fill <- ts_fill[
  1:which(
    ts_fill$time ==
    prediction$time[nrow(prediction)]
  ), # match time points
]

# Save last four time points (we'll compare with prediction)
actual <- ts[
  c((ts_length - 7):ts_length), # keeps last 4 points
] %>%
  fill_gaps()
```

2. Visualize data

```r
# Visualize time series
prediction %>%
  gather(
    "Measure", "Change",
    relax, irritable, worry,
    nervous, future, anhedonia,
    tired, alone
  ) %>%
  ggplot(aes(x = time, y = Change, colour = Measure)) +
  geom_line() +
  facet_grid(vars(Measure), scales = "free_y") +
  labs(y="") +
  guides(colour="none")
```

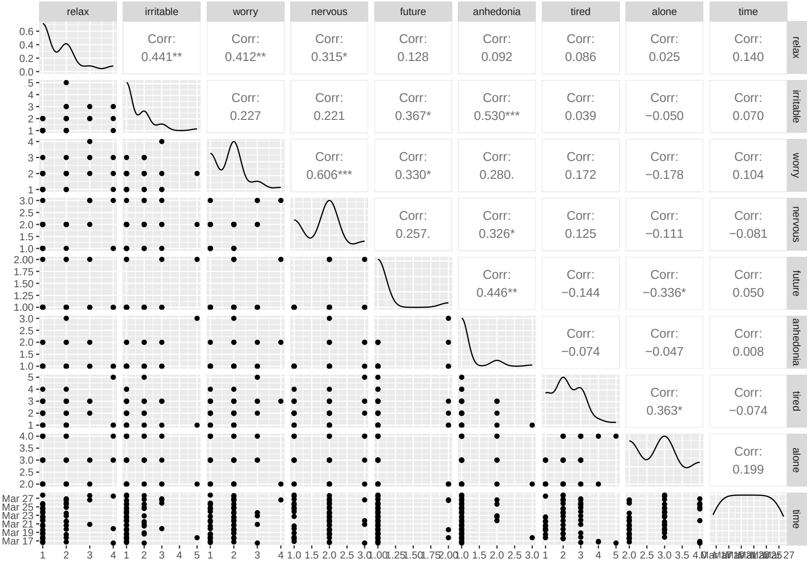
Pipeline with TSLM | Visualize data



Notice any patterns?

```r
# Plot correlations
prediction %>%
  select(-time) %>%
  GGally::ggpairs()
```

Pipeline with TSLM | Visualize data



3. Model estimation

```r
# Fit linear model
fit <- prediction_fill %>% # our data
  model( # model for time series
    tslm = TSLM( # time series linear model
      worry ~ relax + irritable +
      nervous + future + anhedonia +
      tired + alone
    )
  )
```

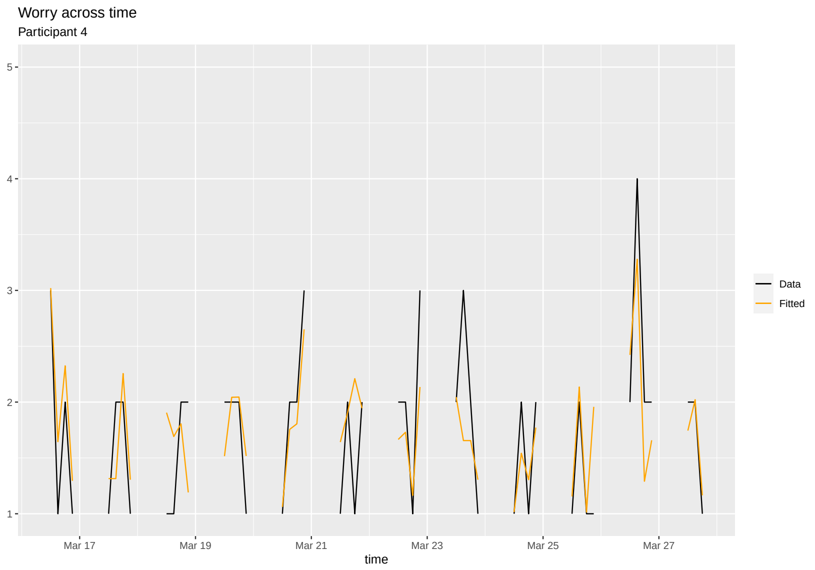
```r
# Report fit
report(fit)
```

```
Series: worry
Model: TSLM

Residuals:
     Min       1Q   Median       3Q      Max
-1.20989 -0.30489 -0.02129  0.30248  1.34452

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.12718    0.57276   0.222  0.82543
relax        0.23974    0.11095   2.161  0.03691 *
irritable   -0.11367    0.13216  -0.860  0.39497
nervous      0.49036    0.15300   3.205  0.00269 **
future       0.37748    0.34810   1.084  0.28485
anhedonia    0.21384    0.25910   0.825  0.41420
tired        0.13977    0.09411   1.485  0.14556
alone       -0.14978    0.12527  -1.196  0.23904
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5465 on 39 degrees of freedom
Multiple R-squared: 0.4921, Adjusted R-squared: 0.4009
F-statistic: 5.398 on 7 and 39 DF, p-value: 0.00022108
```

```r
# Glance fit
glance(fit) %>%
  select(
    adj_r_squared, CV, AIC, AICc, BIC, log_lik
  )
```

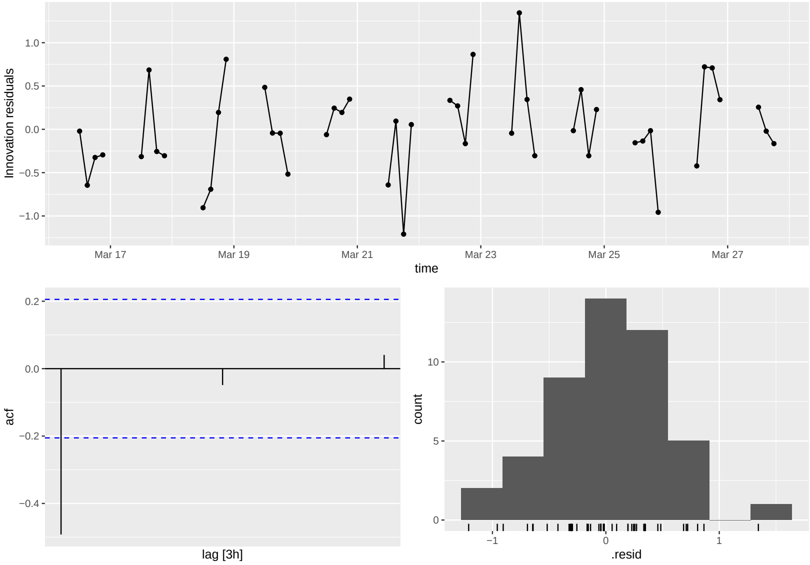
```
# A tibble: 1 x 6
  adj_r_squared    CV   AIC  AICc   BIC log_lik
          <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
1         0.401 0.395 -47.6 -42.7 -30.9   -33.9
```

```r
# Plot model
augment(fit) %>%
  # Plot quarter on x-axis
  ggplot(aes(x = time)) +
  # Plot actual values
  geom_line(aes(y = worry, colour = "Data")) +
  # Plot fit values
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(
    # No y-axis label
    y = NULL,
    # Change title
    title = "Worry across time",
    subtitle = "Participant 4"
  ) +
  # Change colors
  scale_colour_manual(
    values = c(
      Data = "black", # Make data line black
      Fitted = "orange" # Make fitted line orange
    )
  ) +
  # No title for legend
  guides(colour = guide_legend(title = NULL)) +
  scale_y_continuous(
    limits = c(1, 5), # minimum and maximum of y-axis
    breaks = seq(1, 5, 1) # breaks on y-axis
  )
```


Worry across time
Participant 4
5
4
3
2
1
Mar 17
Mar 19
Mar 21
Mar 23
Mar 25
Mar 27
time
Data
Fitted

```r
# Plot residuals
fit %>%
  gg_tsresiduals()
```

Forecasting | Model estimation



Box-Pierce

```r
# Plot residuals
fit %>%
  augment() %>%
  na.omit() %>%
  features(.resid, box_pierce, lag = 10)
```

```
# A tibble: 1 x 3
  .model bp_stat bp_pvalue
  <chr>    <dbl>     <dbl>
1 tslm      14.2     0.164
```

Ljung-Box

```
# Plot residuals
fit %>%
  augment() %>%
  na.omit() %>%
  features(.resid, ljung_box, lag = 10)
```

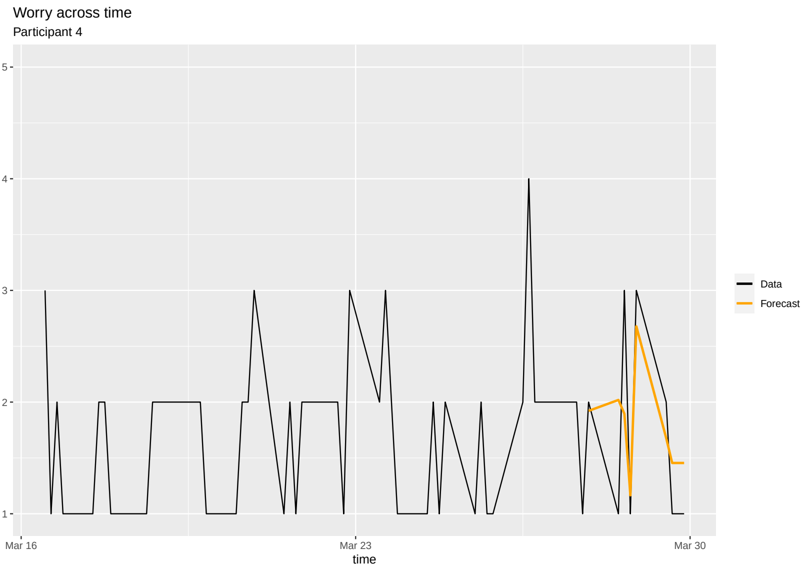
```
# A tibble: 1 x 3
  .model lb_stat lb_pvalue
  <chr>    <dbl>     <dbl>
1 tslm      17.1    0.0715
```

```r
# Make forecast
fc_actual <- fit %>%
  forecast(new_data = actual)
```

```r
# Peek at forecast
head(fc_actual)
```

```
# A fable: 6 x 11 [3h] <UTC>
# Key:     .model [1]
  .model time                          worry .mean relax irritable nervous future
  <chr>  <dttm>                       <dist> <dbl> <int>     <int>   <int>  <int>
1 tslm   2020-03-27 21:00:00 N(1.9, 0.32)  1.92     2         2       2      1
2 tslm   2020-03-28 00:00:00    N(NA, NA) NA       NA        NA      NA     NA
3 tslm   2020-03-28 03:00:00    N(NA, NA) NA       NA        NA      NA     NA
4 tslm   2020-03-28 06:00:00    N(NA, NA) NA       NA        NA      NA     NA
5 tslm   2020-03-28 09:00:00    N(NA, NA) NA       NA        NA      NA     NA
6 tslm   2020-03-28 12:00:00   N(2, 0.38)  2.02     1         1       2      1
# i 3 more variables: anhedonia <int>, tired <int>, alone <int>
```

```r
# Plot forecast
ts %>%
  # Plot quarter on x-axis
  ggplot(aes(x = time)) +
  # Plot actual values
  geom_line(aes(y = worry, colour = "Data")) +
  # Plot predicted values
  geom_line(
    data = na.omit(fc_actual),
    aes(y = .mean, colour = "Forecast"),
    size = 1
  ) +
  labs(
    # No y-axis label
    y = NULL,
    # Change title
    title = "Worry across time",
    subtitle = "Participant 4"
  ) +
  # Change colors
  scale_colour_manual(
    values = c(
      Data = "black", # Make data line black
      Forecast = "orange" # Make fitted line orange
    )
  ) +
  # No title for legend
  guides(colour = guide_legend(title = NULL)) +
  scale_y_continuous(
    limits = c(1, 5), # minimum and maximum of y-axis
    breaks = seq(1, 5, 1) # breaks on y-axis
  )
```


Worry across time
Participant 4
5
4
3
2
1
Mar 16
Mar 23
Mar 30
time
Data
Forecast

Point Estimates

R-squared

```
[1] 0.5085333
```

MAE

```
[1] 0.4901048
```

RMSE

```
[1] 0.6034543
```

MBE

```
[1] 0.03328566
```

```
# A tibble: 1 x 10
  .model .type        ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
  <chr>  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
1 tslm   Test   -0.0333 0.603 0.490 -17.7  34.6 0.659 0.611 -0.809
```

Distributional Estimates (see Section 5.9 of FPP3)

```
# Winkler and Continuous Ranked Probability Scale
fc_actual %>%
  accuracy(ts_fill, list(
    winkler = winkler_score,
    crps = CRPS,
    skill = skill_score(ME)
  ))
```

```
# A tibble: 1 x 5
  .model .type winkler  crps skill
  <chr>  <chr>   <dbl> <dbl> <dbl>
1 tslm   Test     2.37 0.344 0.734
```

- Winkler: penalizes estimates outside of interval proportional to distance from interval
- Continuous Ranked Probability Score (CRPS): average quantile scores over all time series values
- Skill score: scale-free comparison based on relative measure (e.g., RMSE, CRPS)

What if you want to forecast without actual data?

Random

x_i **drawn from** X_i
textbfbased on probability of x_i
across all time points

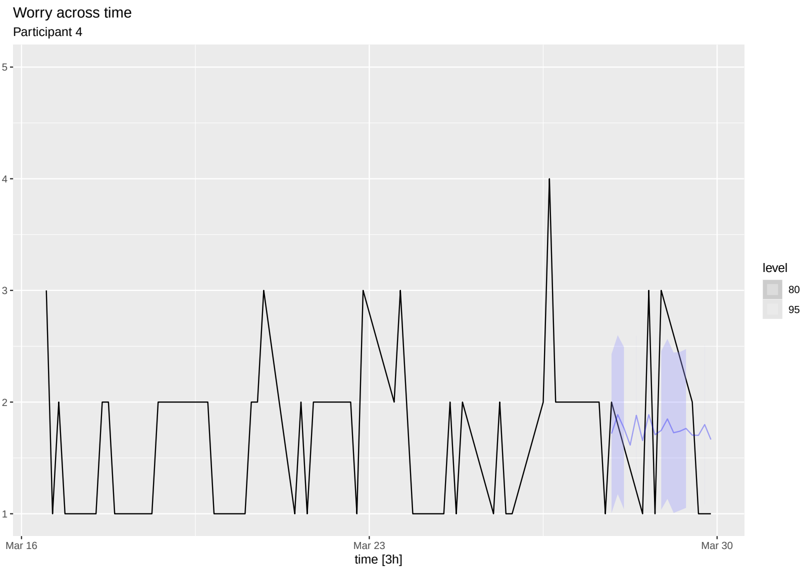
$$\boxed{x_{ih}} = \sum_{n=1}^{N} \frac{\boxed{x_i \in X_i}}{N}$$

predictor x_i **for forecast** h

What if you want to forecast without actual data?

```r
# Obtain useful functions
source("../Useful Functions/lm_new_data.R")

# Make future possibilities
future_scenarios <- scenarios(
  random = lm_new_data( # Random forecasts
    model = fit, # set model
    df = prediction, # set data
    iterations = 10, # number of iterations
    h = nrow(actual), type = "random"
  ),
  names_to = "Scenario")

# Make forecast
fc <- fit %>% forecast(new_data = future_scenarios)
```

```r
# Plot forecasts simultaneously
ts %>%
  autoplot(worry) +
  autolayer(fc, alpha = 0.333) +
  labs(
    # No y-axis label
    y = NULL,
    # Change title
    title = "Worry across time",
    subtitle = "Participant 4"
  ) +
  scale_y_continuous(
    limits = c(1, 5), # minimum and maximum of y-axis
    breaks = seq(1, 5, 1) # breaks on y-axis
  )
```



Worry across time
Participant 4
5
4
3
2
1
Mar 16
Mar 23
Mar 30
time [3h]
level
80
95

```r
# Obtain metrics
random_scenario <- fc[
  fc$Scenario == "random",
]
random <- random_scenario[
  !is.na(match(random_scenario$time, actual$time)),
]$.mean
```

```r
# Make data frame table
df_table <- data.frame(
  "Measure" = c(
    "R-squared", "MAE",
    "RMSE", "MBE"
  ),
  "Random" = c(
    cor(random, actual$worry, use = "pairwise")^2, # R-squared
    mean(abs(random - actual$worry), na.rm = TRUE), # MAE
    sqrt(mean((random - actual$worry)^2, na.rm = TRUE)), # RMSE
    mean(random - actual$worry, na.rm = TRUE) # MBE
  )
)

# Print data frame
df_table
```

```
     Measure      Random
1 R-squared  0.60807556
2       MAE  0.70948627
3      RMSE  0.77819465
4       MBE -0.02694606
```

```
  .model .type          ME      RMSE       MAE       MPE     MAPE      MASE
1   tslm  Test -0.03328566 0.6034543 0.4901048 -17.72087 34.61818 0.6585783
      RMSSE       ACF1
1 0.6105961 -0.8089913
```

Forecasting | Quantify forecast (no actual data)

```r
# Make data frame table
df_table <- rbind.data.frame(
  random = fc[fc$Scenario == "random",] %>%
  accuracy(ts_fill, list(
    winkler = winkler_score,
    crps = CRPS
  )),
  actual = fc_actual %>%
  accuracy(ts_fill, list(
    winkler = winkler_score,
    crps = CRPS
  ))
)

# Report table
df_table$.type <- c("random", "actual")
df_table
```

```
# A tibble: 2 x 4
  .model .type  winkler  crps
  <chr>  <chr>    <dbl> <dbl>
1 tslm   random    3.12 0.480
2 tslm   actual    2.37 0.344
```

In your assignment, you will use generative AI to come up with alternative methods of creating new data for the TSLM to forecast on

Exploring Time Series

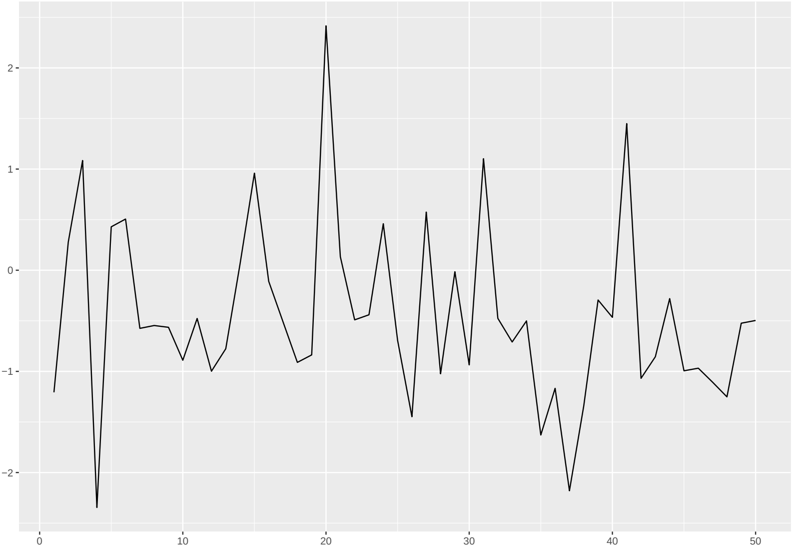
Exploring Time Series

Is there a trend or seasonal component to the time series?

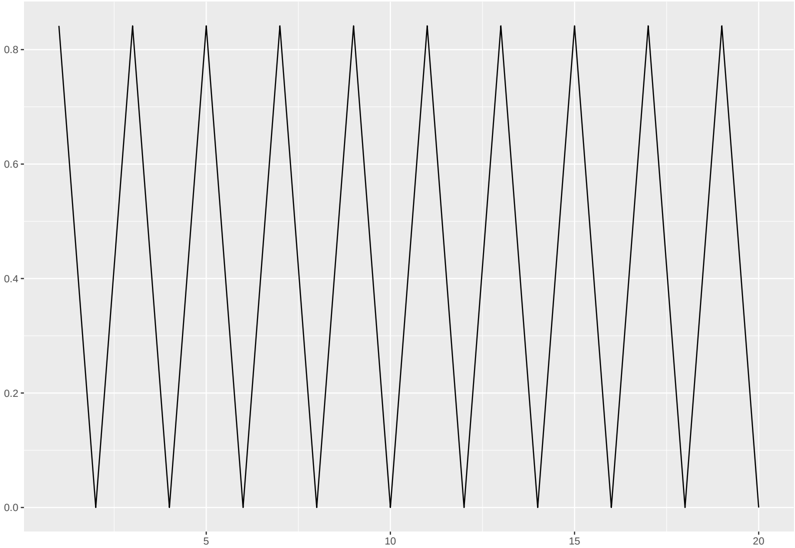
Trend? Seasonal?

- *trend*: long-term increase or decrease in the time series (does not need to be linear)
- *seasonal*: time series is affected by seasonal factors (day of week, time of year)
- *cyclic*: rises and falls that are not on a fixed frequency (economic conditions, business cycle)

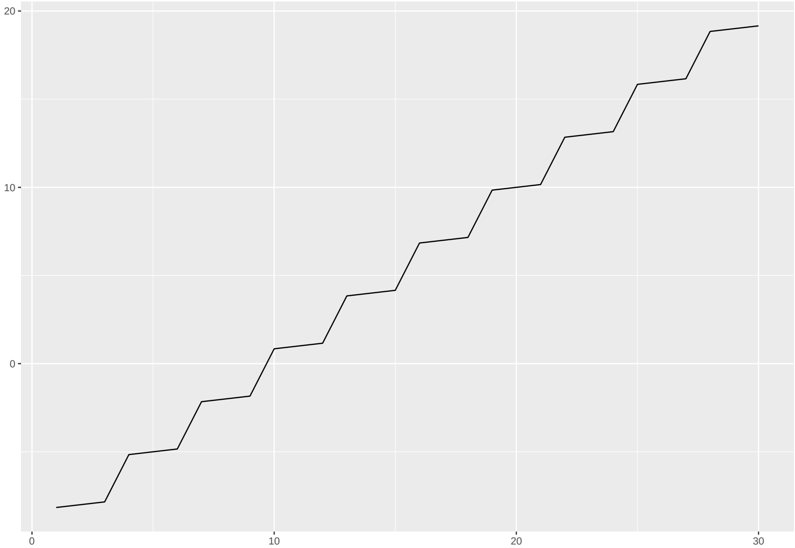
Trend? Seasonal?



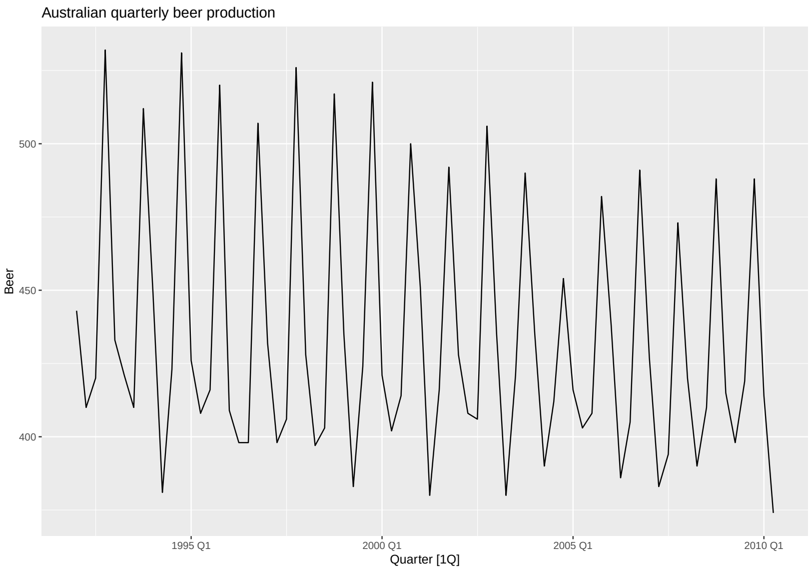
Trend? Seasonal?



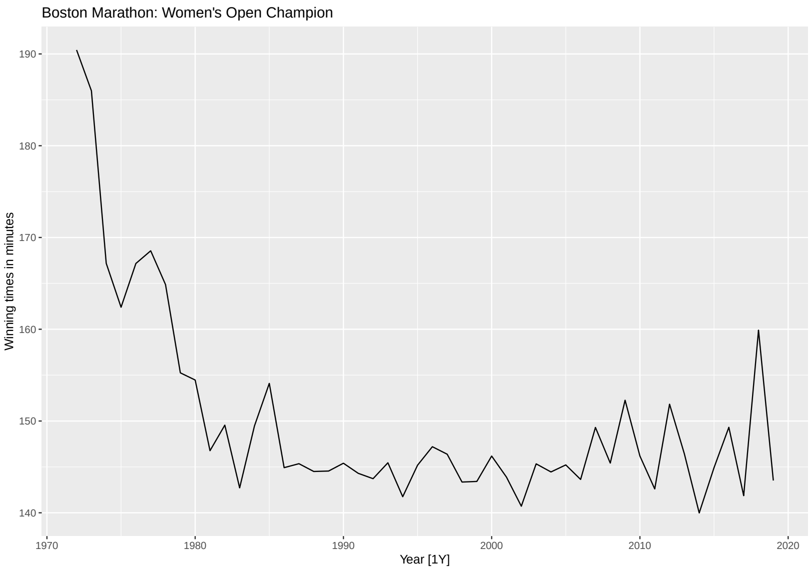
Trend? Seasonal?



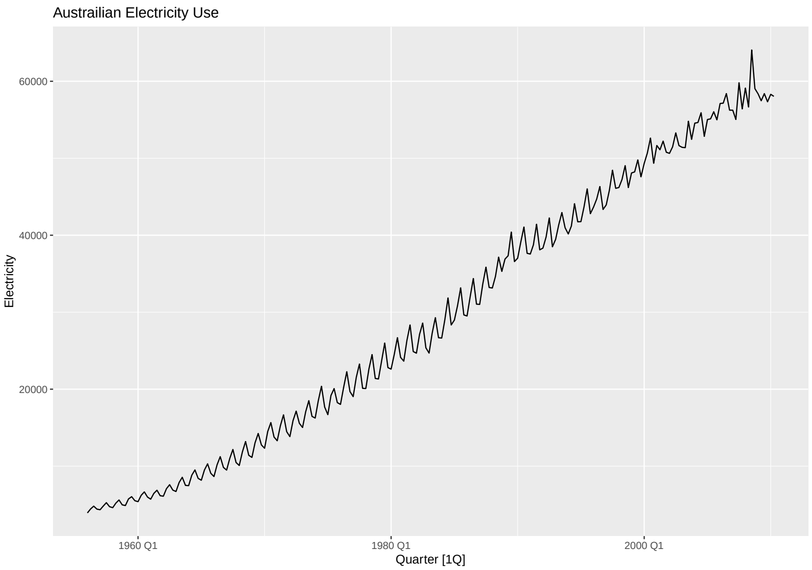
Trend? Seasonal?



Trend? Seasonal?



Trend? Seasonal?



Decomposition

How can we know for certain?

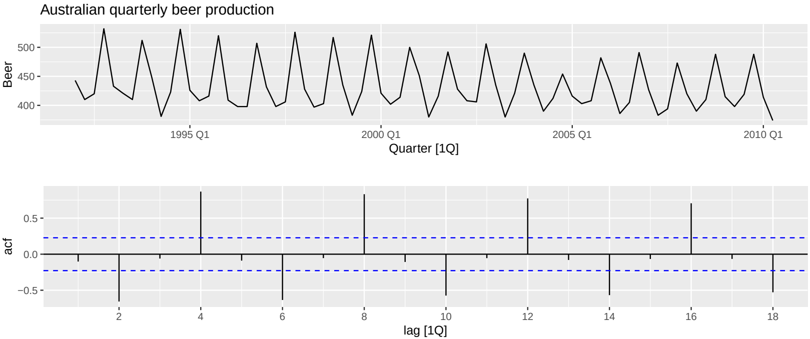
- Examining autocorrelations can provide some inference into whether the data has a trend or seasonal pattern

autocorrelation for lag *k*

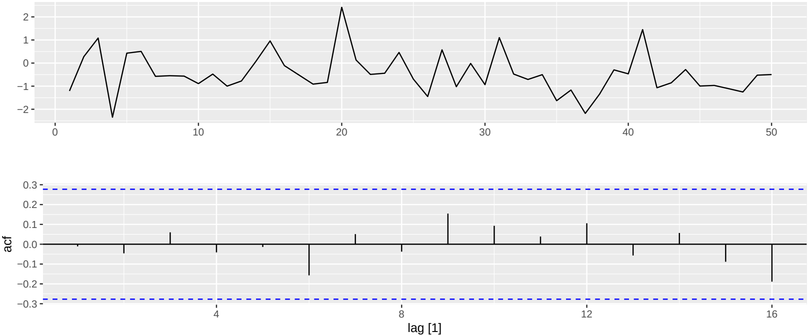
?

$$\boxed{r_k} = \frac{\sum_{t=k+1}^{T} \boxed{(y_t - \bar{y})(y_{t-k} - \bar{y})}}{\sum_{t=1}^{T} \boxed{(y_t - \bar{y})^2}}$$

?



Australian beer production

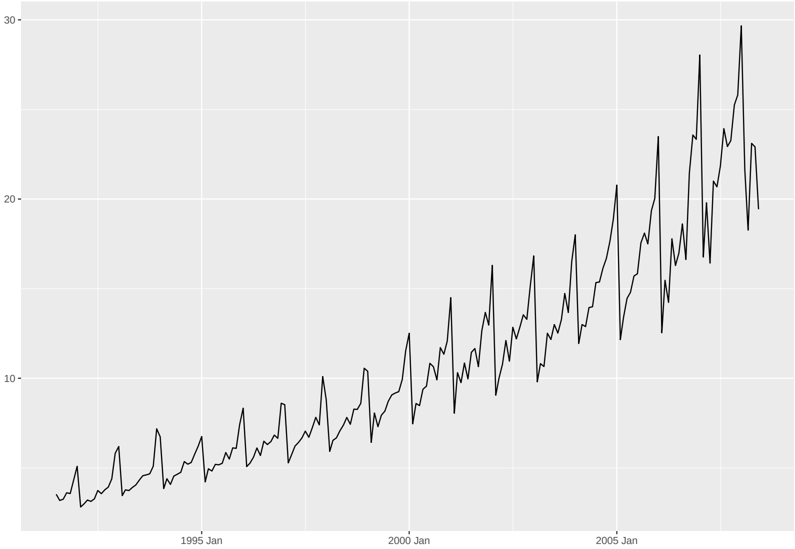


White noise

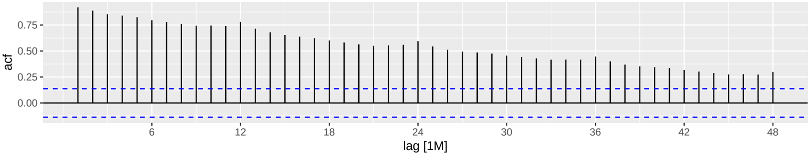
Austrailian antidiabetic drug sales

```
# Antidiabetic
a10 %>% # data
  as_tsibble() %>% # convert to `tsibble` format
 autoplot(value) +
  labs(x = "", y = "")
```

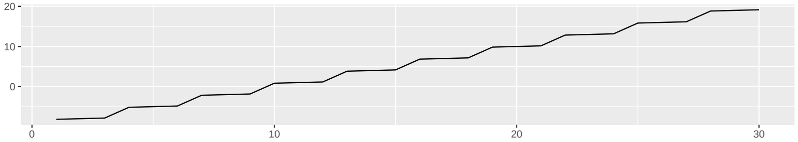
Decomposition | Autocorrelations

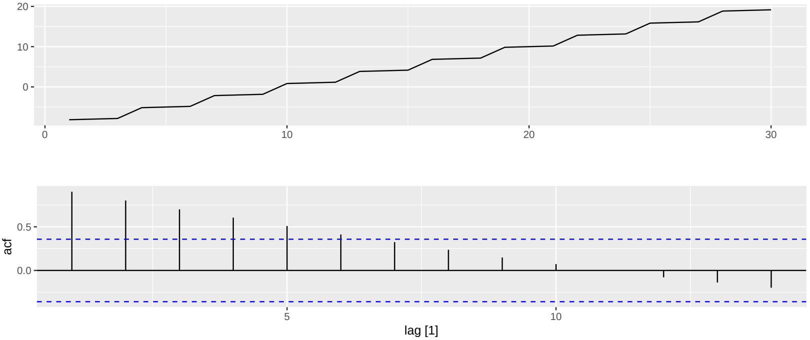


```r
# Antidiabetic
a10 %>% # data
  as_tsibble() %>% # convert to `tsibble` format
  ACF(
    value, # sales from `tsibble`
    lag_max = 48 # maximum lag
  ) %>%
  autoplot()
```



Austrailian antidiabetic drug sales


20
10
0
0
10
20
30



Linear trend with sine wave

Decomposition

Decomposition | Additive

outcome (at time *t*)

season

$$\underbrace{y_t}_{\text{outcome (at time } t)} = \underbrace{T_t}_{\text{trend}} + \underbrace{S_t}_{\text{season}} + \underbrace{R_t}_{\text{remainder}}$$

trend

remainder

- More appropriate if seasonal fluctuations do not vary with level (average of period)
- The model you'll use most often

$$\overbrace{y_t}^{\text{outcome (at time } t\text{)}} = \underbrace{T_t}_{\text{trend}} \times \overbrace{S_t}^{\text{season}} \times \underbrace{R_t}_{\text{remainder}}$$

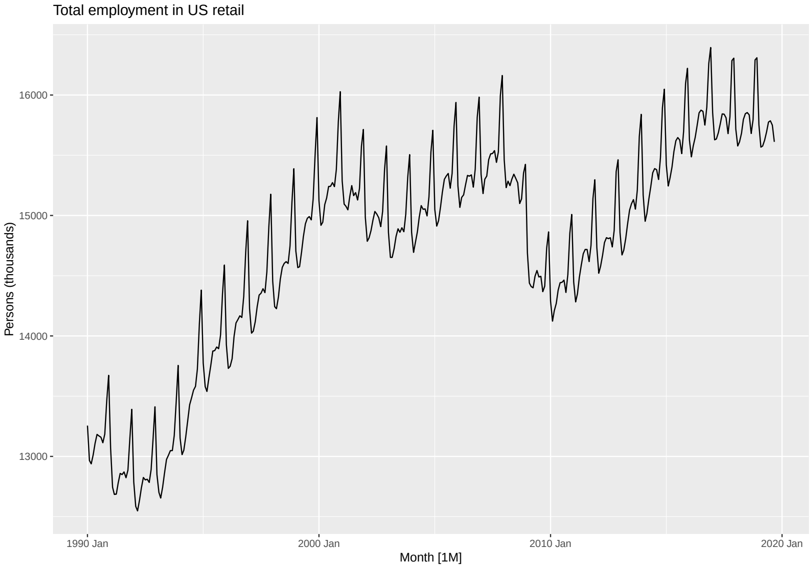
- More appropriate if seasonal fluctuations are proportional with level (average of period)
- More common with economic series
- Can be made into additive relationship with log-transformation (i.e., $\log_{y_t} = \log S_t + \log T_t + \log R_t$)

Decomposition with STL

Decomposition | STL

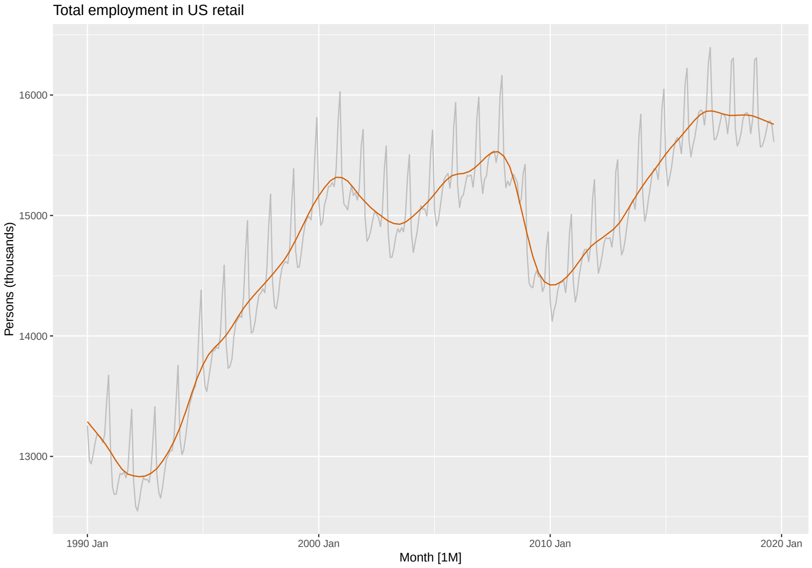
- **S**easonal and **T**rend decomposition using **L**oess (STL)
- Good general decomposition method
- Mainly uses additive decomposition (use log for multiplicative)
- Handles any type of seasonality
- Robust to outliers

```r
# Select US retail data
us_retail_employment <- us_employment %>%
  filter(year(Month) >= 1990, Title == "Retail Trade") %>%
  select(-Series_ID)

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


Total employment in US retail
Persons (thousands)
16000
15000
14000
13000
1990 Jan
2000 Jan
2010 Jan
2020 Jan
Month [1M]

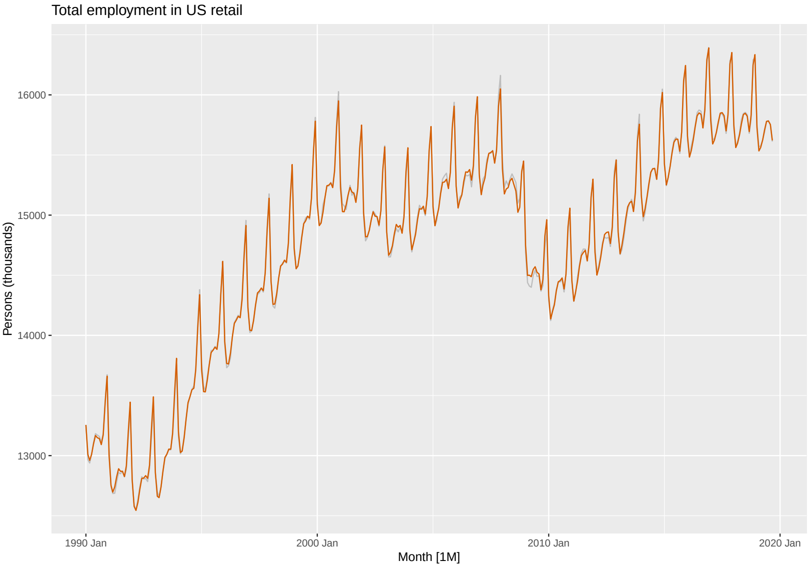
```r
# Store components
us_comps <- us_retail_employment %>%
  model(stl = STL(Employed))

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

Decomposition | STL Trend



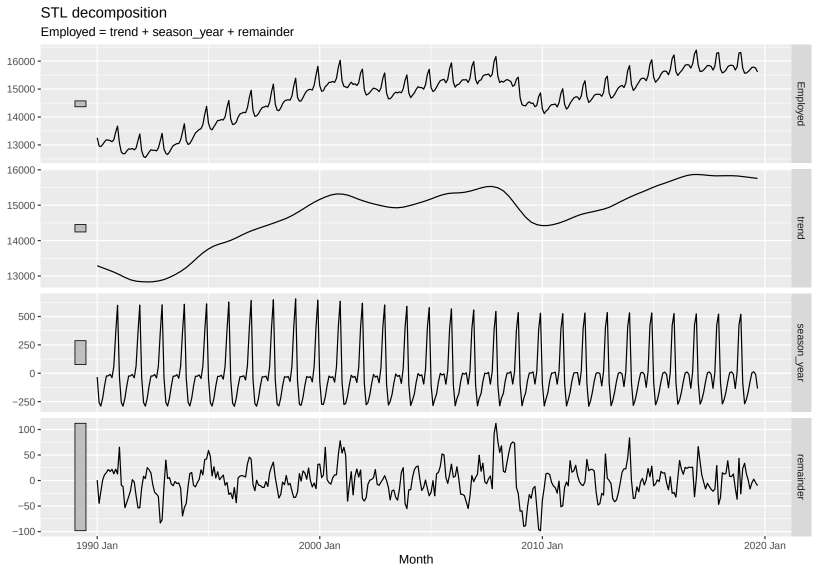
```
# STL Trend
us_retail_employment %>%
  autoplot(Employed, color = 'gray') +
  autolayer(
    components(us_comps),
    trend + season_year, # plot trend + seasonlity
    color = '#D55E00'
  ) +
  labs(
    y = "Persons (thousands)",
    title = "Total employment in US retail"
  )
```

Decomposition | STL Trend + Season



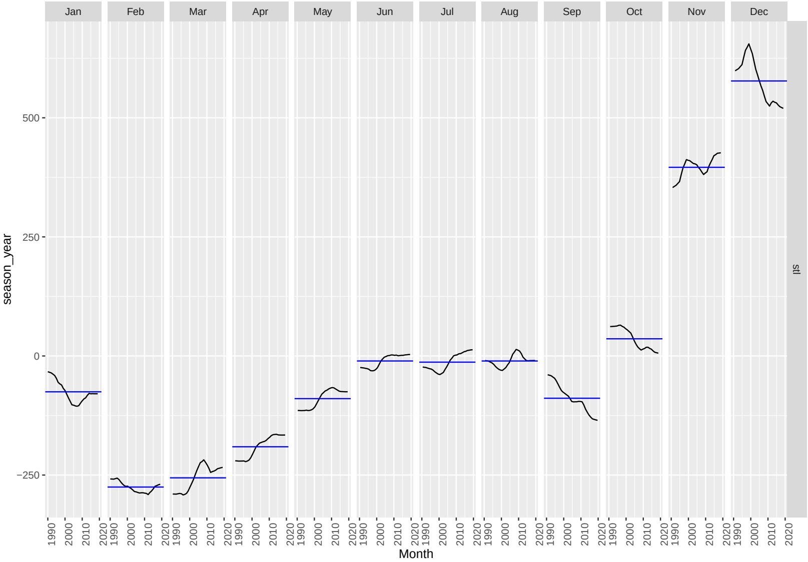
```r
# STL decomposition
us_retail_employment %>% # dataset
  model(stl = STL(Employed)) %>% # model (STL)
  components() %>% # components of decomposition
  autoplot() # plot
```

Decomposition | STL



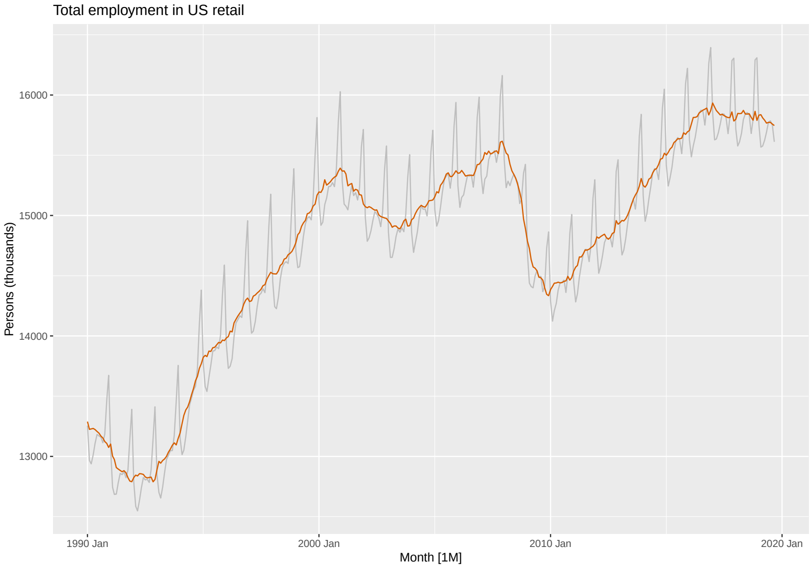
```r
# Monthly
us_retail_employment %>% # dataset
  model(stl = STL(Employed)) %>% # model (STL)
  components() %>% # components of decomposition
  gg_subseries(season_year) # broken down by month
```

Decomposition | STL by Month



```r
# STL Season Adjustment
us_retail_employment %>%
  autoplot(Employed, color = 'gray') +
  autolayer(
    components(us_comps),
    season_adjust, # plot season adjustment
    color = '#D55E00'
  ) +
  labs(
    y = "Persons (thousands)",
    title = "Total employment in US retail"
  )
```

Decomposition | STL with Seasonal Adjustment



- Adjustments are based on past values to adjust current value
- Adjusted series reflect trend and remainders (error)

- `trend(window = nextodd(ceiling((1.5*period) / (1-(1.5/s.window)))))`: controls smoothness of trend (should be odd)
- `season(window = 13)`: controls variation of season
- `season(window = "periodic")`: “infinite” window
- `robust`: boolean (`TRUE` for robust estimates)

X Methods

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

X-11

Advantages

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

X-11

Advantages

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

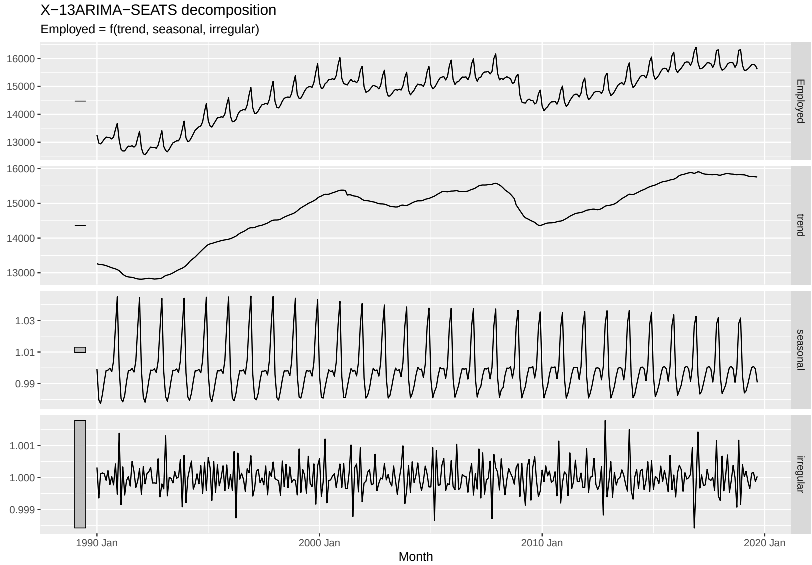
Disadvantages

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

X-13ARIMA-SEATS

- Mainly developed for economic data
- Trend and seasonal data only
- Allows seasons to change across time
- Allows adjustments for explanatory variables
- Outliers can be omitted
- Missing values can be estimated and replaced
- Holidays can be estimated

```r
# X-13ARIMA-SEATS
us_retail_employment %>% # data
  model(X_13ARIMA_SEATS(Employed)) %>%  # X13 decomposition
  components() %>% # get components from decomposition
  autoplot() # plot decomposition
```


X−13ARIMA−SEATS decomposition
Employed = f(trend, seasonal, irregular)
16000
15000
14000
13000
Employed
16000
15000
14000
13000
trend
1.03
1.01
0.99
seasonal
1.001
1.000
0.999
irregular
1990 Jan
2000 Jan
2010 Jan
2020 Jan
Month

```r
# X-13ARIMA-SEATS
us_retail_employment %>%
  model(X_13ARIMA_SEATS(Employed)) %>%
  report()
```

```
Series: Employed
Model: X-13ARIMA-SEATS

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
Easter[15]         0.0008935  0.0002984   2.994  0.00275 **
LS2001.Apr        -0.0085821  0.0019577  -4.384 1.17e-05 ***
LS2008.Nov        -0.0079473  0.0019389  -4.099 4.15e-05 ***
AR-Nonseasonal-01  0.9284818  0.0344282  26.969  < 2e-16 ***
MA-Nonseasonal-01  0.7478066  0.0600573  12.452  < 2e-16 ***
MA-Seasonal-12     0.5187352  0.0476718  10.881  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

SEATS adj.  ARIMA: (1 1 1)(0 1 1)  Obs.: 357  Transform: log
AICc:  3422, BIC:  3449  QS (no seasonality in final):    0
Box-Ljung (no autocorr.): 18.26   Shapiro (normality): 0.983 ***
```