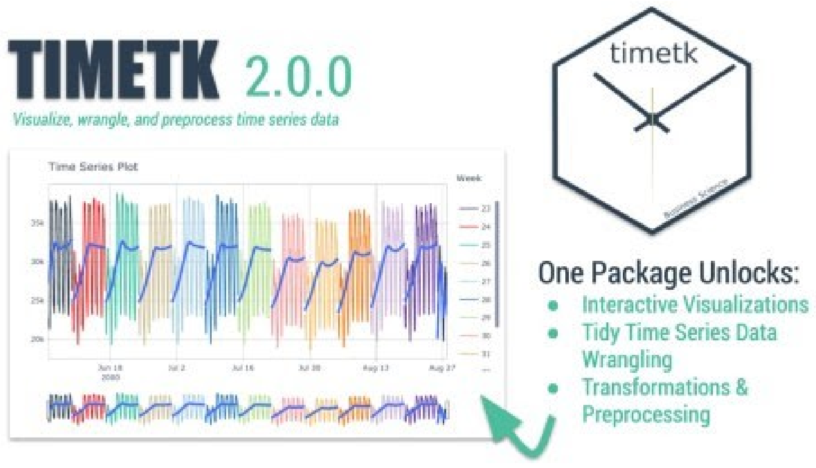
Background

Time series has been a passion project for me since my days of forecasting sales and economic data for a manufacturing company I worked for. My one gripe has always been that I had to use 50 different packages (`zoo`, `xts`, `dplyr`, etc) made by 50 different people to perform common data wrangling and visualization analyses. `timetk` solves this problem by making a consistent approach to visualize, wrangle, and preprocess time series data inside the `tidyverse` and `tidymodels` ecosystem.

With advancements in tidy-time series, the combination of the `tidyverse` and time series is an amazingly powerful concept. I'm not the first one to think of this idea. In fact, Davis Vaughan created `tibbletime` and the "tidyverts" (Rob Hyndman, Earo Wang, and Mitchell O'Hara-Wild) have created a whole forecasting and data wrangling system using a `tsibble` data structure.

These are amazing packages, but they solve different needs. `tibbletime` focused on data wrangling. The `tidyverts` focused on forecasting at scale using ARIMA and company.


TIMETK 2.0.0
Visualize, wrangle, and preprocess time series data
Time Series Plot
Week
timetk
One Package Unlocks:
Interactive Visualizations
Tidy Time Series Data Wrangling
Transformations & Preprocessing

Visualize, wrangle, and preprocess time series data

My needs are different. I need:

- **Interactive visualizations** for easy data exploration
- **Time series data wrangling** for doing time series summarization, filtering, padding, and simple date-based arithmetic
- **Transformations and preprocessing** that fit into the NEW `tidymodels` ecosystem (so I can do *Time Series Machine Learning* in addition to ARIMA forecasting)

So I created `timetk` version 2.0.0 to solve these needs.

Here's the new *mission*, and what you can do *in 1-line of code* with `timetk` >= 2.0.0.

Mission

To make it easy to ***visualize, wrangle and preprocess time series data*** for forecasting and machine learning prediction.

What Can You Do in 1-Line of Code?

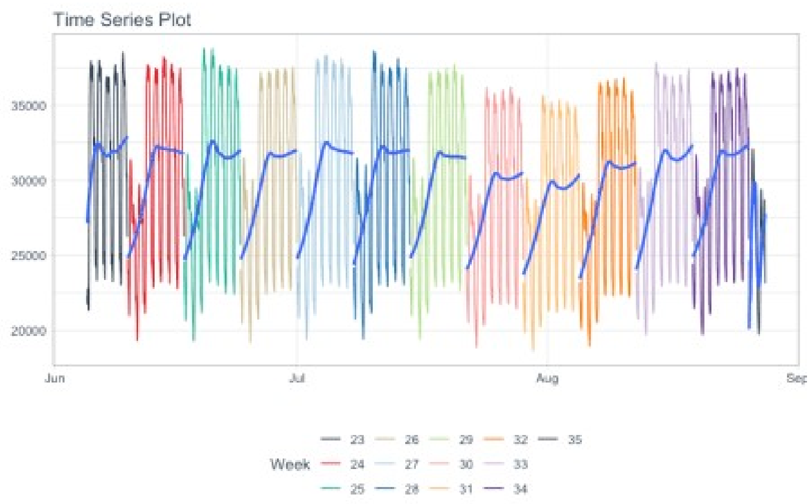
First step, load these R packages.

```
library(tidyverse)
library(lubridate)
library(timetk)
```

Investigate a time series…

This is fun! In 1 line of code we can visualize a dataset.

```
# Static ggplot
taylor_30_min %>%
    plot_time_series(date, value, .color_var = week(date),
                     .interactive = FALSE, .color_lab = "Week")
```



So what did we just do?

We are exploring `taylor_30_min`, which is a classic electricity-demand time series that comes from one of my all-time-favorite packages, `forecast`. It's been updated to the `tibble` structure with a time stamp column called "date" and a value column called "value".

We can plotted it using `plot_time_series()`. I set `.interactive = FALSE` to return a `ggplot` (I'll do that for all of the visualizations in this tutorial). The default is to return an interactive `plotly` graph, which is great for `shiny` apps, `rmarkdown` HTML documents, and super powerful for exploring time series (zooming, panning, etc).

Want the an interactive `plotly` visualization?

Just try this code and explore the `plotly` visualization.

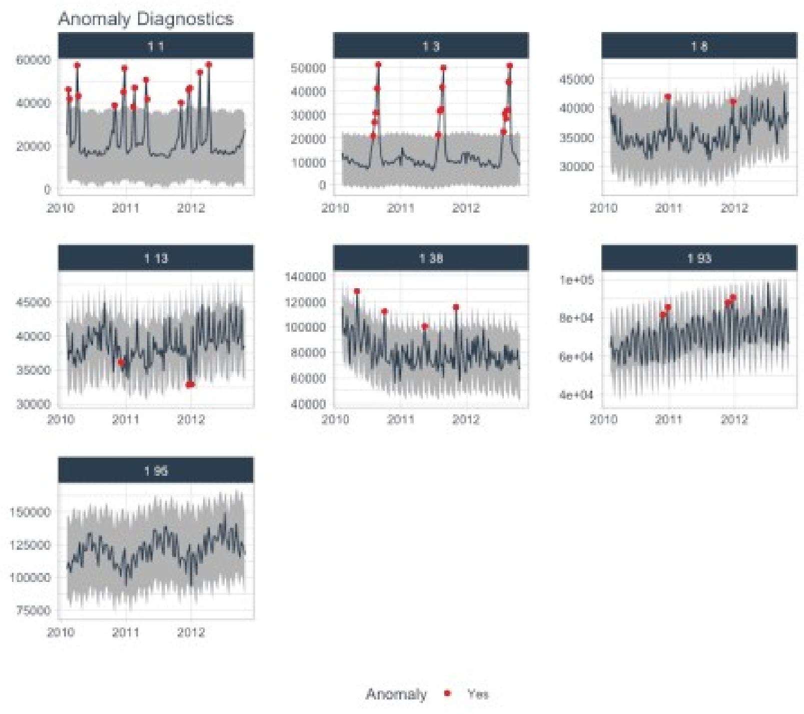
```
# INTERACTIVE Plotly
taylor_30_min %>%
    plot_time_series(date, value, .color_var = week(date),
                     .interactive = TRUE, .plotly_slider = TRUE,
.color_lab = "Week")
```

Let's pick up the pace. Here's some more amazing visualization capabilities!

Visualize anomalies…

We can visualize anomalies for multiple time series groups. Here we use `group_by()` to group the time series. Note this is a different dataset, `walmart_sales_weekly`.

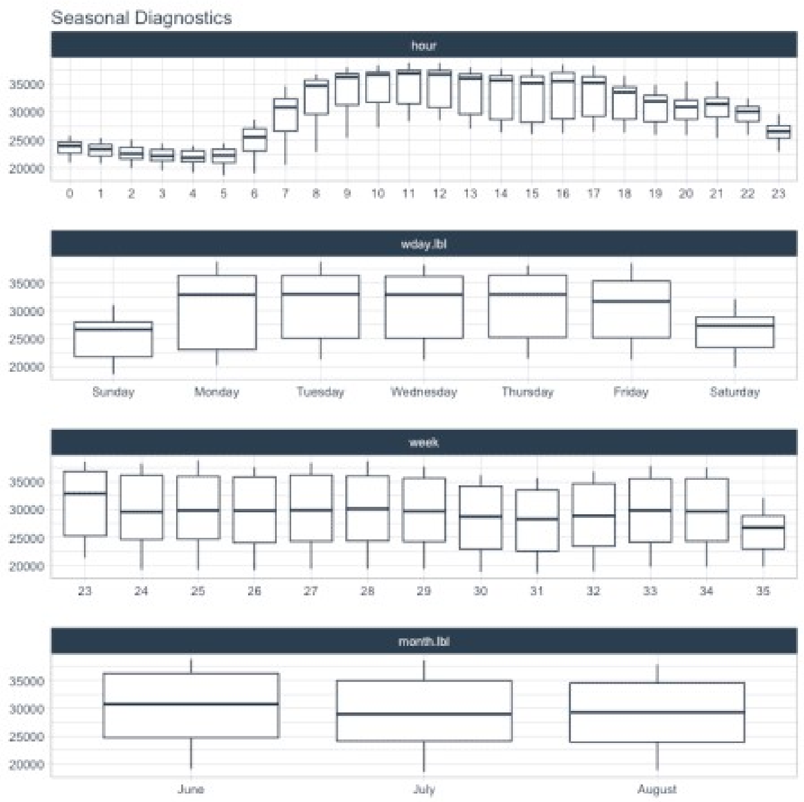
```
walmart_sales_weekly %>%
    group_by(Store, Dept) %>%
    plot_anomaly_diagnostics(Date, Weekly_Sales,
                             .facet_ncol = 3, .interactive = FALSE)
```



Make a seasonality plot…

We can get seasonality plots.

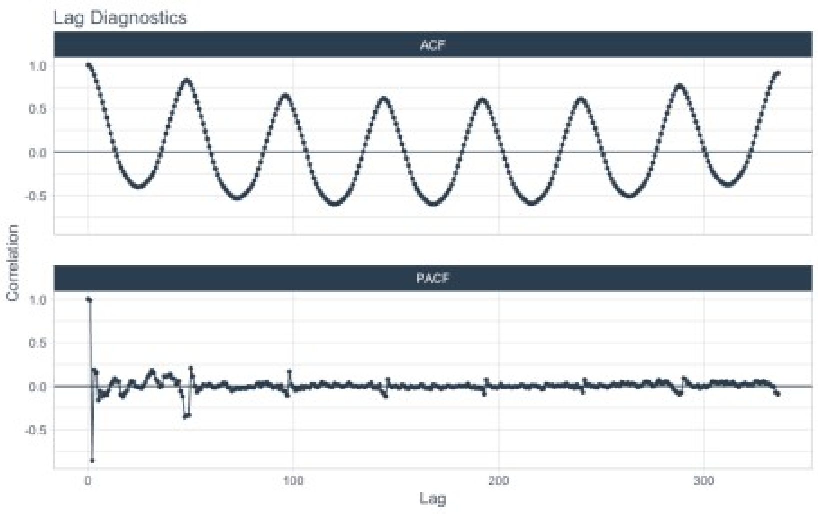
```
taylor_30_min %>%
    plot_seasonal_diagnostics(date, value, .interactive = FALSE)
```



Inspect autocorrelation, partial autocorrelation (and cross correlations too)…

And we can search the Autocorrelation and Partial Autocorrelation.

```
taylor_30_min %>%
    plot_acf_diagnostics(date, value, .lags = "1 week", .interactive =
  FALSE)
```



Lag Diagnostics
ACF
PACF
Correlation
Lag
1.0
0.5
0.0
-0.5
0
100
200
300