DATA SCIENCE

Predicting Chicago Taxi Trips with R Time Series Model – BSTS

Step-by-step tutorial on how to forecast number of tax series model

Lea Wu
Jun 4, 2024 9 min read

Photo by Waldemar on Unsplash

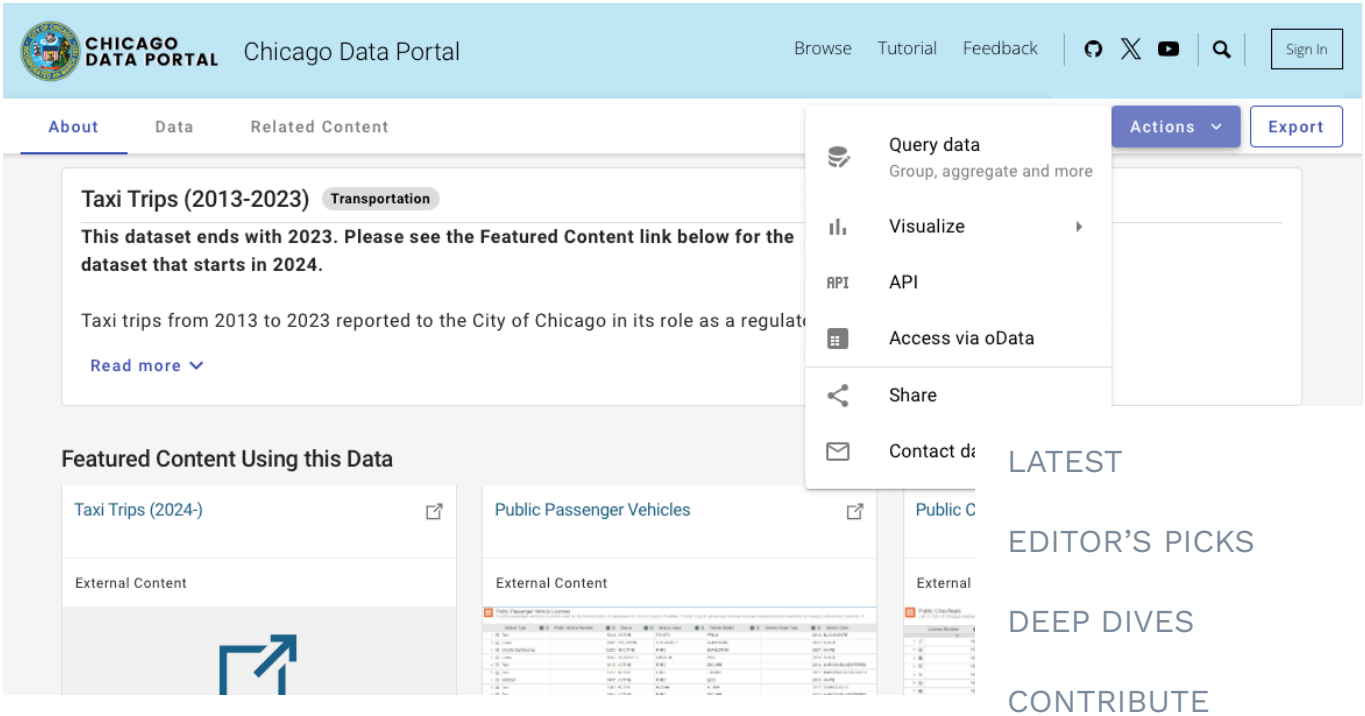
Introduction

Imaging you're planning marketing strategies for your taxi company or even considering market entry as a new competitor – predicting number of taxi trips in the big cities can be an interesting business problem. Or, if you're just a curious resident like me, this article is perfect for you to learn how to use the R Bayesian Structural Time Series (BSTS) model to forecast daily taxi trips an fascinating insights.

In this article, I will walk you through the pipeline preparation, explorational data analysis, time serie forecast result analysis, and business insight. I ai daily number of trips for the second half year of 2

Data Preparation

The data was acquired from the Chicago Data Por this platform with access to various government website, simply find the "Action" drop down list to



Chicago Data Portal provides query tool to acquire select

Within the query tool, you'll find filter, group, and tools. You can simply download the raw dataset. H computing complexity, I grouped the data by pick aggregate the count of trips per 15 minutes.

> With exploration of the dataset, I also filtered c 0 trip miles and N/A pickup area code (which m location is not within Chicago). You should expl decide how you would like to query the data. It the use case of your analysis.

Then, export the processed data. Downloading co !

Exploratory Data Analysis

Understanding the data is the most crucial step t and model choices reasoning. In the following par

different characteristics of the dataset including seasonality, trend, and statistical test for stationary and autocorrelation in the lags.

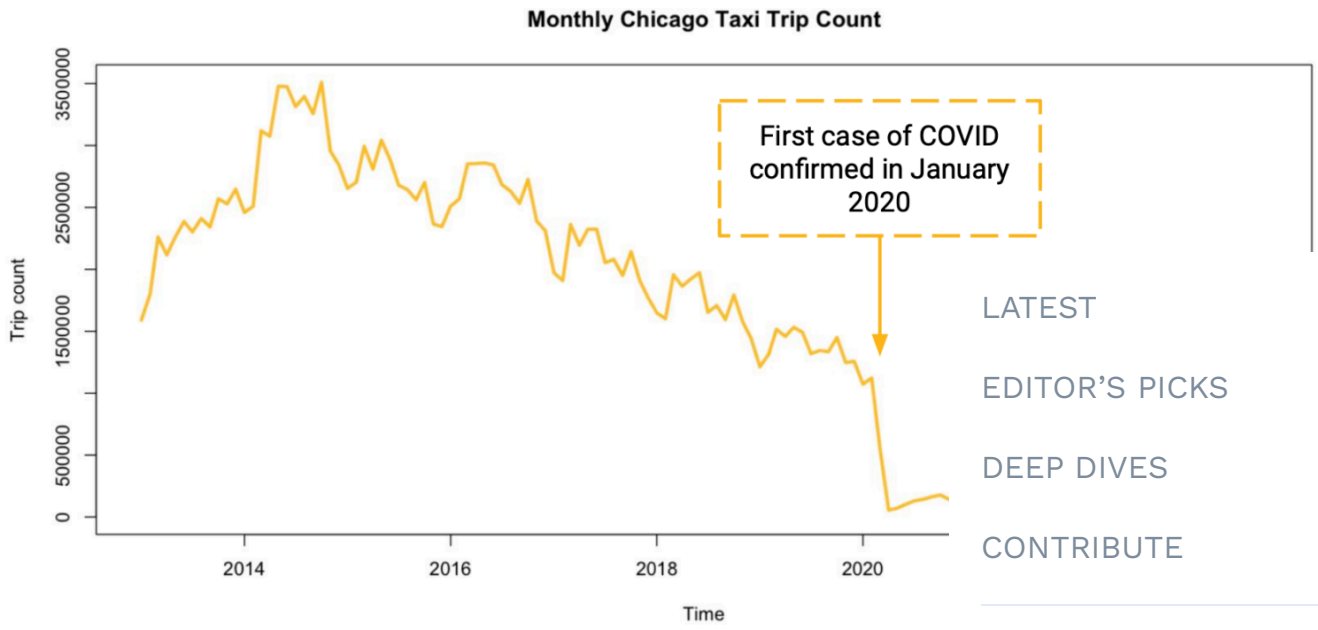
Seasonality refers to periodic fluctuations in data that occur at regular intervals. These patterns repeat over a specific period, such as days, weeks, months, or quarters.

To understand the seasonality, we first aggregate and month to visualize the effect.


```
 library(lubridate)
library(dplyr)
library(xts)
library(bsts)
library(forecast)
library(tseries)

demand_data <- read.csv("taxi_trip_data.csv")
colnames(demand_data) <- c('trip_cnt','trip_datet
demand_data$trip_datetime <- mdy_hms(demand_data$
demand_data$rounded_day <- floor_date(demand_data
demand_data$rounded_month <- floor_date(demand_da

monthly_agg <- demand_data %>%
    group_by(rounded_month) %>%
    summarise(
        trip_cnt = sum(trip_cnt, na.rm = TRUE)
)

daily_agg <- demand_data %>%
    group_by(rounded_day) %>%
    summarise(
        trip_cnt = sum(trip_cnt, na.rm = TRUE)
)
```

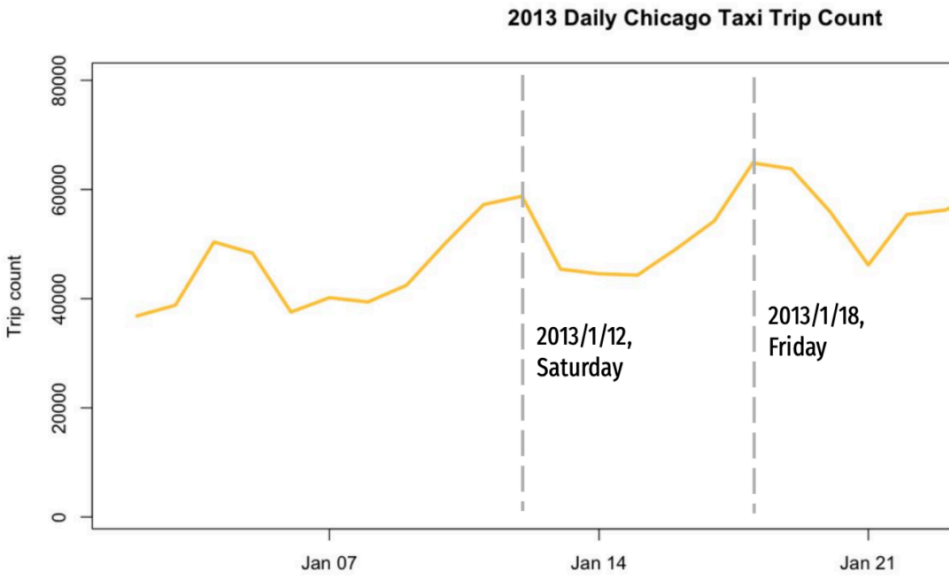
The taxi demand in Chicago peaked in 2014, show trend with annual seasonality, and was drastically

in 2020.



Monthly counts of taxi trips | Graph by author

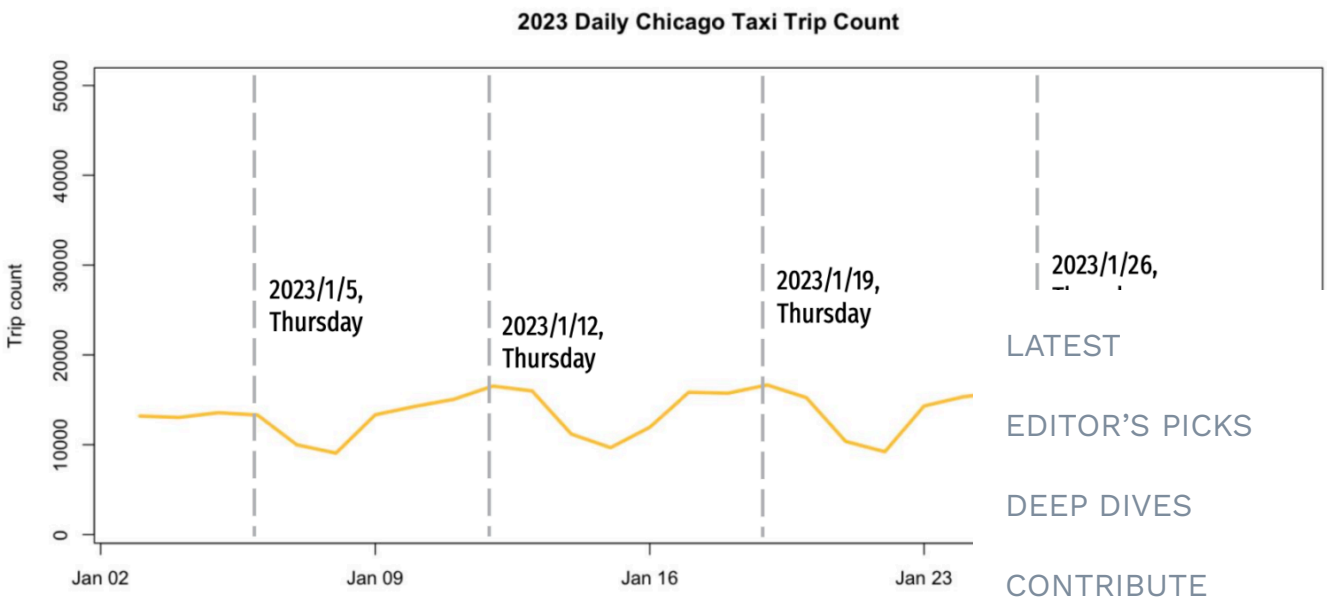
Daily count before COVID suggests weekly season trip numbers on Fridays and Saturdays.



Daily counts of taxi trips | Graph by author

Interestingly, the post-COVID weekly seasonality h Thursdays now have the highest demand. This pro

about COVID intervention.

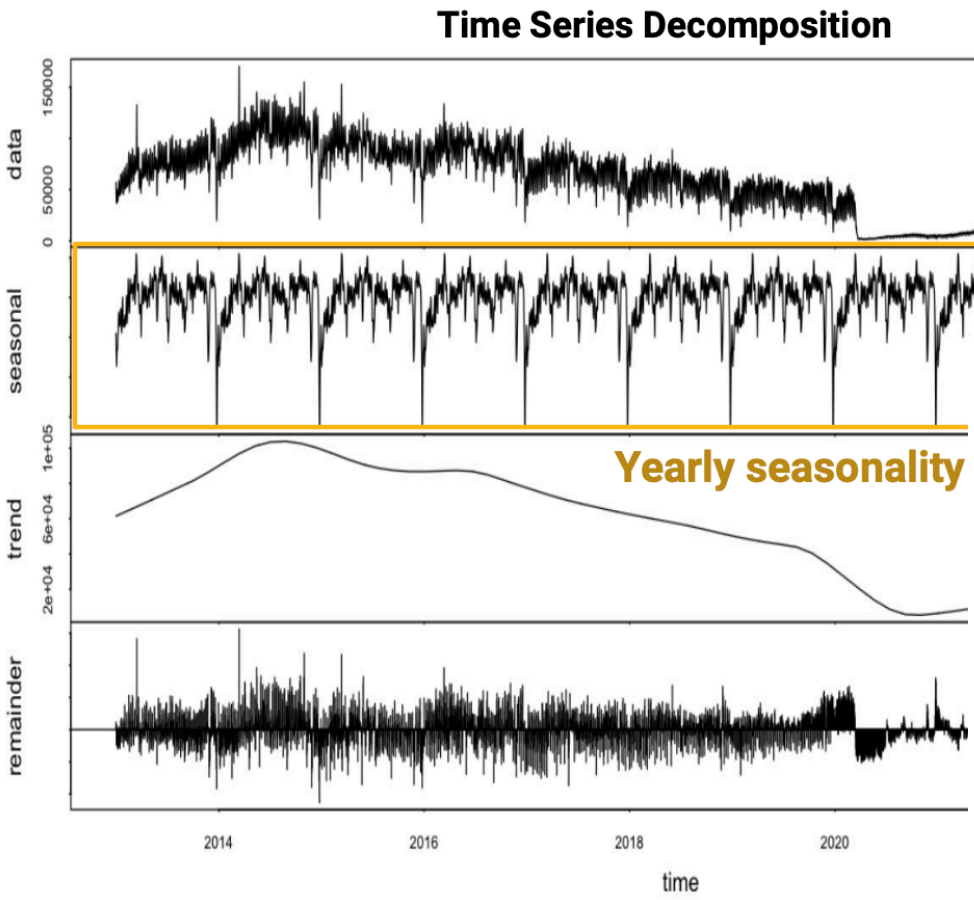


Daily counts of taxi trips – post COVID | Graph by aut

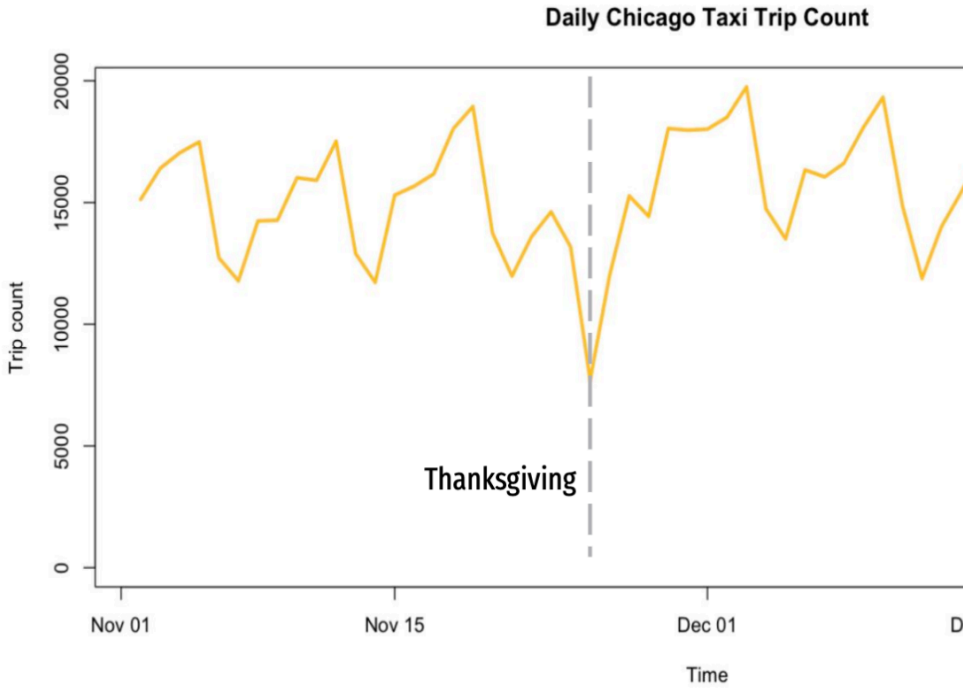
Trend in time series data refers to underlying patt the data to increase, decrease, or remain stable c transformed the dataframe to a time series data decomposition to monitor the trend.

```
zoo_data <- zoo(daily_agg$trip_cnt, order.by = da
start_time <- as.numeric(format(index(zoo_data)[1
ts_data <- ts(coredata(zoo_data), start = start_t
stl_decomposition <- stl(ts_data, s.window = "per
plot(stl_decomposition)
```

The result of STL composition shows that there's The seasonal part also shows a yearly seasonality to the yearly seasonality, I found that Thanksgivin have the lowest demand every year.



STL decomposition | Graph by author



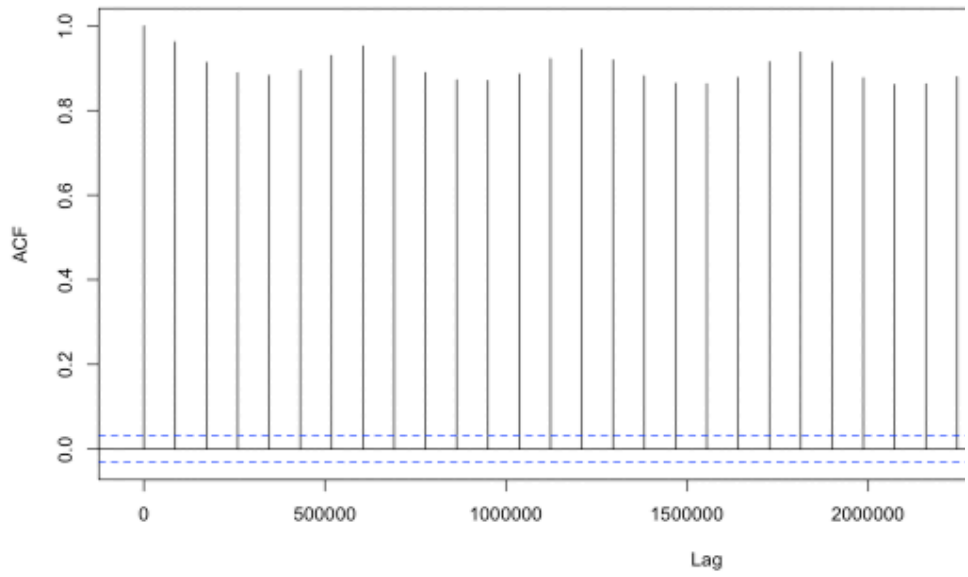
Daily counts of taxi trips for holidays | Graph by autl

A time series is considered **stationary** if its statistical properties (e.g. mean, variance, and autocorrelation) remain constant over time. From the above graphs we already know this data is not stationary since it exhibits trends and seasonality. If you would like to be more robust, ADF and KPSS test are usually leveraged to validate the null hypothesis of non-stationary and stationary respectively.

```
adf.test(zoo_data)
kpss.test(zoo_data)
```

Lag autocorrelation measures the correlation bet
and its lagged over successive time intervals. It e
current value is related to its past values. By autc
lags we can identify patterns and help us select a
series models (For example, understanding the au
structure helps determine the order of AR and M
ARIMA model). The graph shows significant autoc
lags.

```
acf(zoo_data)
```



ACF result | Graph by author

Data Transformation

The EDA provided crucial insights into how we should transform and preprocess the data to achieve the best forecasting results.

COVID changed the time series significantly. It is u
include data that had changed so much. Here I fit
from 2020 June to 2023 June. This still remains a
predicting numbers for the second half of 2023.

```
train <- window(zoo_data, start = as.Date("2020-0
test <- window(zoo_data, start = as.Date("2023-07
```

The non-stationary data shows huge variance and
Here I applied log and differencing transformatior
effect of these characteristics on the predicting p

```
train_log <- log(train + 1)
train_diff <- diff(train, differences = 1)
```

The following code operates on the log-transform
yielded better forecasting performance during pre

Choosing and Designing the Model

Let's quickly recap the findings from the EDA:

1. Multiple Seasonality and Non-Linear Trend

2. Impact of Holidays and Events: Significant events like holidays affect taxi demand.
3. Long Prediction Horizon: We need to forecast for 180 days.

Given these characteristics, the Bayesian Structural Time Series (BSTS) model is a suitable choice. The BSTS model decomposes a time series into multiple components using Bayes
capturing the underlying latent variables that evol
key components typically include:

1. Trend Component
2. Seasonal Component
3. Regressor Component: Incorporates the influe variables that might affect the time series.

This is the model I used to predict the taxi trips:

```
 ss <- AddSemilocalLinearTrend(list(), train_log)
ss <- AddSeasonal(ss, train_log, nseasons = 7)
ss <- AddSeasonal(ss, train_log, nseasons = 365)
ss <- AddMonthlyAnnualCycle(ss, train_log)
ss <- AddRegressionHoliday(ss, train_log, holiday
model_log_opti <- bsts(train_log, state.specifica
summary(model_log_opti)
```

AddSemilocalLinearTrend() From the EDA, the tre not a random walk. Therefore, we use a semi-loca which assumes the level component moves acco walk, but the slope component follows an AR1 pr a potentially non-zero value. This is useful for lon

AddSeasonal() The seasonal model can be thought of as a regression on `nseasons` dummy variables. Here we include weekly and yearly seasonality by setting `nseasons` to 7 and 365.

AddMonthlyAnnualCycle() This represents the contribution of each month. Alternatively, you can set `nseasons=12` in `AddSeasonal()` to address monthly seasonality.

AddRegressionHoliday() Previously in EDA we learı
Thanksgiving and Christmas negatively impact tax
function estimate the effect of each holiday or ev
regression. For this, I asked a friend of mine who
Chicago (well, ChatGPT of course) for the list of h
events in Chicago. For example, the Chicago Mara
the number of taxi trips.

Photo by Joseph Two on Unsplash

Then I set up the date of these dates:

```
christmas <- NamedHoliday("Christmas")
new_year <- NamedHoliday("NewYear")
thanksgiving <- NamedHoliday("Thanksgiving")
independence_day <- NamedHoliday("IndependenceDay")
labor_day <- NamedHoliday("LaborDay")
memorial_day <- NamedHoliday("MemorialDay")

auto.show <- DateRangeHoliday("Auto_show", start

                              end = as.Date(c("20
                                            ,
                                            ,
st.patrick <- DateRangeHoliday("stPatrick", start

                              end = as.Date(c("20

air.show <- DateRangeHoliday("air_show", start =


                              end = as.Date(c("20
                                          , "
lolla <- DateRangeHoliday("lolla", start = as.Dat

                              end = as.Date(c("20
                                          , "
marathon <- DateRangeHoliday("marathon", start =

                              end = as.Date(c("20
```


DateRangeHoliday() allows us to define events th different date each year or last for multiple days. helps with federal holidays.

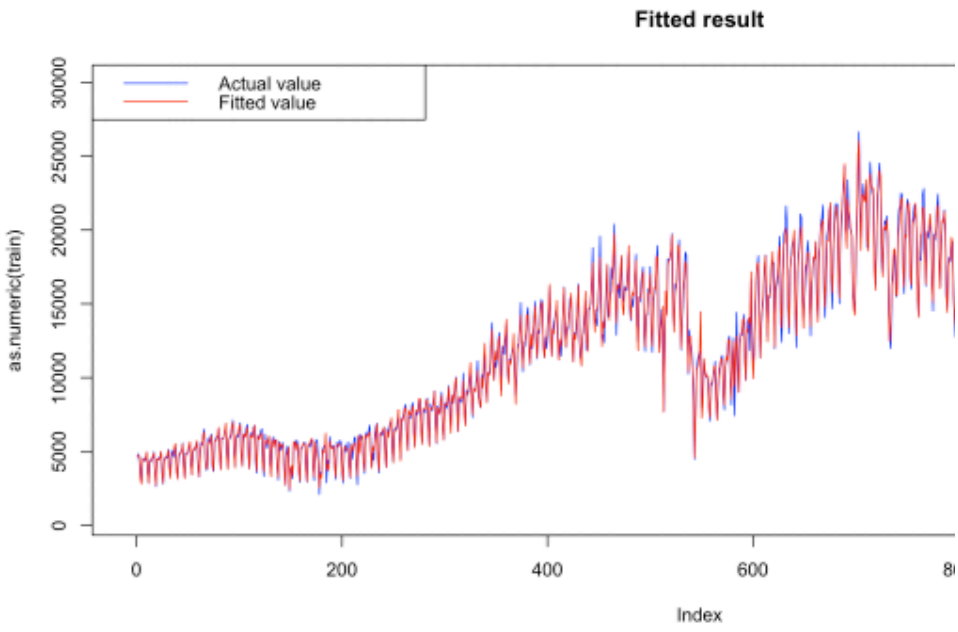
Then, define the list of these holidays for the AddRegressionHoliday() attribute:

```
holiday_list <- list(auto.show, st.patrick, air.show, lolla, marathon
              , christmas, new_year, thanksgiving, independence_day
              , labor_day, memorial_day)
```

I found this website very helpful in exploring diffe and parameters.

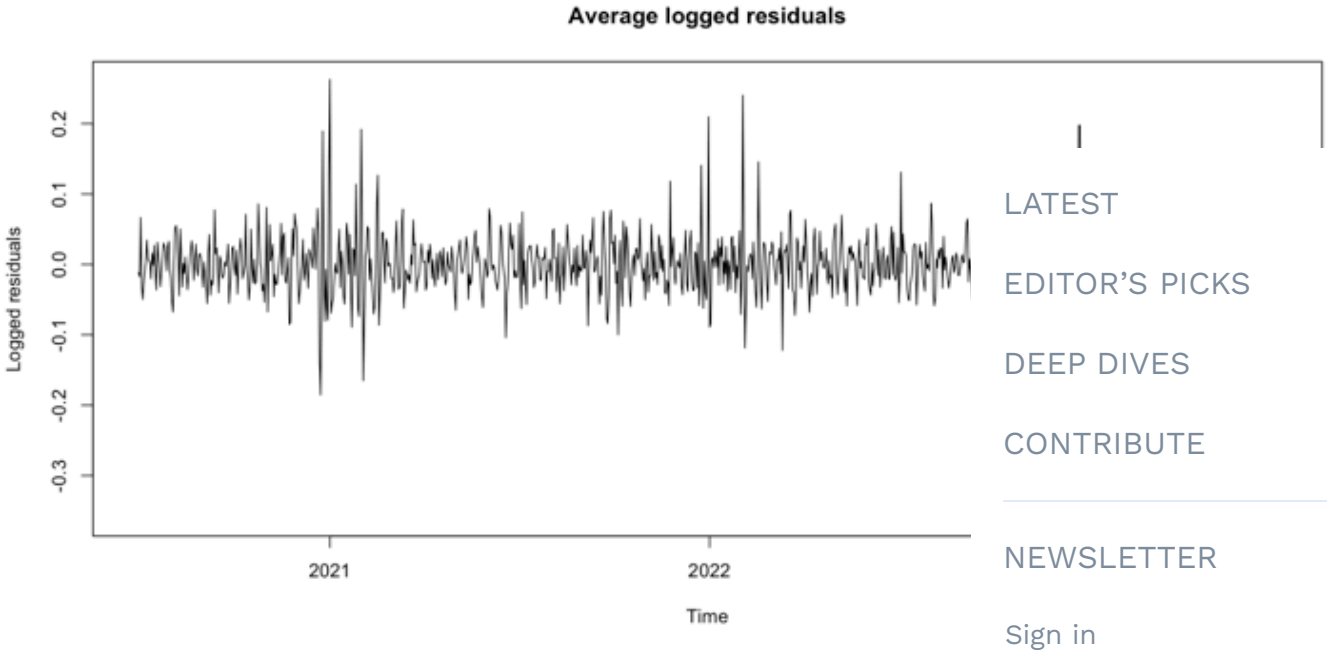
The fitted result shows that the model well captu of the time series.

```
fitted_values <- as.numeric(residuals.bsts(model_
train_hat <- exp(fitted_values) - 1
plot(as.numeric(train), type = "l", col = "blue",
lines(train_hat, col = "red")
legend("topleft", legend = c("Actual value", "Fit
```

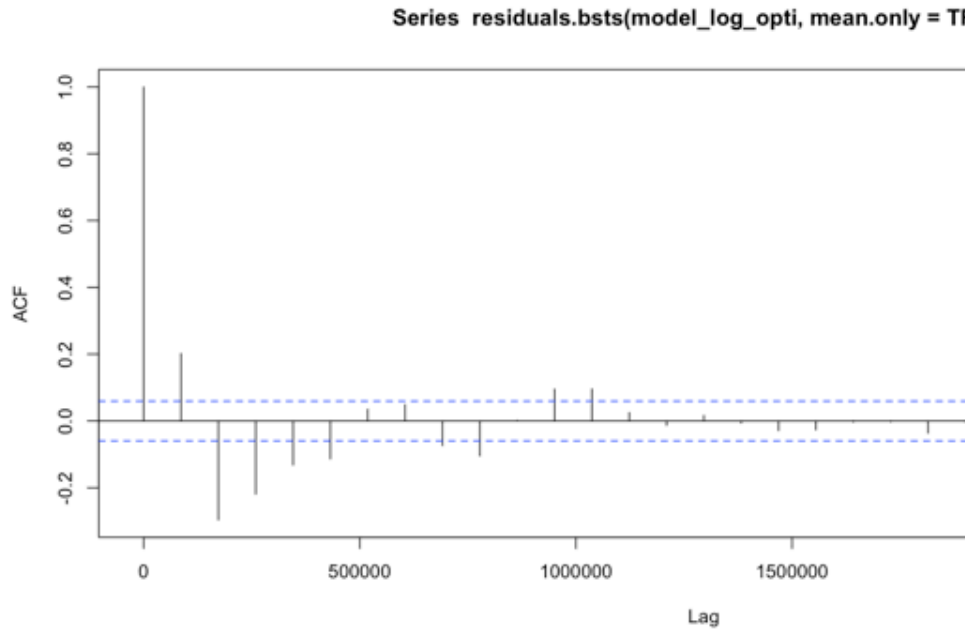


BSTS fitted result | Graph by author

In the residual analysis, although the residuals have a mean of zero, there is still some remaining seasonality. Additionally, the residuals exhibit autocorrelation in the first few lags.



Residuals of the BSTS model | Graph by author



ACF on the residuals | Graph by author

However, when comparing these results to the ori
is evident that the model has successfully captur

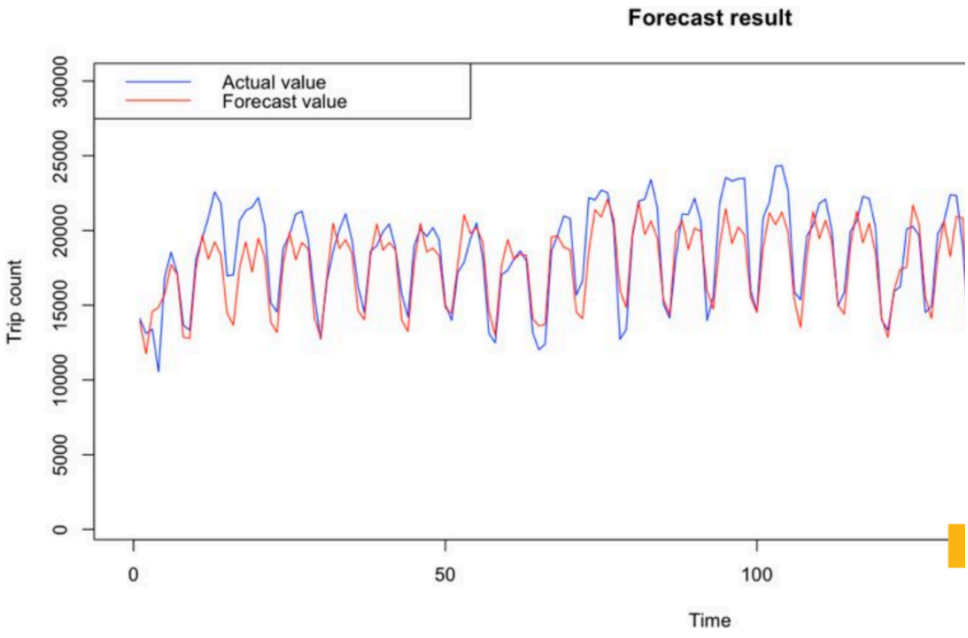
seasonality, holiday effects, and trend components. This indicates that the BSTS model effectively addresses the key patterns in the data, leaving only minor residual structures to be examined further.

Forecast Result and Insights

Now, let's evaluate the forecast results of the mo transform the predicted values, as the model pro

```
horizon <- length(test)
pred_log_opti <- predict(model_log_opti, horizon
forecast_values_log_opti <- exp(pred_log_opti$mea
```

```
plot(as.numeric(test), type = "l", col = "blue",
lines(forecast_values_log_opti, col = "red")
legend("topleft", legend = c("Actual value", "For
```



Forecast and actual values | Graph by author

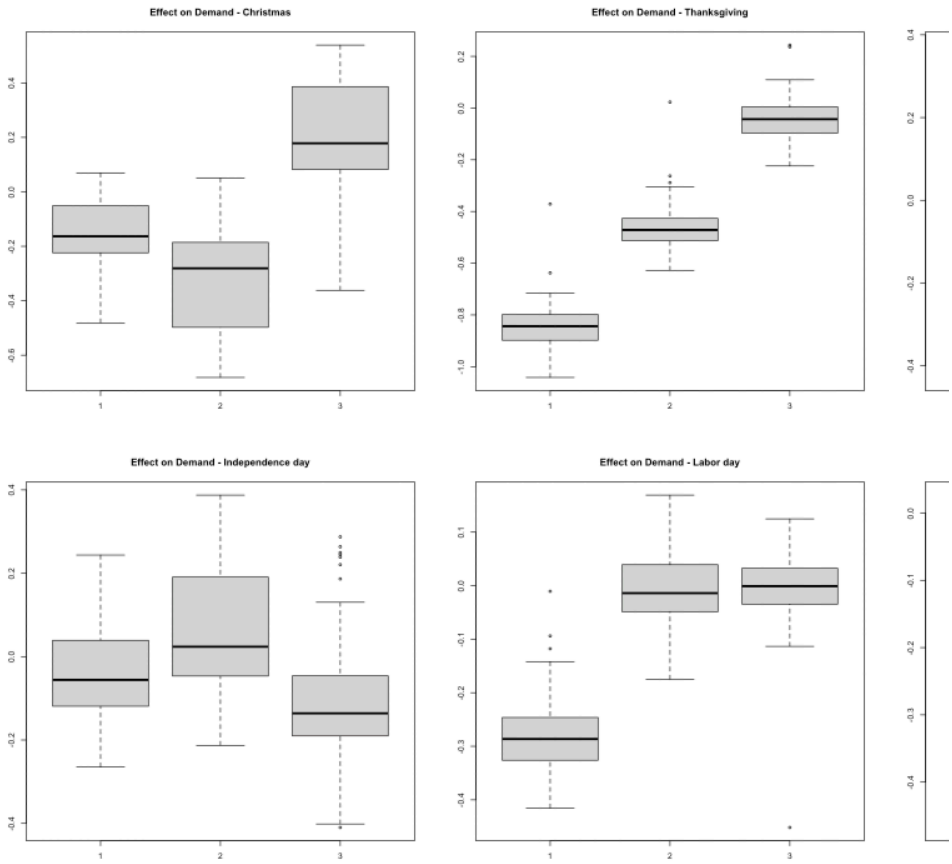
The model achieves a Mean Absolute Percentage 9.76%, successfully capturing the seasonality and

holidays.

The analysis of holiday and event effects offers valuable insights for business strategies. The following graphs illustrate the impact of holiday regressions:

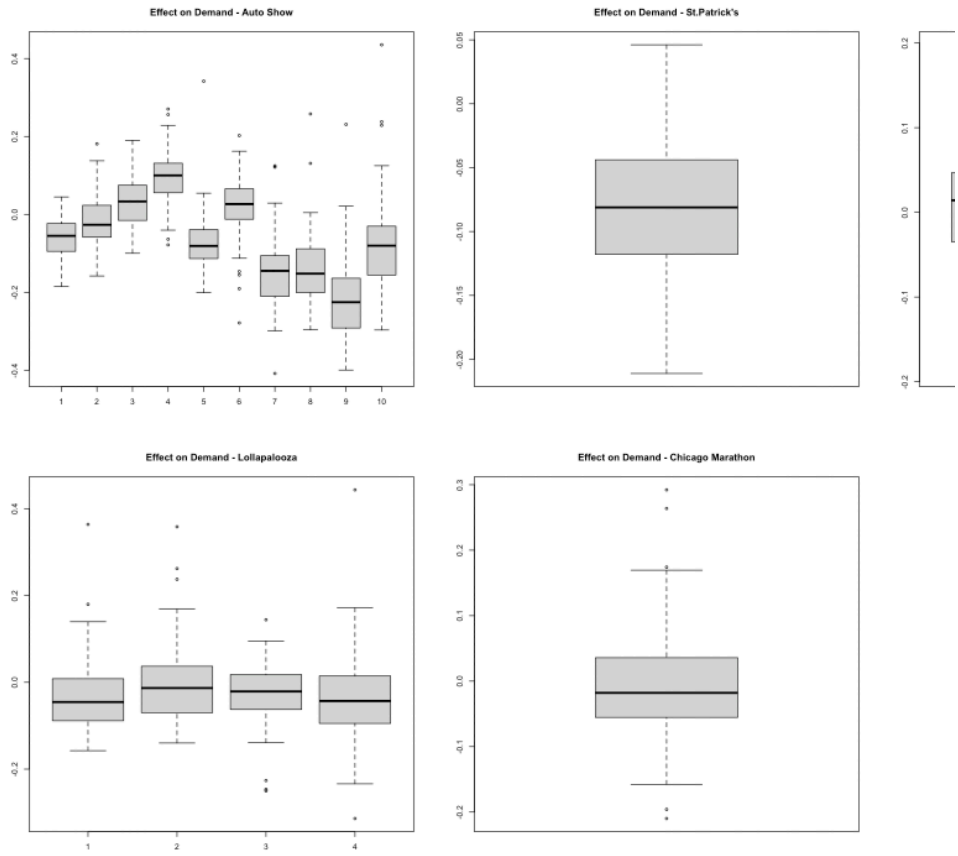
```
PlotHoliday(thanksgiving, model_log_opti)
PlotHoliday(marathon, model_log_opti)
```

The day before federal holidays has a significantly on the number of trips. For example, both Thanks day before show a noticeable drop in taxi trips. Th be due to lower demand or limited supply. Compa investigate these reasons further and develop str them.



Effect of holidays | Graph by author

Contrary to the initial hypothesis, major events like the Chicago Marathon did not show a significant increase in taxi demand. This suggests that demand during such events may not be as high as expected. Conducting customer segmentation research can help identify specific groups that might be influenced by events, revealing potential opportunities for targeted marketing and services. Breaking down the data by sub-areas in Chicago can also p insights. The impact of events might vary across c neighborhoods, and understanding these variatior localized strategies.



Effect of events | Graph by author

Conclusion

So this is how you can use BSTS model to predict You can experiment different state component or

how the model fit the data differently. Hope you enjoy the process and please give me claps if you find this article helpful!

WRITTEN BY

Lea Wu

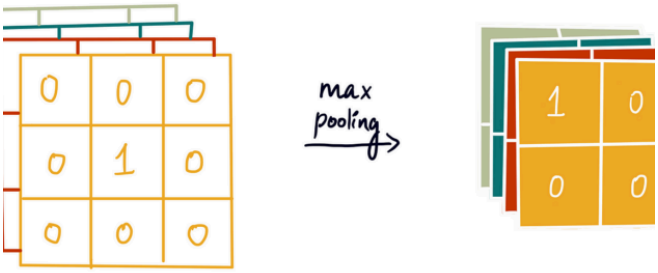
See all from Lea Wu

Topics:

Data Science | Hands On Tutorials | Programming | Time Series Forecasting

Share this article:

Related Articles



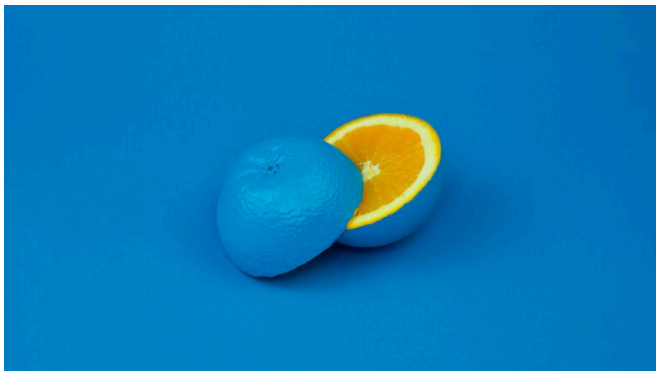
ARTIFICIAL INTELLIGENCE

Implementing Convolutional Neural Networks in TensorFlow

Step-by-step code guide to building a Convolutional Neural Network

Shreya Rao

August 20, 2024 6 min read



DATA SCIENCE

Hands-on Tim Detection usi with Python

Here's how to use detect signals wi lines of...

Piero Paialunga

August 21, 2024 1

DATA SCIENCE

Solving a Constrained Project Scheduling Problem with Quantum Annealing

Solving the resource constrained project scheduling problem (RCPSP) with D-Wave's hybrid constrained quadratic model (CQM)

Luis Fernando PÉREZ ARMAS, Ph.D.

August 20, 2024 29 min read

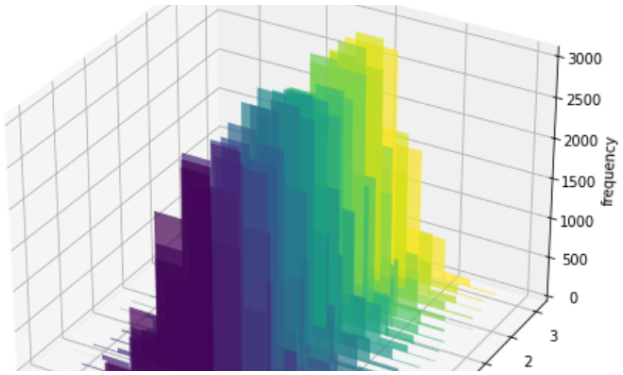
DATA SCIENCE

Back To Basic Regression ar

An illustrated gui learning concept

Shreya Rao

February 3, 2023



DATA SCIENCE

Must-Know in Statistics: The Bivariate Normal Projection Explained

Derivation and practical examples of this powerful concept

Luigi Battistoni

August 14, 2024 7 min read

DATA SCIENCE

Optimizing Marketing Campaigns with Budgeted Multi-Armed Bandits

With demos, our new solution, and a video

Vadim Arzamasov

August 16, 2024 10 min read

DATA SCIENCE

Our Columns

Columns on TDS collections of po or category...

TDS Editors

November 14, 2020

LATEST

EDITOR'S PICKS

DEEP DIVES

CONTRIBUTE

NEWSLETTER

Sign in

Contributor Portal

Your home for data science and Al. The world's leading publication for data science, data analytics, data engineering, machine learning, and artificial intel

Subscribe to Our Newsletter