UC business Analytics R programming Guide 4

Koji Mizumura

2019-10-11

Table of Contents

library(tidyverse)  
library(forecast)  
library(magrittr)  
library(tidymodels)  
library(readxl)

# Exploring and visualizing time series

## Creating time series objects in R

A time series can be thought of as a vector or matrix of numbers along with some information about what times those numbers were recorded. This information is stored in a ts object in R. In most exercises, you will use time series that are part of existing packages. However, if you want to work with your own data, you need to know how to create a ts object in R.

Let’s look at an example usnim\_2002 below, containing net interest margins for US banks for the year 2002 (source: FFIEC).

The function ts() takes in three arguments:

* data is set to everything in usnim\_2002 except for the date column; it isn’t needed since the ts object will store time information separately.
* start is set to the form c(year, period) to indicate the time of the first observation. Here, January corresponds with period 1; likewise, a start date in April would refer to 2, July to 3, and October to 4. Thus, period corresponds to the quarter of the year.
* frequency is set to 4 because the data are quarterly. In this exercise, you will read in some time series data from an xlsx file using read\_excel(), a function from the readxl package, and store the data as a ts object. Both the xlsx file and package have been loaded into your workspace.

# Read the data from Excel into R  
mydata <- read\_excel("data/exercise1.xlsx")  
  
# Look at the first few lines of mydata  
head(mydata)  
## # A tibble: 6 x 4  
## ...1 Sales AdBudget GDP  
## <chr> <dbl> <dbl> <dbl>  
## 1 Mar-81 1020. 659. 252.  
## 2 Jun-81 889. 589 291.  
## 3 Sep-81 795 512. 291.  
## 4 Dec-81 1004. 614. 292.  
## 5 Mar-82 1058. 647. 279.  
## 6 Jun-82 944. 602 254  
  
# Create a ts object called myts  
myts <- ts(mydata[, 2:4], start = c(1981, 1), frequency = 4)

## Time series plots

The first step in any data analysis task is to plot the data. Graphs enable you to visualize many features of the data, including patterns, unusual observations, changes over time, and relationships between variables. Just as the type of data determines which forecasting method to use, it also determines which graphs are appropriate.

You can use the autoplot() function to produce a time plot of the data with or without facets, or panels that display different subsets of data:

autoplot(usnim\_2002, facets = FALSe)

The above method is one of the many taught in this course that accepts boolean arguments. Both T and TRUE mean “true”, and F and FALSE mean “false”, however, T and F can be overwritten in your code. Therefore, you should only rely on TRUE and FALSE to set your indicators for the remainder of the course.

You will use two more functions in this exercise, which.max() and frequency(). which.max() can be used to identify the smallest index of the maximum value

x <- c(4, 5, 5)  
which.max(x)  
## [1] 2  
# [1] 2

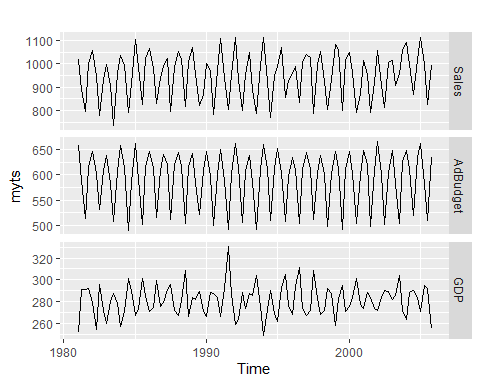
To find the number of observations per unit time, use frequency(). Recall the usnim\_2002 data from the previous exercise:

> frequency(usnim\_2002)  
[1] 4

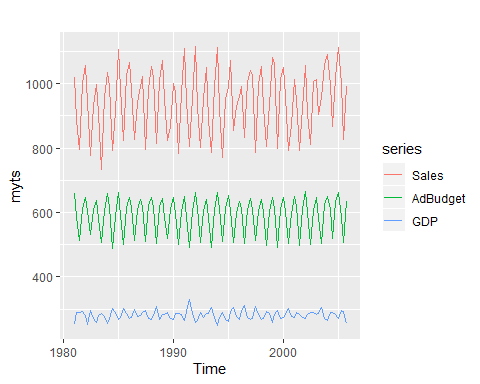
Because this course involves the use of the forecast and ggplot2 packages, they have been loaded into your workspace for you, as well as myts from the previous exercise and the following three series (available in the package forecast):

* gold containing gold prices in US dollars
* woolyrnq containing information on the production of woollen yarn in Australia
* gas containing Australian gas production

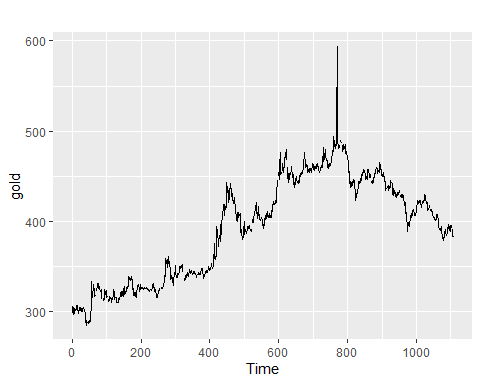
# Plot the data with facetting  
autoplot(myts, facets = TRUE)



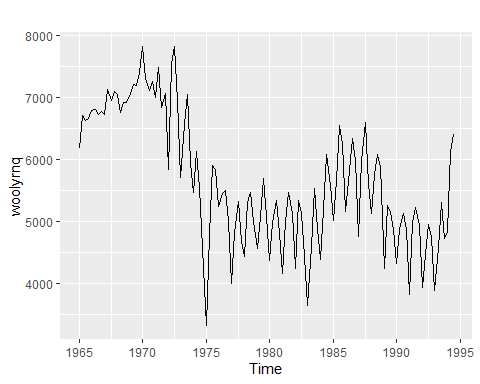
# Plot the data without facetting  
autoplot(myts, facets = FALSE)



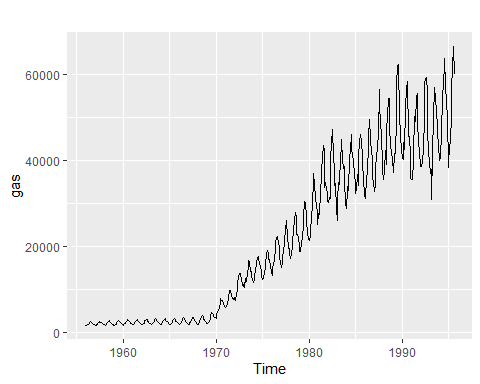
# Plot the three series  
autoplot(gold)



autoplot(woolyrnq)



autoplot(gas)



# Find the outlier in the gold series  
goldoutlier <- which.max(gold)  
  
# Look at the seasonal frequencies of the three series  
frequency(gold)  
## [1] 1  
frequency(woolyrnq)  
## [1] 4  
frequency(gas)  
## [1] 12

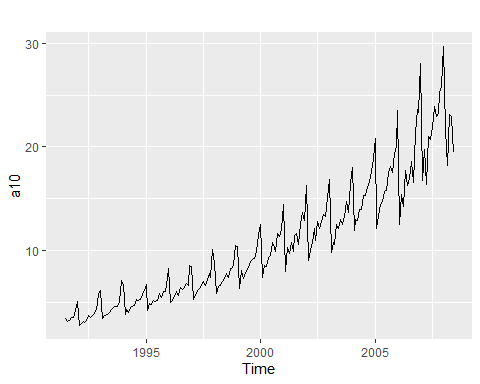
## Seasonal plots

Along with time plots, there are other useful ways of plotting data to emphasize seasonal patterns and show changes in these patterns over time.

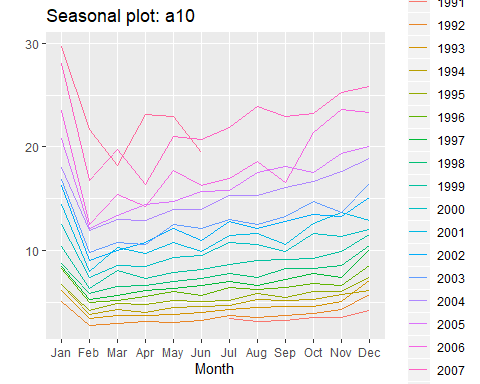
A seasonal plot is similar to a time plot except that the data are plotted against the individual “seasons” in which the data were observed. You can create one using the ggseasonplot() function the same way you do with autoplot().

An interesting variant of a season plot uses polar coordinates, where the time axis is circular rather than horizontal; to make one, simply add a polar argument and set it to TRUE.A subseries plot comprises mini time plots for each season. Here, the mean for each season is shown as a blue horizontal line.One way of splitting a time series is by using the window() function, which extracts a subset from the object x observed between the times start and end.

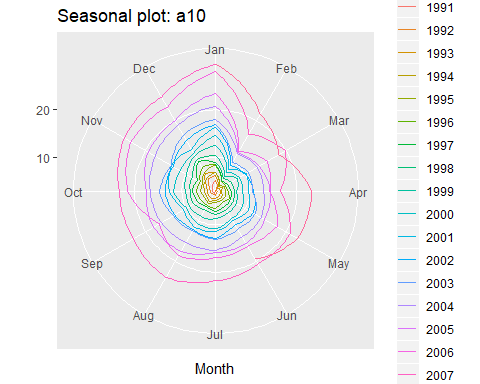
# Load the fpp2 package  
library(fpp2)  
  
# Create plots of the a10 data  
autoplot(a10)



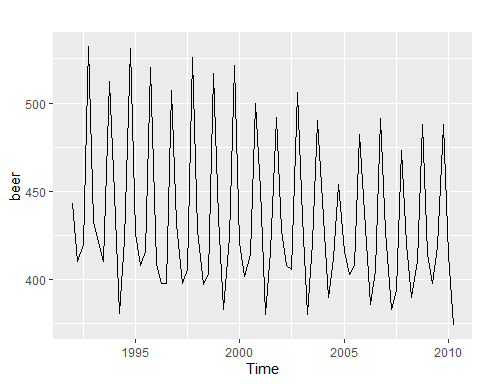
ggseasonplot(a10)



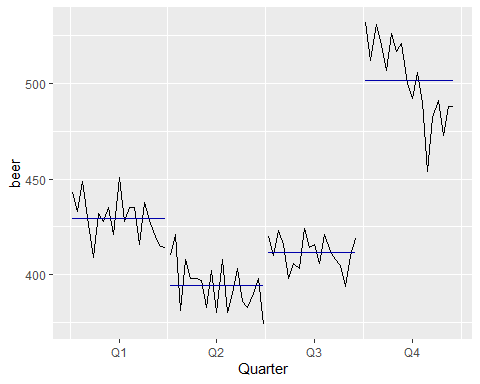
# Produce a polar coordinate season plot for the a10 data  
ggseasonplot(a10, polar = TRUE)



# Restrict the ausbeer data to start in 1992  
beer <- window(ausbeer, start = 1992)  
  
# Make plots of the beer data  
autoplot(beer)



ggsubseriesplot(beer)



## Trends, seasonality, and cyclicity

# Benchmark methods and forecast accuracy

# Exponential smoothing

# Forecasting with ARIMA models

# Advanced methods