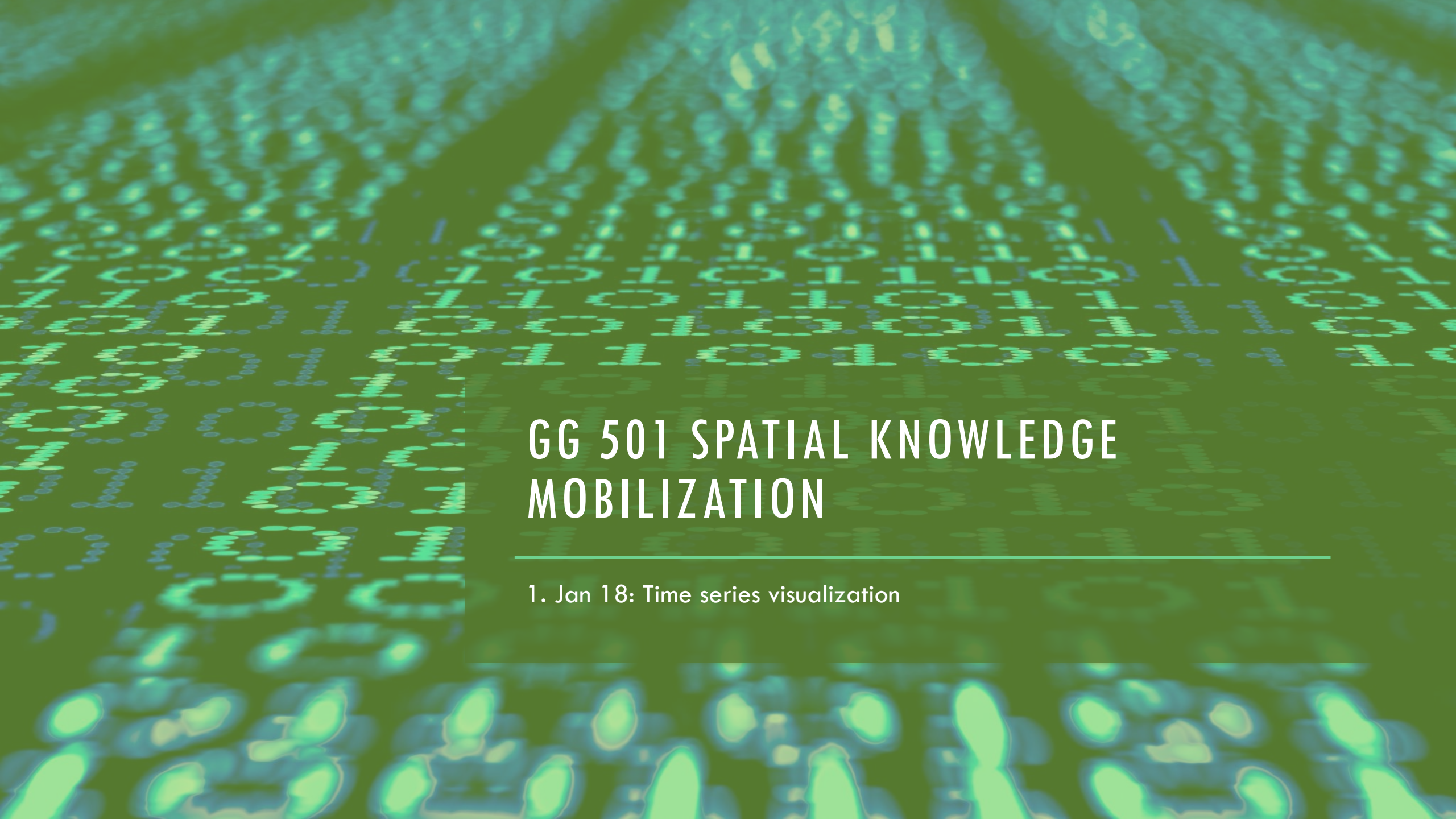


GG 501 SPATIAL KNOWLEDGE MOBILIZATION

1. Jan 18: Time series visualization

REVIEW

Typical visible aesthetics

Aesthetic	Description
x	X axis position
y	Y axis position
fill	Fill color
color	Color of points, outlines of other geoms
size	Area or radius of points, thickness of lines

Aesthetic	Description
alpha	Transparency
linetype	line dash pattern
labels	Text on a plot or axes
shape	Shape

REVIEW – TAKE 15 MINS

https://ggplot2-book.org/statistical-summaries.html#statistical-summaries

5.4.1 Exercises

1. What binwidth tells you the most interesting story about the distribution of `carat`?

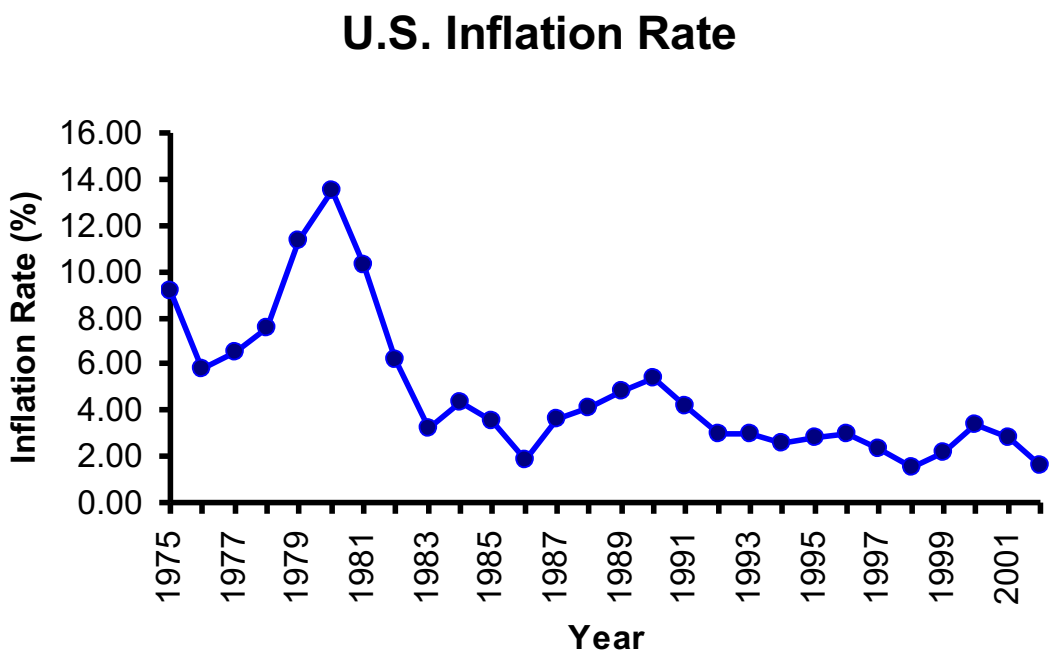
2. Draw a histogram of `price`. What interesting patterns do you see?

3. How does the distribution of `price` vary with `clarity`?

TIME SERIES DATA IN R

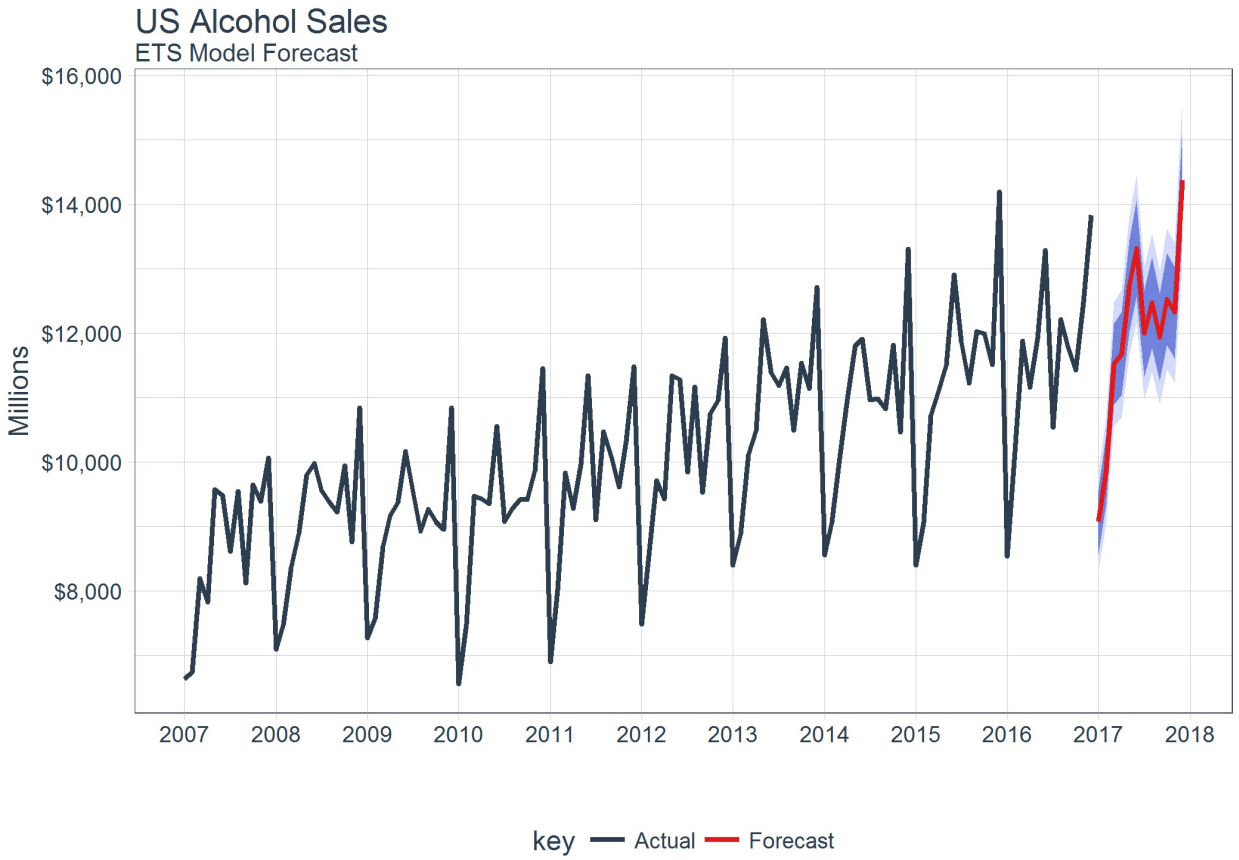
- Time series analysis is a branch of analysis methods focused on treatment of time series data, which are any metric measured over a regular interval over time
- Time Series data are extremely common
  - weather variables measured over time
  - stock prices
  - financial / sales data
  - etc.
- Time series are typically represented as a line plot, with time on the X-axis and metric variable on the Y-axis

Time series plot



TIME SERIES DATA IN R

- Forecasting is a major focus of time series analysis
  - can we infer what tomorrow will look like based on previous values/patterns in the time series?
- Modelling is a major focus of time series analysis
- As part of many aspects of time series analysis, visualization of models via plots is an important step in the workflow



TIME SERIES DATA IN R

- There are many many packages available for temporal data and time series analysis in R
  - ts in stats package
  - lubridate
  - zoo
  - xts
  - forecast
  - tsibble
  - fable
  - prophet..
- Use TimeSeries view on CRAN to review different packages available

https://cran.r-project.org/web/views/TimeSeries.html

CRAN Task View: Time Series Analysis

Maintainer: Rob J Hyndman
Contact: Rob.Hyndman at monash.edu
Version: 2021-12-20
URL: https://CRAN.R-project.org/view=TimeSeries

Base R ships with a lot of functionality useful for time series, in particular in the stats package. This is complemented by many packages on CRAN is also a considerable overlap between the tools for time series and those in the Econometrics and Finance task views. The packages in this view cai topics. If you think that some package is missing from the list, please let us know.

Basics

- *Infrastructure* : Base R contains substantial infrastructure for representing and analyzing time series data. The fundamental class is `"ts"` that ((using numeric time stamps). Hence, it is particularly well-suited for annual, monthly, quarterly data, etc.
- *Rolling statistics* : Moving averages are computed by `ma` from forecast, and `rollmean` from zoo. The latter also provides a general function `rc` statistics functions. slider calculates a diverse and comprehensive set of type-stable running functions for any R data types. tsibble provides `s` overlapping sliding windows, and `stretch()` for expanding windows. tbrf provides rolling functions based on date and time windows instead parallel functions for computing rolling statistics. runner provides tools for running any R function in rolling windows or date windows. runst some running sample statistics. For data.table, `froll()` can be used for high-performance rolling statistics.
- *Graphics* : Time series plots are obtained with `plot()` applied to `ts` objects. (Partial) autocorrelation functions plots are implemented in `acf(` provided by `Acf()` and `Pacf()` in forecast, along with a combination display using `tsdisplay()`. Seasonal displays are obtained using `month` `seasplot` in tsutils. feasts and brolgar provide various time series graphics for tsibble objects including time plots, season plots, subseries plo combination displays. Interactive graphics for tsibbles using htmlwidgets are provided by tsibbletalk. SDD provides more general serial depe and plots the distance covariance and correlation functions of time series. ggseas provides additional ggplot2 graphics for seasonally adjusted are implemented in sugrrants. gravitas allows for visualizing probability distributions conditional on bivariate temporal granularities. dygraph interactive time series charting library. TSstudio provides some interactive visualization tools for time series. ZRA plots forecast objects from fan plots of forecast distributions are provided by forecast and vars. More flexible fan plots of any sequential distributions are implemented in

Times and Dates

- Class `"ts"` can only deal with numeric time stamps, but many more classes are available for storing time/date information and computing wit *and Time Classes in R* by Gabor Grothendieck and Thomas Petzoldt in R News 4(1) , 29-32.
- Classes `"yearmon"` and `"yearqtr"` from zoo allow for more convenient computation with monthly and quarterly observations, respectively.
- Class `"Date"` from the base package is the basic class for dealing with dates in daily data. The dates are internally stored as the number of day
- The chron package provides classes for `dates()`, `hours()` and date/time (intra-day) in `chron()`. There is no support for time zones and dayli are (fractional) days since 1970-01-01.
- Classes `"POSIXct"` and `"POSIXlt"` implement the POSIX standard for date/time (intra-day) information and also support time zones and dayl computations require some care and might be system-dependent. Internally, `"POSIXct"` objects are the number of seconds since 1970-01-01 0 functions that facilitate certain POSIX-based computations, while clock provides a comprehensive library for date-time manipulations using a (durations, time points, zoned-times, and calendars). timechange allows for efficient manipulation of date-times accounting for time zones an weekly data to monthly data in several different ways.
- Class `"timeDate"` is provided in the timeDate package (previously: fCalendar). It is aimed at financial time/date information and deals with ti

```r
ts (inputData, frequency = 4, start = c(1959, 2)) # frequency 4 => Quarterly Data
ts (1:10, frequency = 12, start = 1990) # freq 12 => Monthly data.
ts (inputData, start=c(2009), end=c(2014), frequency=1) # Yearly Data
```

TS OBJECTS

scan {base} R Documentation

Read Data Values

Description

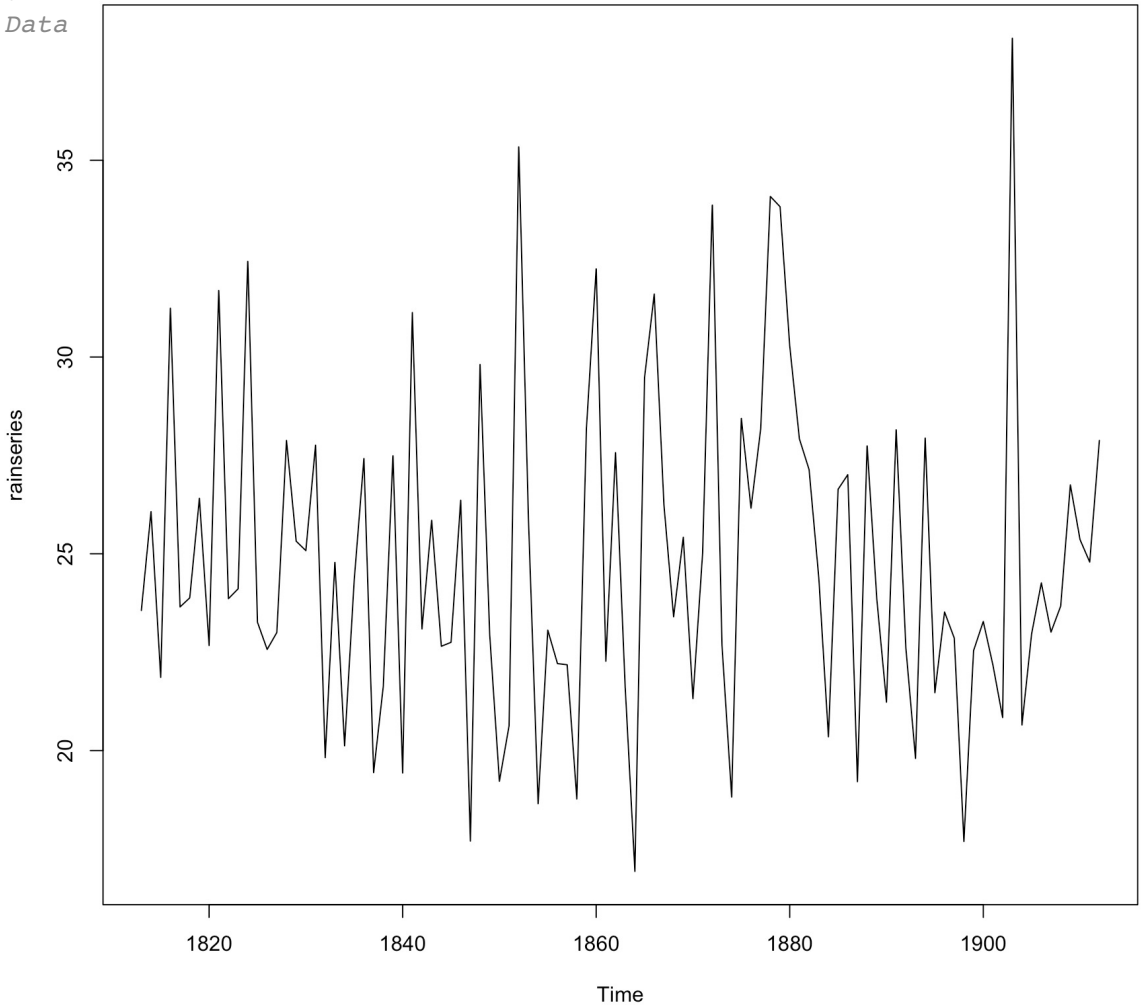
Read data into a vector or list from the console or file.

Usage

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
     quote = if(identical(sep, "\n")) "" else "'\"", dec = "."
     skip = 0, nlines = 0, na.strings = "NA",
     flush = FALSE, fill = FALSE, strip.white = FALSE,
     quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE
     comment.char = "", allowEscapes = FALSE,
     fileEncoding = "", encoding = "unknown", text, skipNul =
```

Arguments

file	the name of a file to read data values from. If the specified file is "", then input is taken from the keyboard




```r
rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)
rainseries <- ts(rain,start=c(1813))
plot.ts(rainseries)
```

ts {stats} R Documentation

Time-Series Objects

Description

The function `ts` is used to create time-series objects.

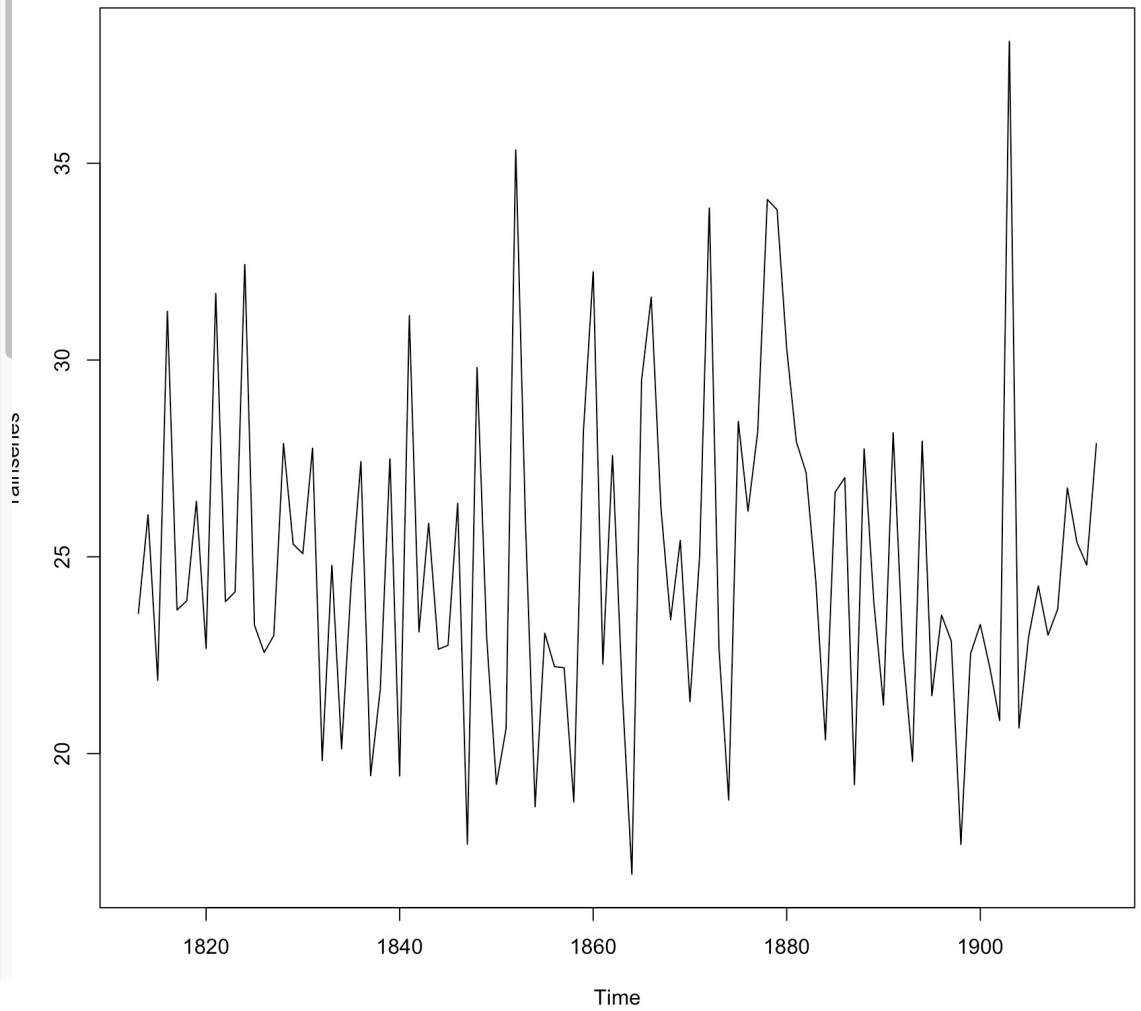
`as.ts` and `is.ts` coerce an object to a time-series and test whether an object is a time series.

Usage

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x, ...)
is.ts(x)
```

Arguments

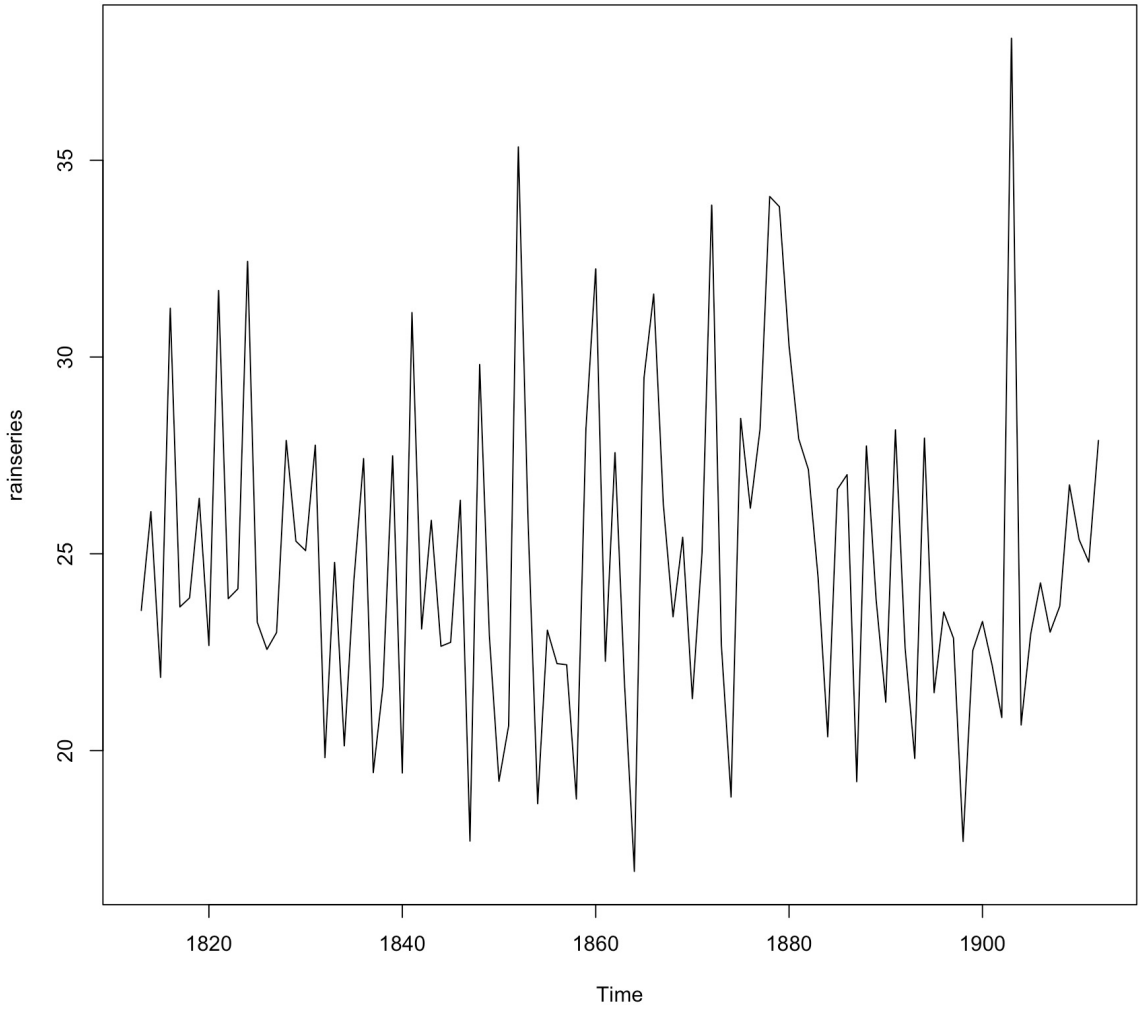
`data`	a vector or matrix of the observed time-series values. A data frame will be coerced to a numeric matrix via `data.matrix`. (See also 'Details'.)
`start`	the time of the first observation. Either a single number or a vector of two numbers (the second of which is an integer), which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
`end`	the time of the last observation, specified in the same way as `start`.
`frequency`	the number of observations per unit of time.
`deltat`	the fraction of the sampling period between successive observations; e.g., 1/12 for monthly data. Only one of `frequency` or `deltat` should be provided.
`ts.eps`	time series comparison tolerance. Frequencies are considered equal if their absolute difference is less than `ts.eps`.
`class`	class to be given to the result, or none if `NULL` or `"none"`. The default is `"ts"` for a single series, `c("mts", "ts", "matrix")` for multiple series.
`names`	a character vector of names for the series in a multiple series: defaults to the colnames of `data`, or `Series 1`, `Series 2`, ...




```
rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)
rainseries <- ts(rain,start=c(1813))
plot.ts(rainseries)
```

Questions to ask when viewing a time series plot

- What is the sampling unit (i.e. what temporal interval is represented in x-axis)
- what is the average value?
- is there a trend in the average value?
- is there a trend in the variance?
- are there gaps in the time series?
- is there any cyclical or periodic behaviour?



TIME SERIES COMPONENTS

Trend Component

long term increase or decrease

Seasonal Component

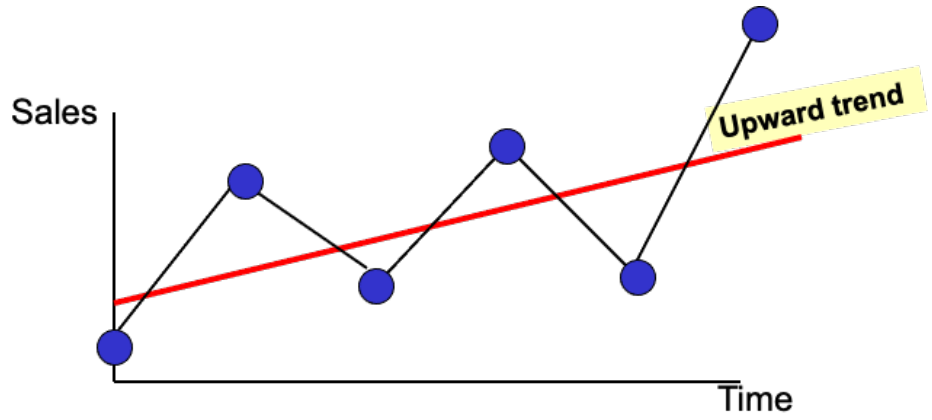
short-term wave-like patterns/changes

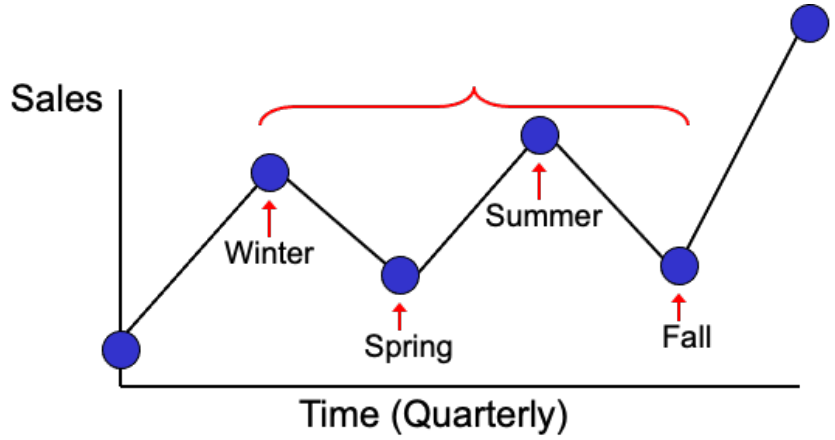
Cyclical Component

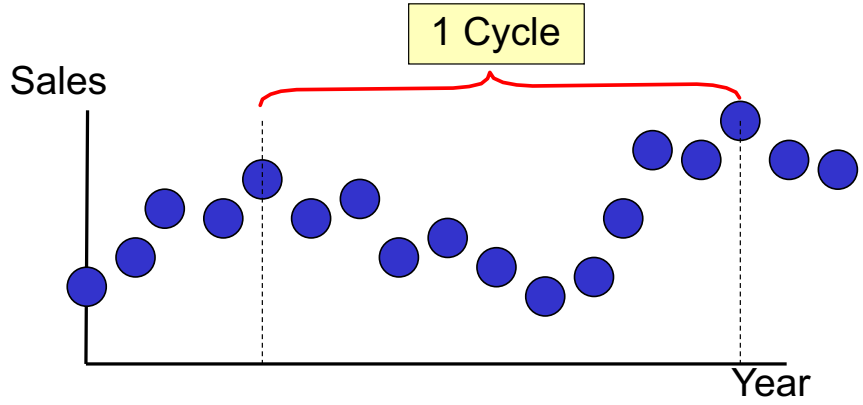
long-term wave-like patterns/changes

Random Component

random noise/fluctuations, unpredictable







Understanding and visualizing time series components

- Time series decomposition is the practice of separating a time series into its components – which are generally
  - Trend, random
  - Trend, seasonal, random
- These components can be combined additively or multiplicatively

Finding the 'trend' in a non-seasonal time-series

- Simple moving average – can use the SMA function in the TTR package

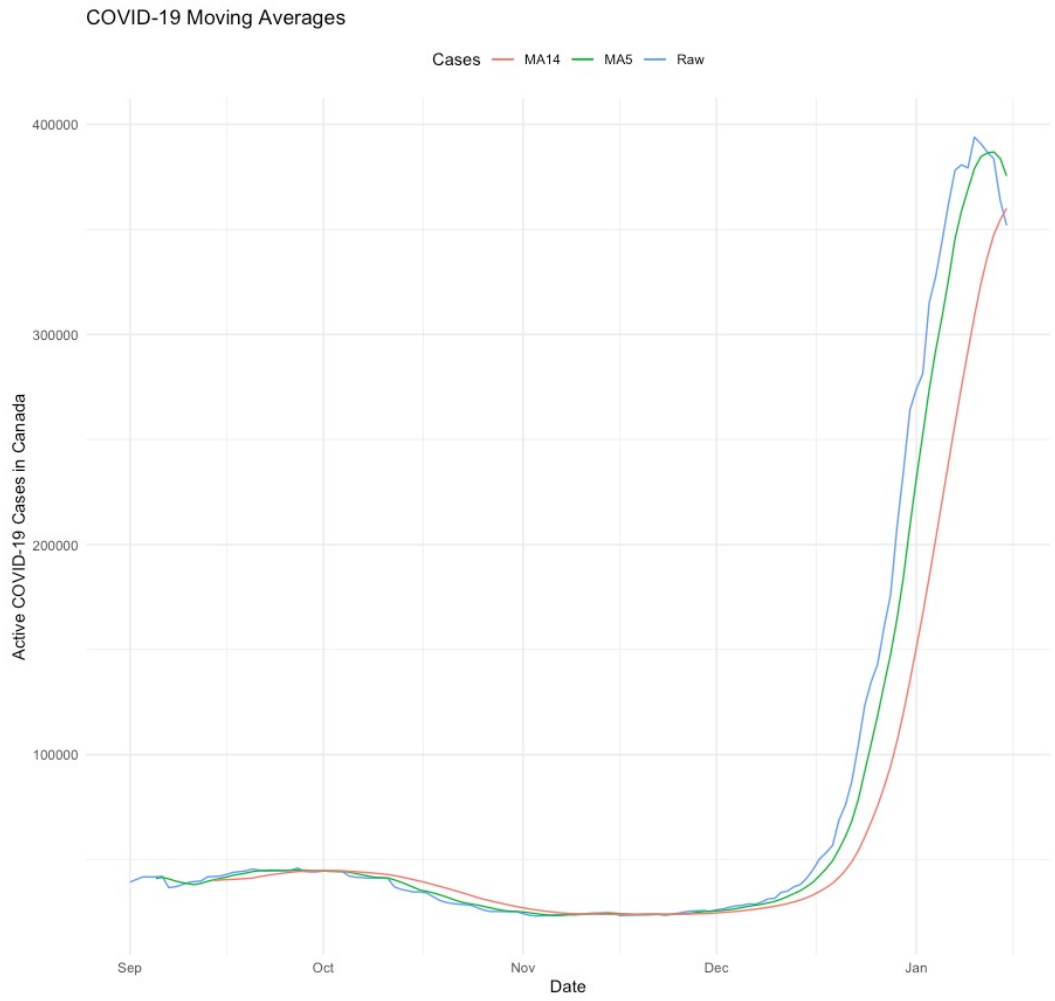
```
library(TTR)
library(readr)
df <- read_csv("https://raw.githubusercontent.com/ccodwg/Covid19Canada/mas
s_canada.csv")

df$date_active <- as.Date(df$date_active, "%d-%m-%Y")

df <- dplyr::filter(df, date_active >= as.Date('01-09-2021', "%d-%m-%Y"))

df$SMA5 <- SMA(df$active_cases, n=5)
df$SMA14 <- SMA(df$active_cases, n=14)

# Now we plot the values in ggplot
pl <- ggplot(df , aes(x = date_active))
pl <- pl + geom_line(aes(y = active_cases, color = "Raw"), group = 1)
pl <- pl + geom_line(aes(y = SMA5, color = "MA5"), group = 1)
pl <- pl + geom_line(aes(y = SMA14, color = "MA14"), group = 1)
pl <- pl +  theme_minimal()
pl <- pl + theme(legend.position = "top")
pl <- pl + labs(title ="COVID-19 Moving Averages")
pl <- pl + labs(color="Cases")
pl
options(scipen=999)
```



Finding the 'trend' in a non-seasonal time-series

- specify the order (span) of the simple moving average, using the parameter "n"
  - calculate a simple moving average of order 5, we set n=5 in the SMA() function

```
library(TTR)
library(readr)
df <- read_csv("https://raw.githubusercontent.com/ccodwg/Covid19Canada/mas
s_canada.csv")

df$date_active <- as.Date(df$date_active, "%d-%m-%Y")

df <- dplyr::filter(df, date_active >= as.Date('01-09-2021', "%d-%m-%Y"))

df$SMA5 <- SMA(df$active_cases, n=5)
df$SMA14 <- SMA(df$active_cases, n=14)

# Now we plot the values in ggplot
pl <- ggplot(df , aes(x = date_active))
pl <- pl + geom_line(aes(y = active_cases, color = "Raw"), group = 1)
pl <- pl + geom_line(aes(y = SMA5, color = "MA5"), group = 1)
pl <- pl + geom_line(aes(y = SMA14, color = "MA14"), group = 1)
pl <- pl +  theme_minimal()
pl <- pl + theme(legend.position = "top")
pl <- pl + labs(title ="COVID-19 Moving Averages")
pl <- pl + labs(color="Cases")
pl
options(scipen=999)
```

Decomposing a seasonal time-series

- Number of births per month in New York city, from January 1946 to December 1959 available in the file http://robjhyndman.com/tsdldata/data/nybirths.dat

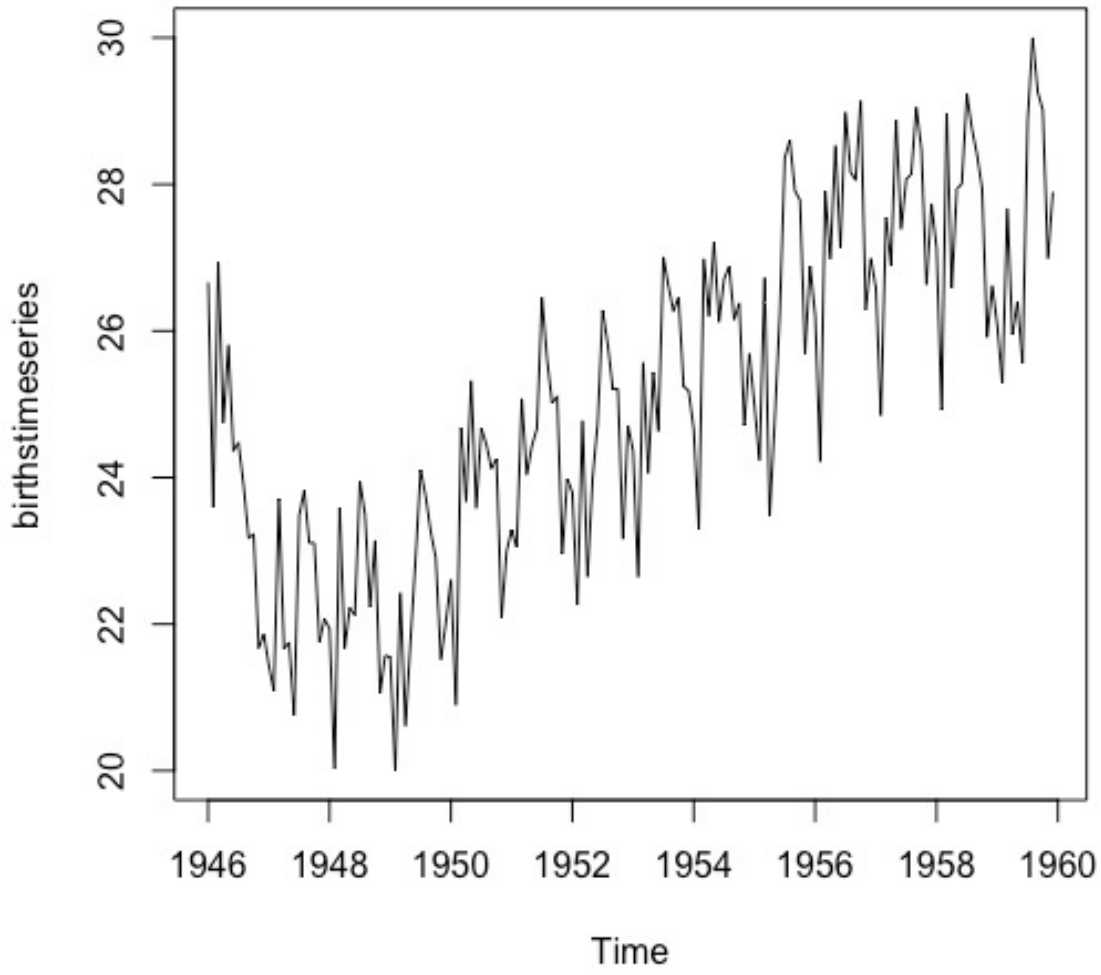
```
> births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
  Read 168 items
> birthstimeseries <- ts(births, frequency=12, start=c(1946,1))
> birthstimeseries
     Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1946 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175 23.227 21.672 21.870
1947 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105 23.110 21.759 22.073
1948 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238 23.142 21.059 21.573
1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907 21.519 22.025
1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252 22.084 22.991
1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110 22.964 23.981
1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199 23.162 24.707
1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462 25.246 25.180
1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379 24.712 25.688
1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784 25.693 26.881
1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136 26.291 26.987
1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484 26.634 27.735
1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945 25.912 26.619
1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012 26.992 27.897
```

Decomposing a seasonal time-series

```
births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

birthstimeseries <- ts(births, frequency=12, start=c(1946,1))

plot.ts(birthstimeseries)
```



Decomposing a seasonal time-series

NO

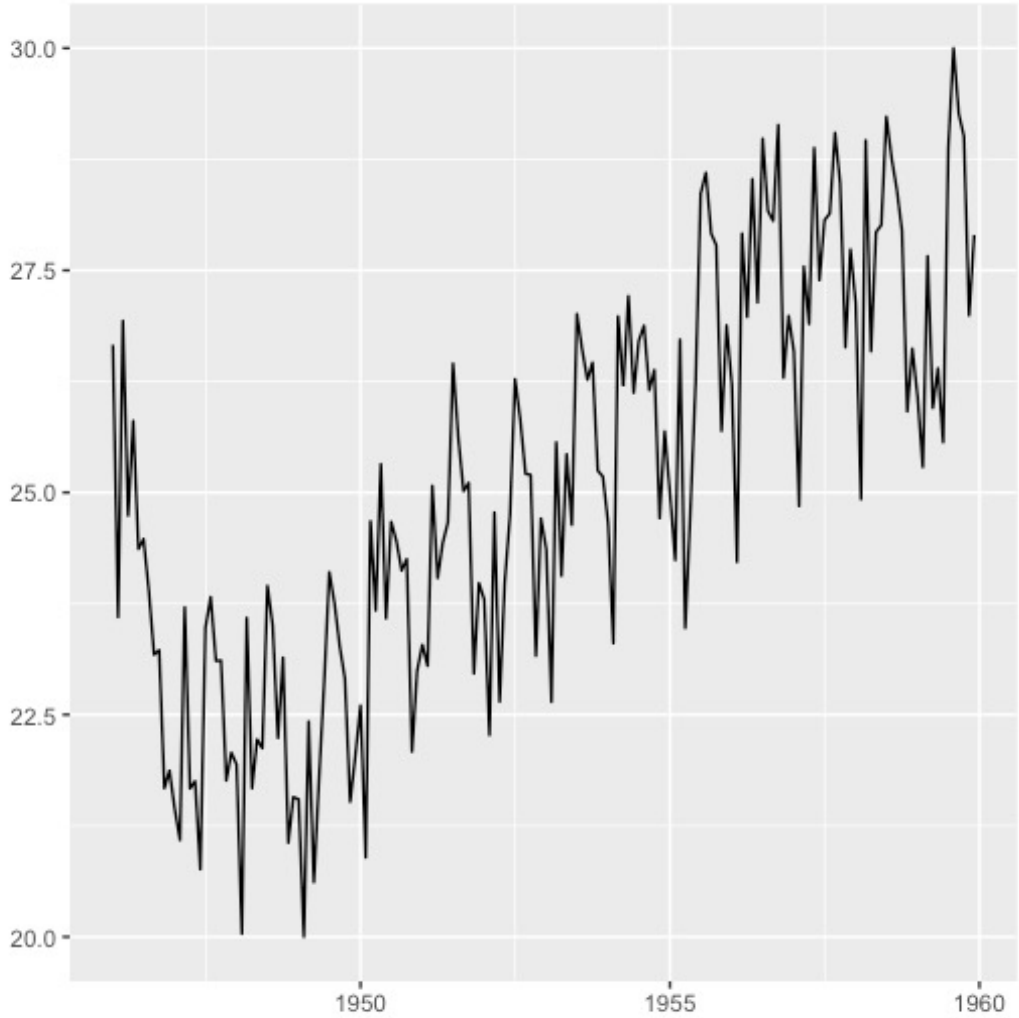
```
ggplot(birthstimeseries)

#Error: `data` must be a data frame, or other ob
object with class ts

#Run `rlang::last_error()` to see where the erro
```

YES

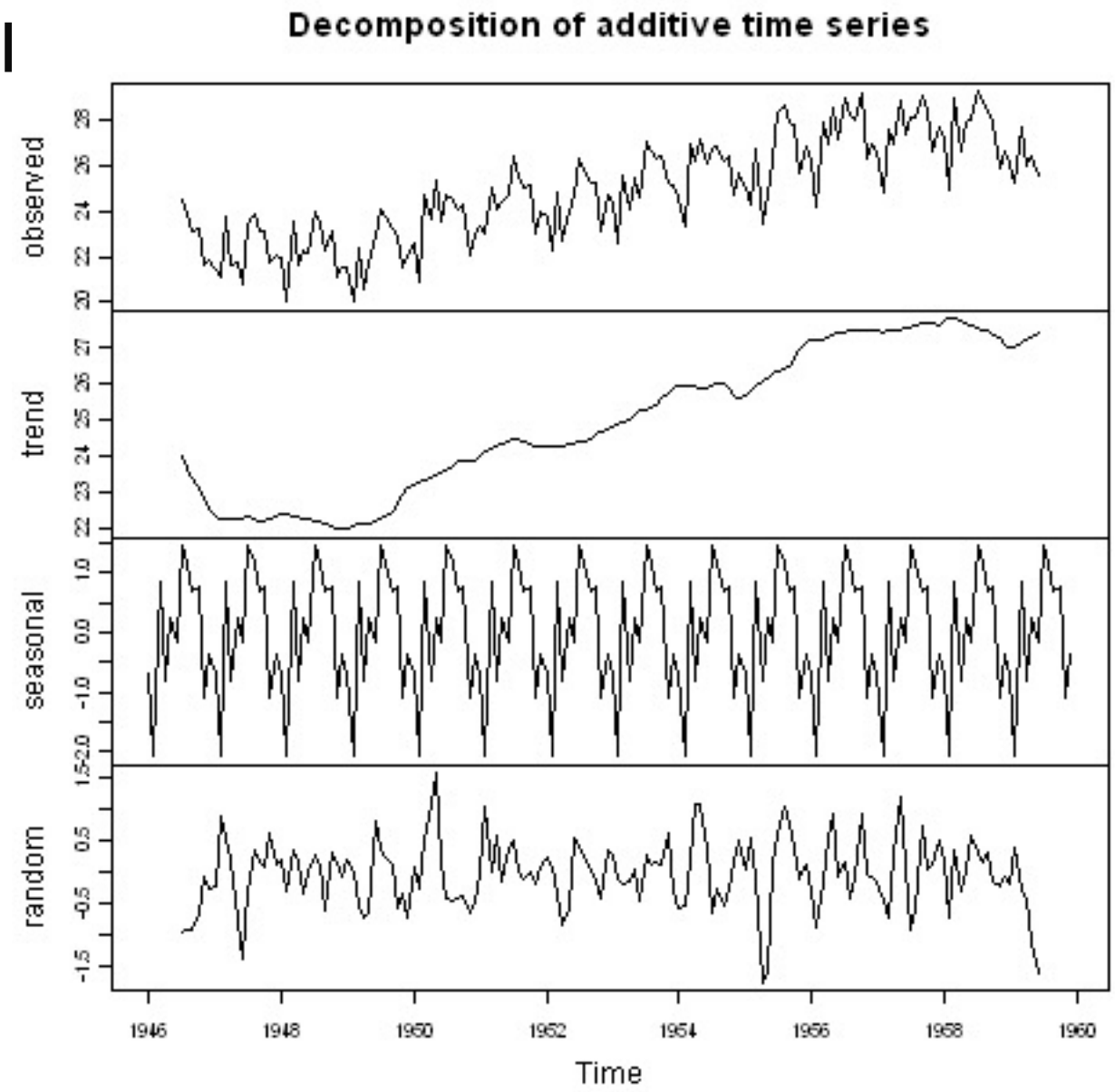
```
library(ggfortify)
autoplot(birthstimeseries)
```



Decomposing a seasonal ti

estimate the trend, seasonal and irregular components of this time series

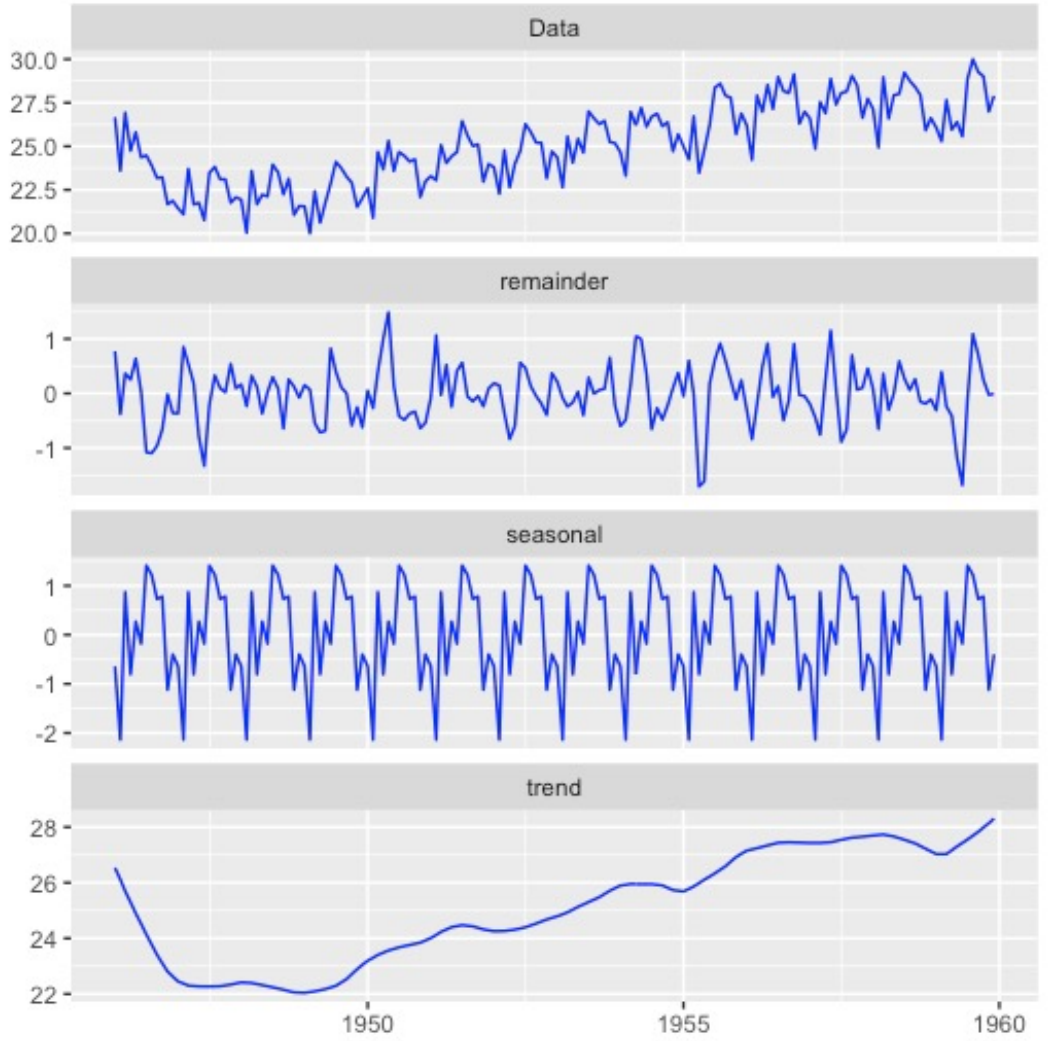
```
birthstimeseriescomponents <- decompose(birthstimeseries)
plot(birthstimeseriescomponents)
```



Decomposing a seasonal time-series

estimate the trend, seasonal and irregular components of this time series

```
autoplot(stl(birthstimeseries, s.window = 'periodic'),
ts.colour = 'blue')
```



COMMON FORMS OF ANALYSIS OF TIME SERIES

- time series smoothing models
  - moving average, weighted moving average, exponentially weighted moving average (EWMA)
- forecast models (forecast package)
  - autoregressive model (AR)
  - autoregressive integrated moving average (ARIMA)
- changepoint models (changepoint package)
  - mean shift
  - variance shift
- autocorrelation functions
  - ACF
  - PACF

All have different packages, classes, etc. but can be easily visualizaed using ggplot with help of **ggfortify** package

https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html

CASE STUDY – REVIEW

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0165688

https://www.sciencedirect.com/science/article/abs/pii/S0304380018302333

GG 501 SPATIAL KNOWLEDGE MOBILIZATION

1. Jan 18: Time series visualization