Introduction to Time Series Analysis and Forecasting in R

Tejendra Pratap Singh

2019-08-19

Contents

Chapter 1

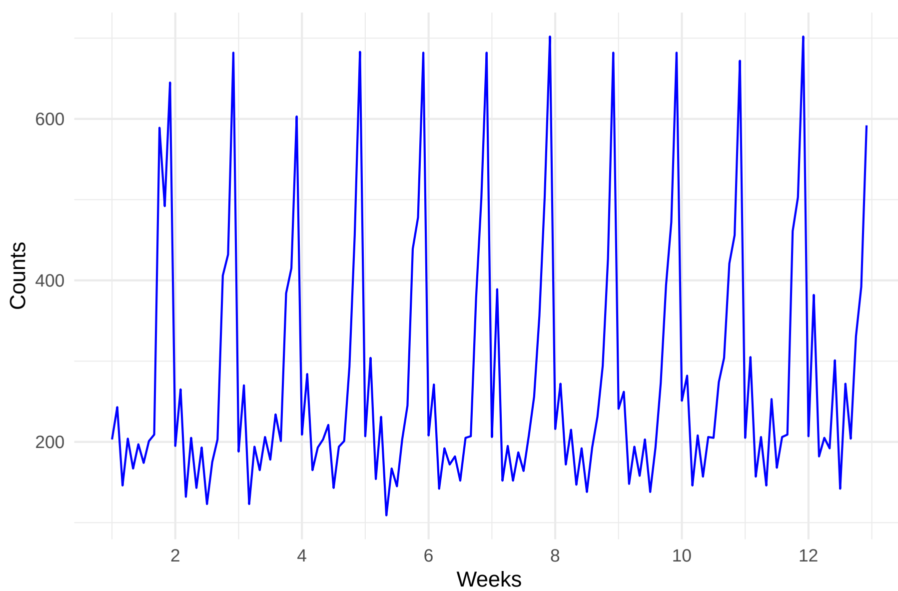
Introduction

```r
# on publishing the website see this
# https://bookdown.org/__docs__/user/publishing.html#publishing
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section1")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
```

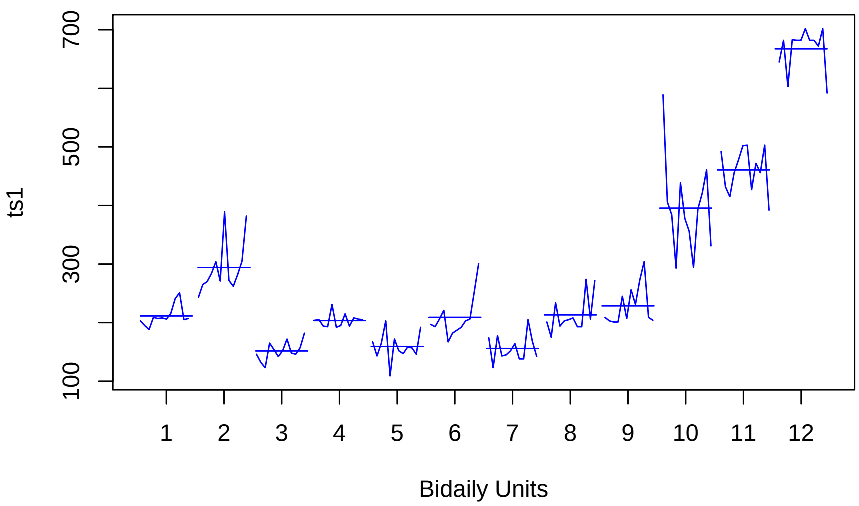
```r
# loading the data
ts1 <- read_delim("ITstore-bidaily.csv", ";", escape_double = FALSE,
    col_names = FALSE, trim_ws = TRUE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
```

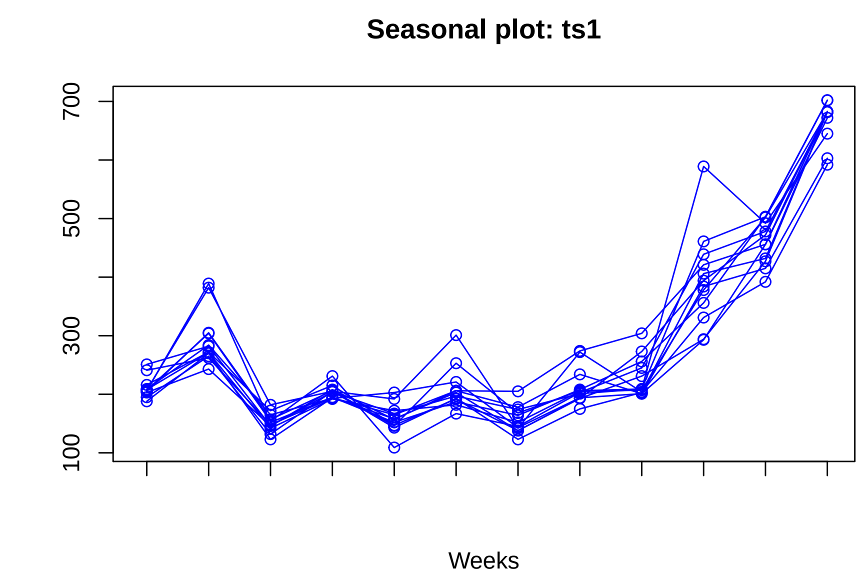
```r
# declaring the data as time series
ts1 <- ts(ts1$X2, start = 1, frequency = 12, class = "ts")
# visualizing the time series
theme_set(theme_minimal())
autoplot(ts1, color = "blue") + xlab("Weeks") + ylab("Counts")
```




```r
monthplot(ts1, labels = 1:12, xlab = "Bidaily Units", col = "blue")
```



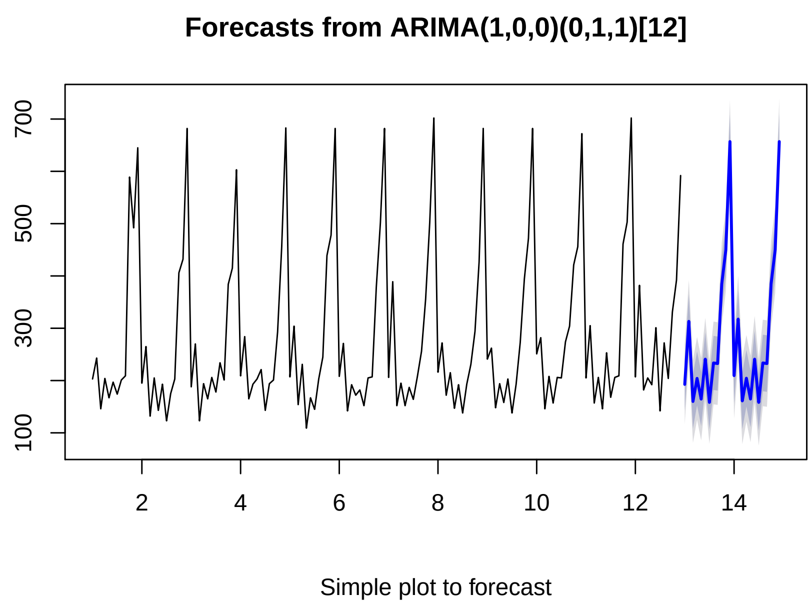

```
seasonplot(ts1, season.labels = FALSE, xlab = "Weeks", col = "blue")
```



Selecting the model.

Due to seasonality involved, simple models will not be able to capture it. We therefore use the seasonal ARIMA and exponential smoothing models. Exponential smoothing models have seasonality built in it by construction. Complex models like mixed models and neural nets will be an overkill.

```
# simple plot to see seasonality
plot(forecast(auto.arima(ts1)), sub = "Simple plot to forecast")
```



Chapter 2

Working With Dates And Time in R

```
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section2")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
```

```
# exploring the packages
OlsonNames()
```

```
##   [1] "Africa/Abidjan"                   "Africa/Accra"
##   [3] "Africa/Addis_Ababa"               "Africa/Algiers"
##   [5] "Africa/Asmara"                    "Africa/Asmera"
##   [7] "Africa/Bamako"                    "Africa/Bangui"
##   [9] "Africa/Banjul"                    "Africa/Bissau"
##  [11] "Africa/Blantyre"                  "Africa/Brazzaville"
```

```
##  [13] "Africa/Bujumbura"                 "Africa/Cairo"
##  [15] "Africa/Casablanca"                "Africa/Ceuta"
##  [17] "Africa/Conakry"                   "Africa/Dakar"
##  [19] "Africa/Dar_es_Salaam"             "Africa/Djibouti"
##  [21] "Africa/Douala"                    "Africa/El_Aaiun"
##  [23] "Africa/Freetown"                  "Africa/Gaborone"
##  [25] "Africa/Harare"                    "Africa/Johannesburg"
##  [27] "Africa/Juba"                      "Africa/Kampala"
##  [29] "Africa/Khartoum"                  "Africa/Kigali"
##  [31] "Africa/Kinshasa"                  "Africa/Lagos"
##  [33] "Africa/Libreville"                "Africa/Lome"
##  [35] "Africa/Luanda"                    "Africa/Lubumbashi"
##  [37] "Africa/Lusaka"                    "Africa/Malabo"
##  [39] "Africa/Maputo"                    "Africa/Maseru"
##  [41] "Africa/Mbabane"                   "Africa/Mogadishu"
##  [43] "Africa/Monrovia"                  "Africa/Nairobi"
##  [45] "Africa/Ndjamena"                  "Africa/Niamey"
##  [47] "Africa/Nouakchott"                "Africa/Ouagadougou"
##  [49] "Africa/Porto-Novo"                "Africa/Sao_Tome"
##  [51] "Africa/Timbuktu"                  "Africa/Tripoli"
##  [53] "Africa/Tunis"                     "Africa/Windhoek"
##  [55] "America/Adak"                     "America/Anchorage"
##  [57] "America/Anguilla"                 "America/Antigua"
##  [59] "America/Araguaina"                "America/Argentina/Buenos_Aires"
##  [61] "America/Argentina/Catamarca"      "America/Argentina/ComodRivadavia"
##  [63] "America/Argentina/Cordoba"        "America/Argentina/Jujuy"
##  [65] "America/Argentina/La_Rioja"       "America/Argentina/Mendoza"
##  [67] "America/Argentina/Rio_Gallegos"   "America/Argentina/Salta"
##  [69] "America/Argentina/San_Juan"       "America/Argentina/San_Luis"
##  [71] "America/Argentina/Tucuman"        "America/Argentina/Ushuaia"
##  [73] "America/Aruba"                    "America/Asuncion"
##  [75] "America/Atikokan"                 "America/Atka"
##  [77] "America/Bahia"                    "America/Bahia_Banderas"
##  [79] "America/Barbados"                 "America/Belem"
##  [81] "America/Belize"                   "America/Blanc-Sablon"
##  [83] "America/Boa_Vista"                "America/Bogota"
##  [85] "America/Boise"                    "America/Buenos_Aires"
##  [87] "America/Cambridge_Bay"            "America/Campo_Grande"
##  [89] "America/Cancun"                   "America/Caracas"
##  [91] "America/Catamarca"                "America/Cayenne"
##  [93] "America/Cayman"                   "America/Chicago"
##  [95] "America/Chihuahua"                "America/Coral_Harbour"
##  [97] "America/Cordoba"                  "America/Costa_Rica"
##  [99] "America/Creston"                  "America/Cuiaba"
## [101] "America/Curacao"                  "America/Danmarkshavn"
## [103] "America/Dawson"                   "America/Dawson_Creek"
```

```
## [105] "America/Denver"                   "America/Detroit"
## [107] "America/Dominica"                 "America/Edmonton"
## [109] "America/Eirunepe"                 "America/El_Salvador"
## [111] "America/Ensenada"                 "America/Fort_Nelson"
## [113] "America/Fort_Wayne"               "America/Fortaleza"
## [115] "America/Glace_Bay"                "America/Godthab"
## [117] "America/Goose_Bay"                "America/Grand_Turk"
## [119] "America/Grenada"                  "America/Guadeloupe"
## [121] "America/Guatemala"                "America/Guayaquil"
## [123] "America/Guyana"                   "America/Halifax"
## [125] "America/Havana"                   "America/Hermosillo"
## [127] "America/Indiana/Indianapolis"     "America/Indiana/Knox"
## [129] "America/Indiana/Marengo"          "America/Indiana/Petersburg"
## [131] "America/Indiana/Tell_City"        "America/Indiana/Vevay"
## [133] "America/Indiana/Vincennes"        "America/Indiana/Winamac"
## [135] "America/Indianapolis"             "America/Inuvik"
## [137] "America/Iqaluit"                  "America/Jamaica"
## [139] "America/Jujuy"                    "America/Juneau"
## [141] "America/Kentucky/Louisville"      "America/Kentucky/Monticello"
## [143] "America/Knox_IN"                  "America/Kralendijk"
## [145] "America/La_Paz"                   "America/Lima"
## [147] "America/Los_Angeles"              "America/Louisville"
## [149] "America/Lower_Princes"            "America/Maceio"
## [151] "America/Managua"                  "America/Manaus"
## [153] "America/Marigot"                  "America/Martinique"
## [155] "America/Matamoros"                "America/Mazatlan"
## [157] "America/Mendoza"                  "America/Menominee"
## [159] "America/Merida"                   "America/Metlakatla"
## [161] "America/Mexico_City"              "America/Miquelon"
## [163] "America/Moncton"                  "America/Monterrey"
## [165] "America/Montevideo"               "America/Montreal"
## [167] "America/Montserrat"               "America/Nassau"
## [169] "America/New_York"                 "America/Nipigon"
## [171] "America/Nome"                     "America/Noronha"
## [173] "America/North_Dakota/Beulah"      "America/North_Dakota/Center"
## [175] "America/North_Dakota/New_Salem"   "America/Ojinaga"
## [177] "America/Panama"                   "America/Pangnirtung"
## [179] "America/Paramaribo"               "America/Phoenix"
## [181] "America/Port-au-Prince"           "America/Port_of_Spain"
## [183] "America/Porto_Acre"               "America/Porto_Velho"
## [185] "America/Puerto_Rico"              "America/Punta_Arenas"
## [187] "America/Rainy_River"              "America/Rankin_Inlet"
## [189] "America/Recife"                   "America/Regina"
## [191] "America/Resolute"                 "America/Rio_Branco"
## [193] "America/Rosario"                  "America/Santa_Isabel"
## [195] "America/Santarem"                 "America/Santiago"
```

```
## [197] "America/Santo_Domingo"            "America/Sao_Paulo"
## [199] "America/Scoresbysund"             "America/Shiprock"
## [201] "America/Sitka"                    "America/St_Barthelemy"
## [203] "America/St_Johns"                 "America/St_Kitts"
## [205] "America/St_Lucia"                 "America/St_Thomas"
## [207] "America/St_Vincent"               "America/Swift_Current"
## [209] "America/Tegucigalpa"              "America/Thule"
## [211] "America/Thunder_Bay"              "America/Tijuana"
## [213] "America/Toronto"                  "America/Tortola"
## [215] "America/Vancouver"                "America/Virgin"
## [217] "America/Whitehorse"               "America/Winnipeg"
## [219] "America/Yakutat"                  "America/Yellowknife"
## [221] "Antarctica/Casey"                 "Antarctica/Davis"
## [223] "Antarctica/DumontDUrville"        "Antarctica/Macquarie"
## [225] "Antarctica/Mawson"                "Antarctica/McMurdo"
## [227] "Antarctica/Palmer"                "Antarctica/Rothera"
## [229] "Antarctica/South_Pole"            "Antarctica/Syowa"
## [231] "Antarctica/Troll"                 "Antarctica/Vostok"
## [233] "Arctic/Longyearbyen"              "Asia/Aden"
## [235] "Asia/Almaty"                      "Asia/Amman"
## [237] "Asia/Anadyr"                      "Asia/Aqtau"
## [239] "Asia/Aqtobe"                      "Asia/Ashgabat"
## [241] "Asia/Ashkhabad"                   "Asia/Atyrau"
## [243] "Asia/Baghdad"                     "Asia/Bahrain"
## [245] "Asia/Baku"                        "Asia/Bangkok"
## [247] "Asia/Barnaul"                     "Asia/Beirut"
## [249] "Asia/Bishkek"                     "Asia/Brunei"
## [251] "Asia/Calcutta"                    "Asia/Chita"
## [253] "Asia/Choibalsan"                  "Asia/Chongqing"
## [255] "Asia/Chungking"                   "Asia/Colombo"
## [257] "Asia/Dacca"                       "Asia/Damascus"
## [259] "Asia/Dhaka"                       "Asia/Dili"
## [261] "Asia/Dubai"                       "Asia/Dushanbe"
## [263] "Asia/Famagusta"                   "Asia/Gaza"
## [265] "Asia/Harbin"                      "Asia/Hebron"
## [267] "Asia/Ho_Chi_Minh"                 "Asia/Hong_Kong"
## [269] "Asia/Hovd"                        "Asia/Irkutsk"
## [271] "Asia/Istanbul"                    "Asia/Jakarta"
## [273] "Asia/Jayapura"                    "Asia/Jerusalem"
## [275] "Asia/Kabul"                       "Asia/Kamchatka"
## [277] "Asia/Karachi"                     "Asia/Kashgar"
## [279] "Asia/Kathmandu"                   "Asia/Katmandu"
## [281] "Asia/Khandyga"                    "Asia/Kolkata"
## [283] "Asia/Krasnoyarsk"                 "Asia/Kuala_Lumpur"
## [285] "Asia/Kuching"                     "Asia/Kuwait"
## [287] "Asia/Macao"                       "Asia/Macau"
```

```
## [289] "Asia/Magadan"                    "Asia/Makassar"
## [291] "Asia/Manila"                     "Asia/Muscat"
## [293] "Asia/Nicosia"                    "Asia/Novokuznetsk"
## [295] "Asia/Novosibirsk"                "Asia/Omsk"
## [297] "Asia/Oral"                       "Asia/Phnom_Penh"
## [299] "Asia/Pontianak"                  "Asia/Pyongyang"
## [301] "Asia/Qatar"                      "Asia/Qostanay"
## [303] "Asia/Qyzylorda"                  "Asia/Rangoon"
## [305] "Asia/Riyadh"                     "Asia/Saigon"
## [307] "Asia/Sakhalin"                   "Asia/Samarkand"
## [309] "Asia/Seoul"                      "Asia/Shanghai"
## [311] "Asia/Singapore"                  "Asia/Srednekolymsk"
## [313] "Asia/Taipei"                     "Asia/Tashkent"
## [315] "Asia/Tbilisi"                    "Asia/Tehran"
## [317] "Asia/Tel_Aviv"                   "Asia/Thimbu"
## [319] "Asia/Thimphu"                    "Asia/Tokyo"
## [321] "Asia/Tomsk"                      "Asia/Ujung_Pandang"
## [323] "Asia/Ulaanbaatar"                "Asia/Ulan_Bator"
## [325] "Asia/Urumqi"                     "Asia/Ust-Nera"
## [327] "Asia/Vientiane"                  "Asia/Vladivostok"
## [329] "Asia/Yakutsk"                    "Asia/Yangon"
## [331] "Asia/Yekaterinburg"              "Asia/Yerevan"
## [333] "Atlantic/Azores"                 "Atlantic/Bermuda"
## [335] "Atlantic/Canary"                 "Atlantic/Cape_Verde"
## [337] "Atlantic/Faeroe"                 "Atlantic/Faroe"
## [339] "Atlantic/Jan_Mayen"              "Atlantic/Madeira"
## [341] "Atlantic/Reykjavik"              "Atlantic/South_Georgia"
## [343] "Atlantic/St_Helena"              "Atlantic/Stanley"
## [345] "Australia/ACT"                   "Australia/Adelaide"
## [347] "Australia/Brisbane"              "Australia/Broken_Hill"
## [349] "Australia/Canberra"              "Australia/Currie"
## [351] "Australia/Darwin"                "Australia/Eucla"
## [353] "Australia/Hobart"                "Australia/LHI"
## [355] "Australia/Lindeman"              "Australia/Lord_Howe"
## [357] "Australia/Melbourne"             "Australia/North"
## [359] "Australia/NSW"                   "Australia/Perth"
## [361] "Australia/Queensland"            "Australia/South"
## [363] "Australia/Sydney"                "Australia/Tasmania"
## [365] "Australia/Victoria"              "Australia/West"
## [367] "Australia/Yancowinna"            "Brazil/Acre"
## [369] "Brazil/DeNoronha"                "Brazil/East"
## [371] "Brazil/West"                     "Canada/Atlantic"
## [373] "Canada/Central"                  "Canada/Eastern"
## [375] "Canada/Mountain"                 "Canada/Newfoundland"
## [377] "Canada/Pacific"                  "Canada/Saskatchewan"
## [379] "Canada/Yukon"                    "CET"
```

```
## [381] "Chile/Continental"                "Chile/EasterIsland"
## [383] "CST6CDT"                          "Cuba"
## [385] "EET"                              "Egypt"
## [387] "Eire"                             "EST"
## [389] "EST5EDT"                          "Etc/GMT"
## [391] "Etc/GMT-0"                        "Etc/GMT-1"
## [393] "Etc/GMT-10"                       "Etc/GMT-11"
## [395] "Etc/GMT-12"                       "Etc/GMT-13"
## [397] "Etc/GMT-14"                       "Etc/GMT-2"
## [399] "Etc/GMT-3"                        "Etc/GMT-4"
## [401] "Etc/GMT-5"                        "Etc/GMT-6"
## [403] "Etc/GMT-7"                        "Etc/GMT-8"
## [405] "Etc/GMT-9"                        "Etc/GMT+0"
## [407] "Etc/GMT+1"                        "Etc/GMT+10"
## [409] "Etc/GMT+11"                       "Etc/GMT+12"
## [411] "Etc/GMT+2"                        "Etc/GMT+3"
## [413] "Etc/GMT+4"                        "Etc/GMT+5"
## [415] "Etc/GMT+6"                        "Etc/GMT+7"
## [417] "Etc/GMT+8"                        "Etc/GMT+9"
## [419] "Etc/GMT0"                         "Etc/Greenwich"
## [421] "Etc/UCT"                          "Etc/Universal"
## [423] "Etc/UTC"                          "Etc/Zulu"
## [425] "Europe/Amsterdam"                 "Europe/Andorra"
## [427] "Europe/Astrakhan"                 "Europe/Athens"
## [429] "Europe/Belfast"                   "Europe/Belgrade"
## [431] "Europe/Berlin"                    "Europe/Bratislava"
## [433] "Europe/Brussels"                  "Europe/Bucharest"
## [435] "Europe/Budapest"                  "Europe/Busingen"
## [437] "Europe/Chisinau"                  "Europe/Copenhagen"
## [439] "Europe/Dublin"                    "Europe/Gibraltar"
## [441] "Europe/Guernsey"                  "Europe/Helsinki"
## [443] "Europe/Isle_of_Man"               "Europe/Istanbul"
## [445] "Europe/Jersey"                    "Europe/Kaliningrad"
## [447] "Europe/Kiev"                      "Europe/Kirov"
## [449] "Europe/Lisbon"                    "Europe/Ljubljana"
## [451] "Europe/London"                    "Europe/Luxembourg"
## [453] "Europe/Madrid"                    "Europe/Malta"
## [455] "Europe/Mariehamn"                 "Europe/Minsk"
## [457] "Europe/Monaco"                    "Europe/Moscow"
## [459] "Europe/Nicosia"                   "Europe/Oslo"
## [461] "Europe/Paris"                     "Europe/Podgorica"
## [463] "Europe/Prague"                    "Europe/Riga"
## [465] "Europe/Rome"                      "Europe/Samara"
## [467] "Europe/San_Marino"                "Europe/Sarajevo"
## [469] "Europe/Saratov"                   "Europe/Simferopol"
## [471] "Europe/Skopje"                    "Europe/Sofia"
```

```
## [473] "Europe/Stockholm"                "Europe/Tallinn"
## [475] "Europe/Tirane"                   "Europe/Tiraspol"
## [477] "Europe/Ulyanovsk"                "Europe/Uzhgorod"
## [479] "Europe/Vaduz"                    "Europe/Vatican"
## [481] "Europe/Vienna"                   "Europe/Vilnius"
## [483] "Europe/Volgograd"                "Europe/Warsaw"
## [485] "Europe/Zagreb"                   "Europe/Zaporozhye"
## [487] "Europe/Zurich"                   "GB"
## [489] "GB-Eire"                         "GMT"
## [491] "GMT-0"                           "GMT+0"
## [493] "GMT0"                            "Greenwich"
## [495] "Hongkong"                        "HST"
## [497] "Iceland"                         "Indian/Antananarivo"
## [499] "Indian/Chagos"                   "Indian/Christmas"
## [501] "Indian/Cocos"                    "Indian/Comoro"
## [503] "Indian/Kerguelen"                "Indian/Mahe"
## [505] "Indian/Maldives"                 "Indian/Mauritius"
## [507] "Indian/Mayotte"                  "Indian/Reunion"
## [509] "Iran"                            "Israel"
## [511] "Jamaica"                         "Japan"
## [513] "Kwajalein"                       "Libya"
## [515] "MET"                             "Mexico/BajaNorte"
## [517] "Mexico/BajaSur"                  "Mexico/General"
## [519] "MST"                             "MST7MDT"
## [521] "Navajo"                          "NZ"
## [523] "NZ-CHAT"                         "Pacific/Apia"
## [525] "Pacific/Auckland"                "Pacific/Bougainville"
## [527] "Pacific/Chatham"                 "Pacific/Chuuk"
## [529] "Pacific/Easter"                  "Pacific/Efate"
## [531] "Pacific/Enderbury"               "Pacific/Fakaofo"
## [533] "Pacific/Fiji"                    "Pacific/Funafuti"
## [535] "Pacific/Galapagos"               "Pacific/Gambier"
## [537] "Pacific/Guadalcanal"             "Pacific/Guam"
## [539] "Pacific/Honolulu"                "Pacific/Johnston"
## [541] "Pacific/Kiritimati"              "Pacific/Kosrae"
## [543] "Pacific/Kwajalein"               "Pacific/Majuro"
## [545] "Pacific/Marquesas"               "Pacific/Midway"
## [547] "Pacific/Nauru"                   "Pacific/Niue"
## [549] "Pacific/Norfolk"                 "Pacific/Noumea"
## [551] "Pacific/Pago_Pago"               "Pacific/Palau"
## [553] "Pacific/Pitcairn"                "Pacific/Pohnpei"
## [555] "Pacific/Ponape"                  "Pacific/Port_Moresby"
## [557] "Pacific/Rarotonga"               "Pacific/Saipan"
## [559] "Pacific/Samoa"                   "Pacific/Tahiti"
## [561] "Pacific/Tarawa"                  "Pacific/Tongatapu"
## [563] "Pacific/Truk"                    "Pacific/Wake"
```

```
## [565] "Pacific/Wallis"                    "Pacific/Yap"
## [567] "Poland"                            "Portugal"
## [569] "PRC"                               "PST8PDT"
## [571] "ROC"                               "ROK"
## [573] "Singapore"                         "Turkey"
## [575] "UCT"                               "Universal"
## [577] "US/Alaska"                         "US/Aleutian"
## [579] "US/Arizona"                        "US/Central"
## [581] "US/East-Indiana"                   "US/Eastern"
## [583] "US/Hawaii"                         "US/Indiana-Starke"
## [585] "US/Michigan"                       "US/Mountain"
## [587] "US/Pacific"                        "US/Pacific-New"
## [589] "US/Samoa"                          "UTC"
## [591] "W-SU"                              "WET"
## [593] "Zulu"
## attr(,"Version")
## [1] "2019a"
```

```
### POSIXt classes in R

x = as.POSIXct("2019-12-25 11:45:34")  # nr of seconds
y = as.POSIXlt("2019-12-25 11:45:34")
x
```

```
## [1] "2019-12-25 11:45:34 IST"
```

```
y  # it gives the same output, but what is behind it?
```

```
## [1] "2019-12-25 11:45:34 IST"
```

```
unclass(x)
```

```
## [1] 1577254534
## attr(,"tzone")
## [1] ""
```

```
unclass(y)
```

```
## $sec
## [1] 34
##
## $min
## [1] 45
```

```
##
## $hour
## [1] 11
##
## $mday
## [1] 25
##
## $mon
## [1] 11
##
## $year
## [1] 119
##
## $wday
## [1] 3
##
## $yday
## [1] 358
##
## $isdst
## [1] 0
##
## $zone
## [1] "IST"
##
## $gmtoff
## [1] NA
```

```r
# what does the number mean?
(50 * 365 * 24 * 60 * 60) - (5.5 * 60 * 60)
```

```
## [1] 1576780200
```

```r
y$zone  # extracting the elements from POSIXlt
```

```
## [1] "IST"
```

```r
x$zone  # not possible since it is simply a number of seconds
```

```
## Error in x$zone: $ operator is invalid for atomic vectors
```

```r
# another class based on days
x = as.Date("2019-12-25")
x
```

```
## [1] "2019-12-25"
```

```r
class(x)
```

```
## [1] "Date"
```

```r
unclass(x)
```

```
## [1] 18255
```

```r
50 * 365 - 5  # nr of days since 1970
```

```
## [1] 18245
```

```r
x = chron("12/25/2019", "23:34:09")
x
```

```
## [1] (12/25/19 23:34:09)
```

```r
class(x)
```

```
## [1] "chron" "dates" "times"
```

```r
unclass(x)
```

```
## [1] 18255.98
## attr(,"format")
##   dates   times
## "m/d/y" "h:m:s"
## attr(,"origin")
## month   day  year
##     1     1  1970
```

```r
### strptime
a = as.character(c("1993-12-30 23:45", "1994-11-05 11:43", "1992-03-09 21:54"))
class(a)
```

```
## [1] "character"
```

```r
b = strptime(a, format = "%Y-%m-%d %H:%M")
b
```

```
## [1] "1993-12-30 23:45:00 IST" "1994-11-05 11:43:00 IST"
## [3] "1992-03-09 21:54:00 IST"
```

```r
class(b)
```

```
## [1] "POSIXlt" "POSIXt"
```

```r
### Lets take a look at the package lubridate which has very
### useful time/date data functions different ways in how to
### input dates
ymd(19931123)
```

```
## [1] "1993-11-23"
```

```r
dmy(23111993)
```

```
## [1] "1993-11-23"
```

```r
mdy(11231993)
```

```
## [1] "1993-11-23"
```

```r
# lets use time and date together
mytimepoint <- ymd_hm("1993-11-23 11:23", tz = "Europe/Prague")
mytimepoint
```

```
## [1] "1993-11-23 11:23:00 CET"
```

```r
class(mytimepoint)
```

```
## [1] "POSIXct" "POSIXt"
```

```r
# extracting the components of it
minute(mytimepoint)
```

```
## [1] 23
```

```r
day(mytimepoint)
```

```
## [1] 23
```

```r
hour(mytimepoint)
```

```
## [1] 11
```

```r
year(mytimepoint)
```

```
## [1] 1993
```

```r
month(mytimepoint)
```

```
## [1] 11
```

```r
# we can even change time values within our object
hour(mytimepoint) <- 14
mytimepoint
```

```
## [1] "1993-11-23 14:23:00 CET"
```

```r
# we can take a look at the most common time zones but be
# aware that the time zone recognition also depends on your
# location and machine lets check which day our time point is
wday(mytimepoint)
```

```
## [1] 3
```

```r
wday(mytimepoint, label = T, abbr = F)  # label to display the name of the day, no abb
```

```
## [1] Tuesday
## 7 Levels: Sunday < Monday < Tuesday < Wednesday < Thursday < ... < Saturday
```

```r
# we can calculate which time our timepoint would be in
# another time zone
with_tz(mytimepoint, tz = "Europe/London")
```

```
## [1] "1993-11-23 13:23:00 GMT"
```

```
mytimepoint
```

```
## [1] "1993-11-23 14:23:00 CET"
```

```
# time intervals
time1 = ymd_hm("1993-09-23 11:23", tz = "Europe/Prague")
time2 = ymd_hm("1995-11-02 15:23", tz = "Europe/Prague")

# getting the interval
myinterval = interval(time1, time2)
myinterval
```

```
## [1] 1993-09-23 11:23:00 CEST--1995-11-02 15:23:00 CET
```

```
class(myinterval)  # interval is an object class from lubridate
```

```
## [1] "Interval"
## attr(,"package")
## [1] "lubridate"
```

```
### Exercise: Creating a Data Frame with lubridate lets now
### build a dataframe with lubridate that contains date and
### time data see the different input formats that are allowed
### in the ymd function

a = c("1998,11,11", "1983/01/23", "1982:09:04", "1945-05-09",
    19821224, "1974.12.03", 19871210)
a = ymd(a, tz = "CET")
a
```

```
## [1] "1998-11-11 CET"  "1983-01-23 CET"  "1982-09-04 CEST" "1945-05-09 CEST"
## [5] "1982-12-24 CET"  "1974-12-03 CET"  "1987-12-10 CET"
```

```
# now I am creating a time vector - using different notations
# of input
b = c("22 4 5", "04;09;45", "11:9:56", "23,15,12", "14 16 34",
    "8 8 23", "21 16 14")
b = hms(b)
b
```

```
## [1] "22H 4M 5S"   "4H 9M 45S"   "11H 9M 56S"  "23H 15M 12S" "14H 16M 34S"
## [6] "8H 8M 23S"   "21H 16M 14S"
```

```
f = rnorm(7, 10)
f = round(f, digits = 2)
f
```

```
## [1] 10.47  8.76 10.89  8.15  9.93 10.00 10.98
```

```
date_time_measurement = cbind.data.frame(date = a, time = b,
    measurement = f)
date_time_measurement
```

```
##         date        time measurement
## 1 1998-11-11   22H 4M 5S       10.47
## 2 1983-01-23   4H 9M 45S        8.76
## 3 1982-09-04  11H 9M 56S       10.89
## 4 1945-05-09 23H 15M 12S        8.15
## 5 1982-12-24 14H 16M 34S        9.93
## 6 1974-12-03   8H 8M 23S       10.00
## 7 1987-12-10 21H 16M 14S       10.98
```

```
## Calculations with time
minutes(7)
```

```
## [1] "7M 0S"
```

```
# note that class 'Period' needs integers - full numbers
minutes(2.5)
```

```
## Error in validObject(.Object): invalid class "Period" object: periods must have inte
```

```
# getting the duration
dminutes(3)
```

```
## [1] "180s (~3 minutes)"
```

```
dminutes(3.5)
```

```
## [1] "210s (~3.5 minutes)"
```

```
# how to add minutes and seconds
minutes(2) + seconds(5)
```

```
## [1] "2M 5S"
```

```r
# more calculations
minutes(2) + seconds(75)
```

```
## [1] "2M 75S"
```

```r
# class 'duration' to perform addition
as.duration(minutes(2) + seconds(75))
```

```
## [1] "195s (~3.25 minutes)"
```

```r
# lubridate has many time classes: period or duration differ!
# which year was a leap year?

leap_year(2009:2014)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
```

```r
ymd(20140101) + years(1)
```

```
## [1] "2015-01-01"
```

```r
ymd(20140101) + dyears(1)
```

```
## [1] "2015-01-01"
```

```r
# lets do the whole thing with a leap year
leap_year(2016)
```

```
## [1] TRUE
```

```r
ymd(20160101) + years(1)
```

```
## [1] "2017-01-01"
```

```r
ymd(20160101) + dyears(1)
```

```
## [1] "2016-12-31"
```

```
# as you see the duration is the one which is always 365 days
# the standard one (the period) makes the year a whole new
# unit (+1)

## Exercise Lubridate create x, with time zone CET and a given
## time point in 2014 of your choosing I use '2014-04-12
## 23:12' the time point consists of year, months, day and
## hour change now the minute of x to 7 and check x in the
## same line of code see which time it would be in London
## create another time point y in 2015 and get the difference
## between those 2 points

x <- ymd_hm(c("2014-04-12 23:12"), tz = "CET")
minute(x) <- 7
x
```

```
## [1] "2014-04-12 23:07:00 CEST"
```

```
with_tz(x, tzone = "Europe/London")
```

```
## [1] "2014-04-12 22:07:00 BST"
```

```
y <- ymd_hm(c("2015-01-01 11:11"), tz = "CET")
y
```

```
## [1] "2015-01-01 11:11:00 CET"
```

```
y - x
```

```
## Time difference of 263.5444 days
```

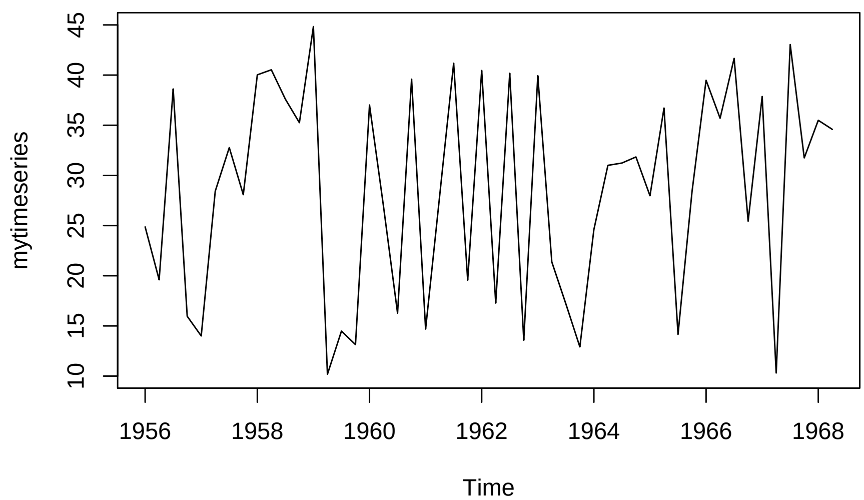
Chapter 3

Time Series Data Pre-Processing and Visualization

```r
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section3")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
```

```r
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section3")
## U Standard time series functions
# Lets create a time series object - class ts
# Getting data
mydata = runif(n = 50, min = 10, max = 45)

# ts for class time series
```

```r
# Data starts in 1956 - 4 observations/year (quarterly)
mytimeseries = ts(data = mydata,
                  start = 1956, frequency = 4)

# Lets see how the data looks
plot(mytimeseries)
```




```r
# Checking the class
class(mytimeseries)
```

```
## [1] "ts"
```

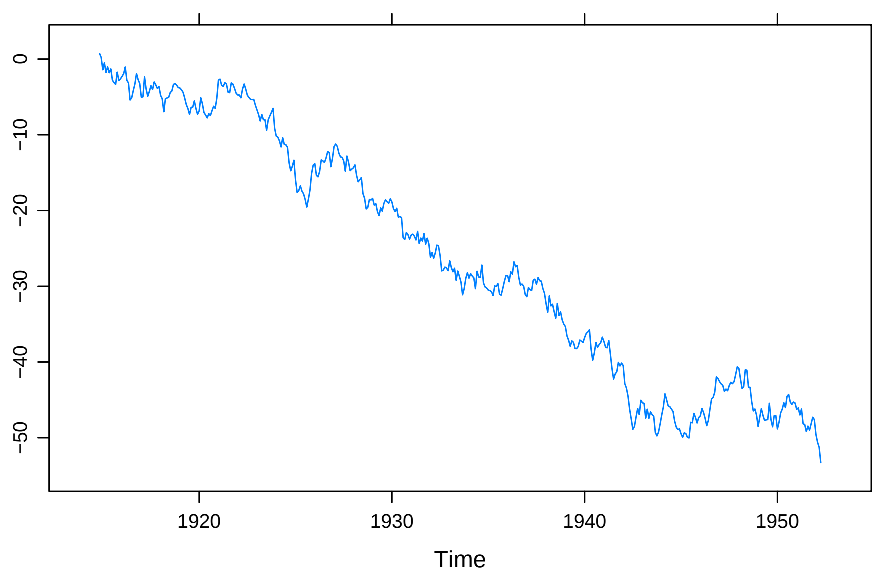
```r
# Checking the timestamp
time(mytimeseries)
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 1956 1956.00 1956.25 1956.50 1956.75
## 1957 1957.00 1957.25 1957.50 1957.75
## 1958 1958.00 1958.25 1958.50 1958.75
## 1959 1959.00 1959.25 1959.50 1959.75
## 1960 1960.00 1960.25 1960.50 1960.75
## 1961 1961.00 1961.25 1961.50 1961.75
## 1962 1962.00 1962.25 1962.50 1962.75
```

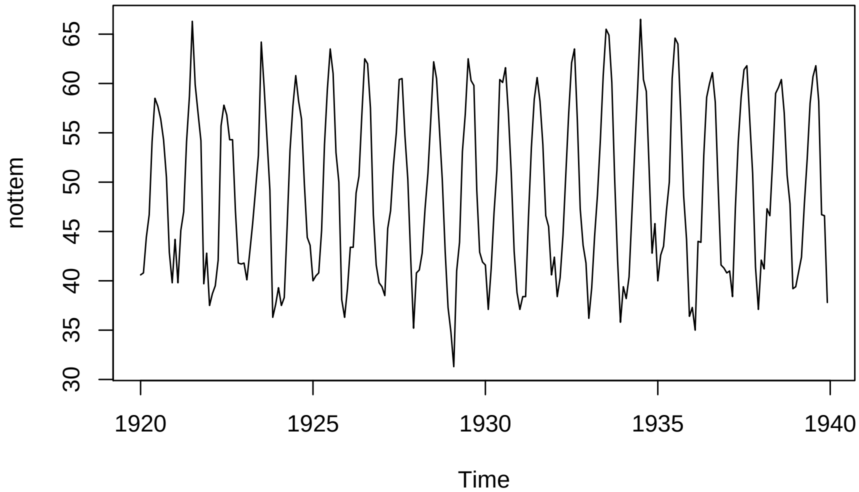
```
## 1963 1963.00 1963.25 1963.50 1963.75
## 1964 1964.00 1964.25 1964.50 1964.75
## 1965 1965.00 1965.25 1965.50 1965.75
## 1966 1966.00 1966.25 1966.50 1966.75
## 1967 1967.00 1967.25 1967.50 1967.75
## 1968 1968.00 1968.25
```

```r
# Refining the start argument
mytimeseries = ts(data = mydata,
                  start = c(1956,3), frequency = 4)
## Creating a ts object - Exercise
# Get a random walk of 450 numbers, eg rnorm, runif, etc
# In the solution I am going to use a cumulative sum on the normal distribution x = cumsum(rnorm(
# If you want it to be reproducible, you can set a seed
# Add the time component: it is a monthly dataset, which starts in November 1914
# Get a simple plot for this time series # Advanced: how would you get the same type with the "la

set.seed(2019)
ts1 <- cumsum(rnorm(n = 450))
ts1 <- ts(ts1,
          start = c(1914, 11),
          frequency = 12)
plot(ts1)
```

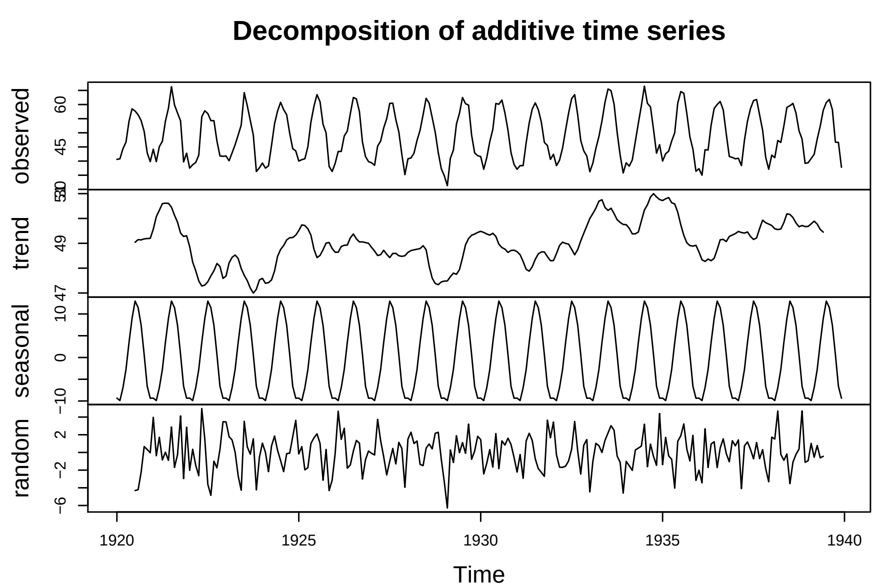
```
lattice::xyplot.ts(ts1)
```




```
## U Plots for time series data
# Standard R Base plots
plot(nottem)
```




```r
# Plot of components
plot(decompose(nottem))
```

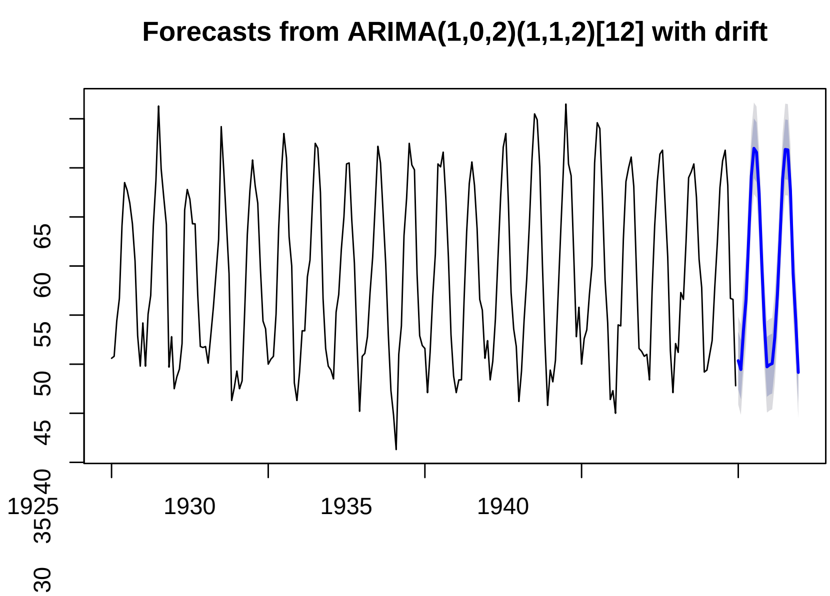


```r
# Directly plotting a forecast of a model
plot(forecast(auto.arima(nottem)), h = 5)
```

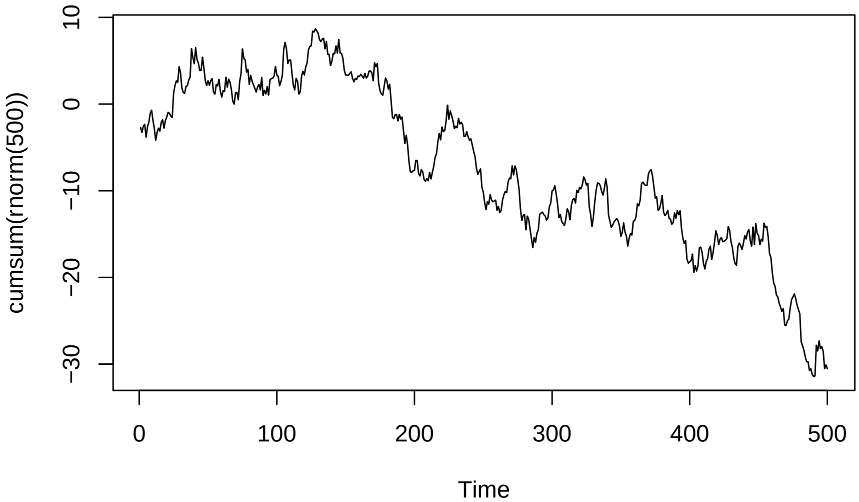
```
## Warning in plot.window(xlim, ylim, log, ...): "h" is not a graphical
## parameter
```

```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "h" is not a
## graphical parameter
```

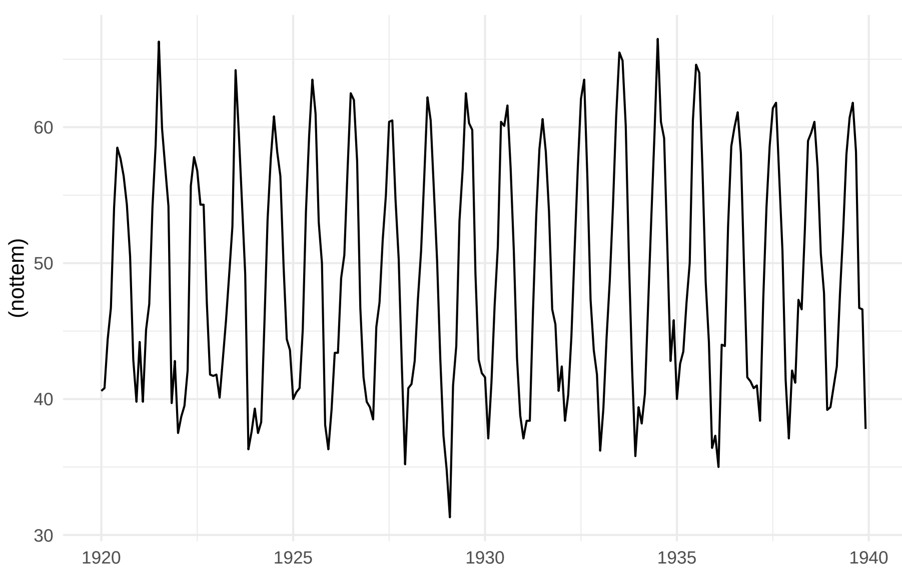
```
## Warning in box(...): "h" is not a graphical parameter
```



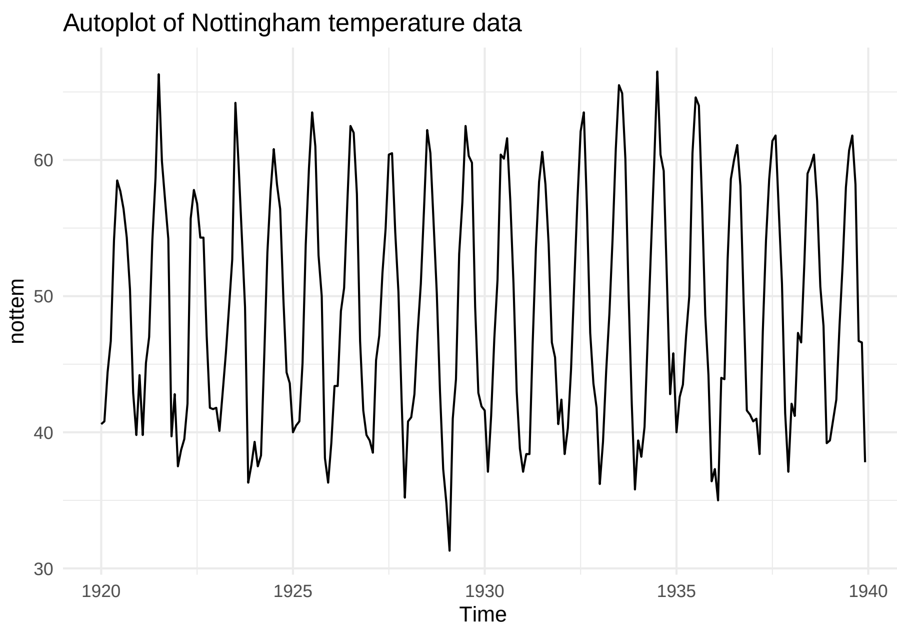

```r
# Random walk
plot.ts(cumsum(rnorm(500)))
```



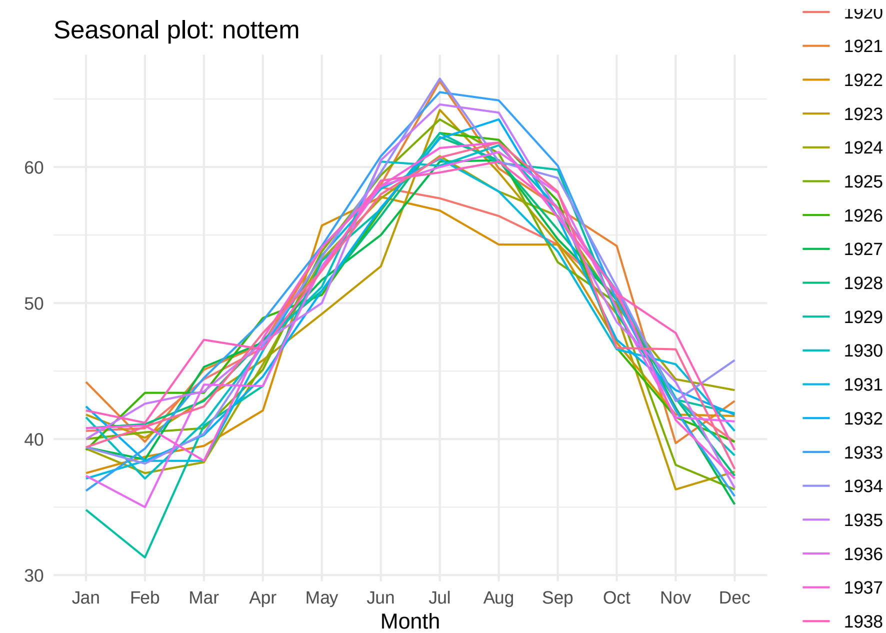

```r
# The ggplot equivalent to plot
autoplot((nottem))
```



```
# Ggplots work with different layers
autoplot(nottem) + ggtitle("Autoplot of Nottingham temperature data") +
  theme_minimal()
```




```
# Time series specific plots
ggseasonplot(nottem) + theme_minimal()
```

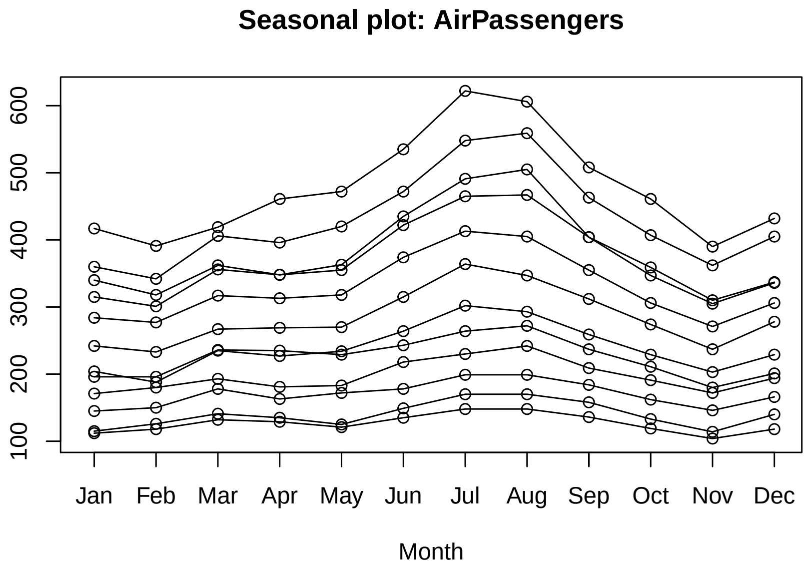



```r
ggmonthplot(nottem)
```

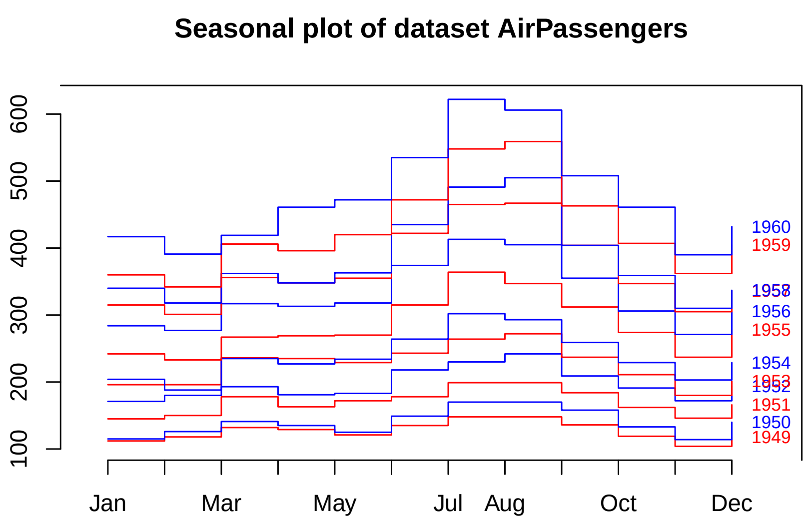
```
## Exercise Seasonplot - library (forecast)
# use the seasonplot function in order
# to resemble the plot shown here
# we are using the AirPassengers dataset
# for all the needed arguments on the plot,
# check out the help for par
# make sure that the labels are visible

AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

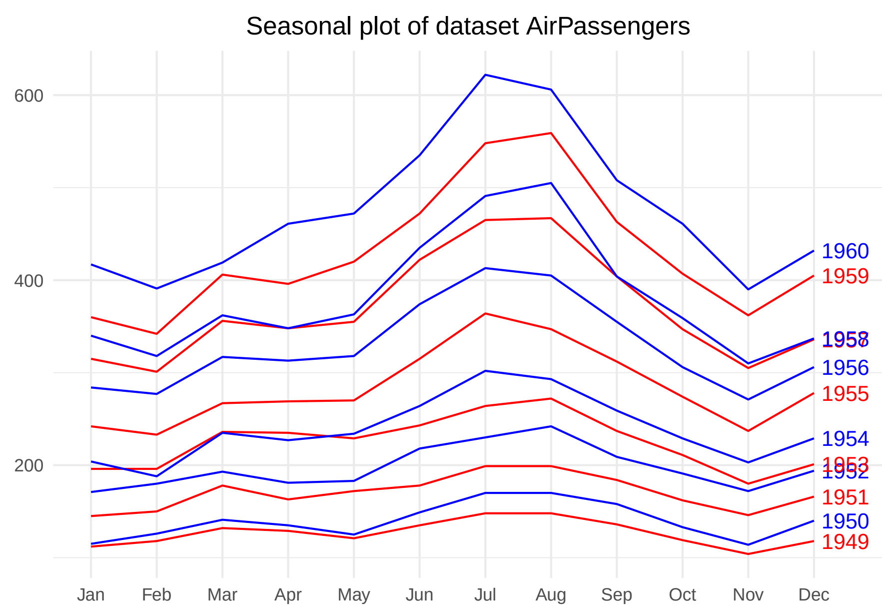
```
seasonplot(AirPassengers)
```




```r
seasonplot(AirPassengers, xlab = "", col = c("red", "blue"), year.labels = T, labelgap = 0.35, ty
```



```
theme_set(theme_minimal())
ggseasonplot(AirPassengers,
             year.labels = TRUE) +
  xlab("") +
  ggtitle("Seasonal plot of dataset AirPassengers") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values=rep(c("red", "blue"), 6))
```




```
## German Inflation Rates
#https://www.statbureau.org/en/germany/inflation-tables

# data is copied from web, first and last col deleted
#mydata = scan()
#plot.ts(mydata)
#germaninfl = ts(mydata, start = 2008, frequency = 12)
#plot(germaninfl)

### Working with Irregular Time Series
## dataset: irregular_sensor

## Method 1 - removing the time component
irregular_sensor <- read_csv("./Data/irregular-sensor.csv",
                             col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_double()
## )
```

```
# Irregular_sensor time series csv
class(irregular_sensor$X1)
```

```
## [1] "character"
```

```
#separate a column into two or more components
#using separate function from tidyr
irreg.split = tidyr::separate(irregular_sensor,
                              col = X1, #column to separate
                       into = c('date', 'time'), #new column names
                       sep = 8, # where to separate from
                       remove = T) #remove the original column after separation

# Using only the date
sensor.date = strptime(irreg.split$date,
                       '%m/%d/%y') #format of the sepapration

# Creating a data.frame for orientation
irregts.df = data.frame(date = as.Date(sensor.date),
                        measurement = irregular_sensor$X2)
irregts.df = data_frame(1:25) %>%
  mutate(date = as.Date(sensor.date),
         measurement = irregular_sensor$X2) %>%
  select(-`1:25`) %>%
  group_by(date) %>%
  mutate(measurement = mean(measurement)) %>%
  arrange(date) %>%
  distinct()
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
# Getting a zoo object
irreg.dates = zoo(irregts.df$measurement,
                  order.by = irregts.df$date)

# Regularizing with aggregate
ag.irregtime = aggregate(irreg.dates,
```

```
                         as.Date, mean)
ag.irregtime
```

```
## 2017-05-16 2017-05-17 2017-05-18 2017-05-19 2017-05-20 2017-05-21
##   334.5000   439.2000   349.2000   345.2000   419.5000   352.9000
## 2017-05-22 2017-05-23 2017-05-24 2017-05-25 2017-05-26 2017-05-27
##   372.2000   402.4500   309.5000   382.0000   432.6000   392.6000
## 2017-05-28 2017-05-29 2017-05-30 2017-05-31
##   391.0000   405.0000   369.9500   338.2333
```

```
length(ag.irregtime)
```

```
## [1] 16
```

```
####
## Method 2 - date and time component kept
sensor.date1 = strptime(irregular_sensor$X1,
                        '%m/%d/%y %I:%M %p')
sensor.date1
```

```
##  [1] "2017-05-16 10:34:00 IST" "2017-05-17 15:23:00 IST"
##  [3] "2017-05-17 20:45:00 IST" "2017-05-18 03:23:00 IST"
##  [5] "2017-05-18 12:34:00 IST" "2017-05-19 11:34:00 IST"
##  [7] "2017-05-20 12:34:00 IST" "2017-05-21 12:34:00 IST"
##  [9] "2017-05-22 17:45:00 IST" "2017-05-22 06:02:00 IST"
## [11] "2017-05-23 04:45:00 IST" "2017-05-23 12:34:00 IST"
## [13] "2017-05-24 02:35:00 IST" "2017-05-25 04:27:00 IST"
## [15] "2017-05-26 15:39:00 IST" "2017-05-27 06:29:00 IST"
## [17] "2017-05-28 07:29:00 IST" "2017-05-29 05:49:00 IST"
## [19] "2017-05-30 07:49:00 IST" "2017-05-30 08:34:00 IST"
## [21] "2017-05-30 13:37:00 IST" "2017-05-30 15:45:00 IST"
## [23] "2017-05-31 05:37:00 IST" "2017-05-31 08:38:00 IST"
## [25] "2017-05-31 16:45:00 IST"
```

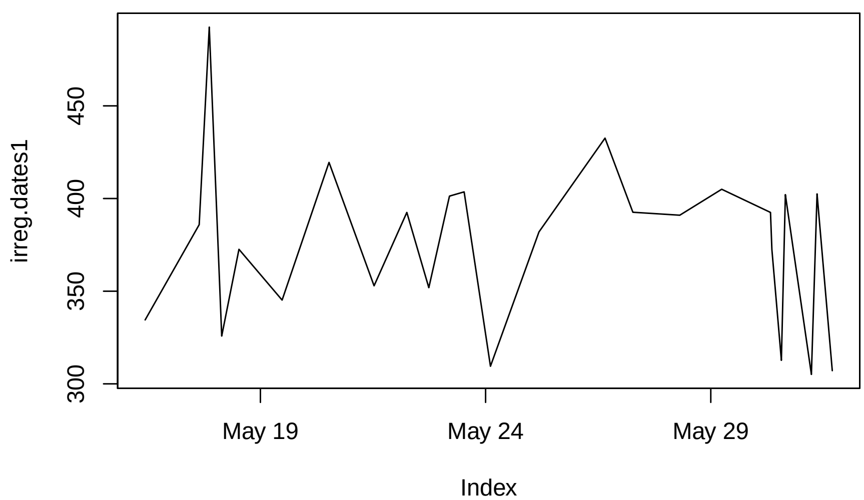
```
irreg.dates1 = data.frame(1:25) %>%
  mutate(date = as.Date(sensor.date1),
         measurement = irregular_sensor$X2) %>%
  select(date, measurement) %>%
  group_by(date) %>%
  mutate(measurement = mean(measurement)) %>%
  arrange(date) %>%
  distinct()
irreg.dates1
```

```
## # A tibble: 16 x 2
## # Groups:   date [16]
##    date       measurement
##    <date>           <dbl>
##  1 2017-05-16        334.
##  2 2017-05-17        439.
##  3 2017-05-18        349.
##  4 2017-05-19        345.
##  5 2017-05-20        420.
##  6 2017-05-21        353.
##  7 2017-05-22        372.
##  8 2017-05-23        402.
##  9 2017-05-24        310.
## 10 2017-05-25        382
## 11 2017-05-26        433.
## 12 2017-05-27        393.
## 13 2017-05-28        391
## 14 2017-05-29        405
## 15 2017-05-30        370.
## 16 2017-05-31        338.
```

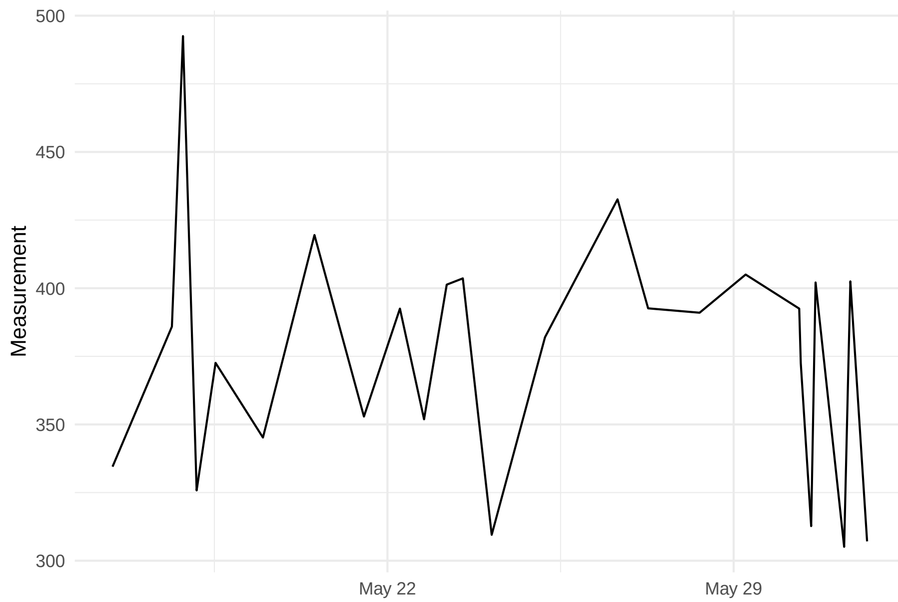
```
# Creating the zoo object
irreg.dates1 = zoo(irregular_sensor$X2,
                   order.by = sensor.date1)
irreg.dates1
```

```
## 2017-05-16 10:34:00 2017-05-17 15:23:00 2017-05-17 20:45:00
##               334.5               385.9               492.5
## 2017-05-18 03:23:00 2017-05-18 12:34:00 2017-05-19 11:34:00
##               325.8               372.6               345.2
## 2017-05-20 12:34:00 2017-05-21 12:34:00 2017-05-22 06:02:00
##               419.5               352.9               392.5
## 2017-05-22 17:45:00 2017-05-23 04:45:00 2017-05-23 12:34:00
##               351.9               401.3               403.6
## 2017-05-24 02:35:00 2017-05-25 04:27:00 2017-05-26 15:39:00
##               309.5               382.0               432.6
## 2017-05-27 06:29:00 2017-05-28 07:29:00 2017-05-29 05:49:00
##               392.6               391.0               405.0
## 2017-05-30 07:49:00 2017-05-30 08:34:00 2017-05-30 13:37:00
##               392.5               372.5               312.7
## 2017-05-30 15:45:00 2017-05-31 05:37:00 2017-05-31 08:38:00
##               402.1               305.1               402.5
## 2017-05-31 16:45:00
##               307.1
```

```
plot(irreg.dates1, type = "l")
```



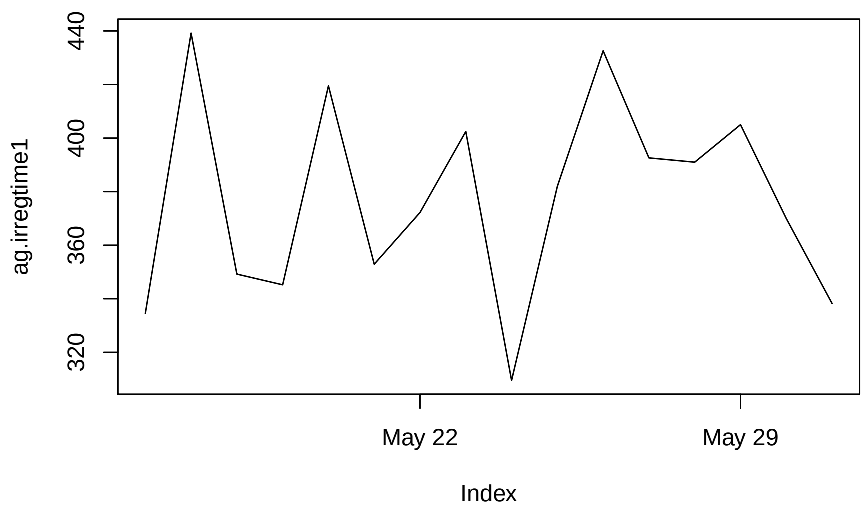

```
autoplot(irreg.dates1) +
  xlab("") +
  ylab("Measurement")
```




```r
# Regularizing with aggregate
ag.irregtime1 = aggregate(irreg.dates1, #data on which to aggregate
                          as.Date, #function on the column by which to aggregate
                          mean) #aggregating function
ag.irregtime1
```

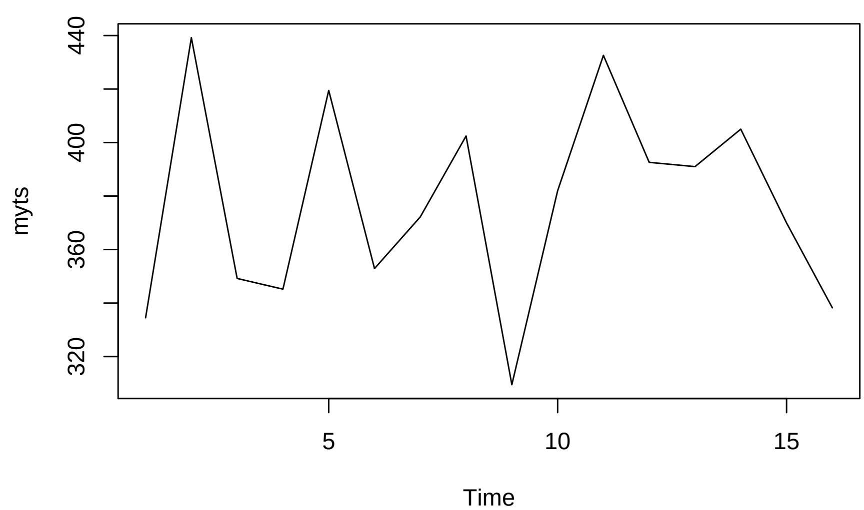
```
## 2017-05-16 2017-05-17 2017-05-18 2017-05-19 2017-05-20 2017-05-21
##   334.5000   439.2000   349.2000   345.2000   419.5000   352.9000
## 2017-05-22 2017-05-23 2017-05-24 2017-05-25 2017-05-26 2017-05-27
##   372.2000   402.4500   309.5000   382.0000   432.6000   392.6000
## 2017-05-28 2017-05-29 2017-05-30 2017-05-31
##   391.0000   405.0000   369.9500   338.2333
```

```r
plot(ag.irregtime1) # plotting the regular zoo object directly
```




```r
autoplot(ag.irregtime1) +
  xlab("") +
  ylab("Measurement")
```

```
myts = ts(ag.irregtime1) # converting to a standard ts, the days start at 1
plot(myts)
```




```
autoplot(ag.irregtime1) +
  xlab("") +
  ylab("Measurement")
```


```
### Working with Missing Data and Outliers
## Import ts.NAandOutliers.csv
mydata <- read_csv("./Data/ts-NAandOutliers.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   mydata = col_double()
## )
```

```
# Convert the 2nd column to a simple ts without frequency
myts = ts(mydata$mydata)
myts
```

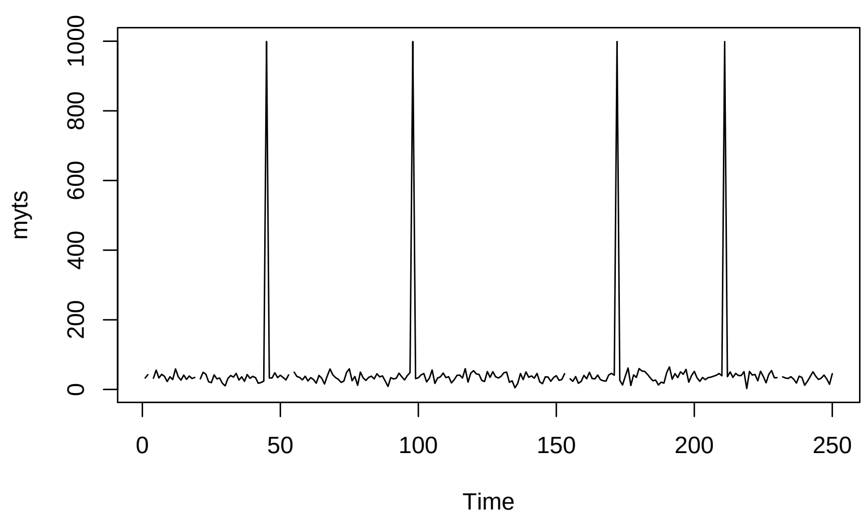
```
## Time Series:
## Start = 1
## End = 250
## Frequency = 1
##   [1]  32.801464  42.465485         NA  32.204058  55.557647  33.050864
##   [7]  43.401620  37.768318  22.844180  36.428877  28.496485  59.037881
##  [13]  36.544163  26.668135  41.325626  28.913199  38.595417  31.341447
```

```
## [19]  34.547023        NA  30.499324  49.391323  43.976004  22.162741
## [25]  19.439525  41.892407  30.321857  32.899878  17.686235  10.332791
## [31]  31.612958  40.011275  35.378517  46.167222  26.903207  36.304821
## [37]  23.408770  42.785841  31.919674  37.571226  33.907485  17.698917
## [43]  19.931775  23.971169 999.000000  32.853670  33.012320  47.893249
## [49]  33.961104  40.826518  34.389579  27.210322  41.815827         NA
## [55]  49.711080  37.246486  34.472507  27.554913  37.976930  24.503481
## [61]  33.941547  28.582326  17.945402  40.335543  32.103075  15.609346
## [67]  38.637130  58.877558  42.178769  34.075469  29.208206  20.409934
## [73]  23.682860  49.014566  59.160903  24.994359  37.321672  11.830421
## [79]  49.907975  33.288427  25.900307  34.661099  38.170951  30.246685
## [85]  45.001326  36.082827  38.969588  24.260726   8.619401  33.933167
## [91]  30.158056  32.211135  46.688584  36.399098  27.266510  39.706101
## [97]  48.560701 999.000000  31.011612  33.565184  41.850476  45.780926
## [103]  21.679404  32.340497  55.904896  17.349895  32.994516  36.155426
## [109]  47.089342  33.955275  36.563838  18.773382  28.077605  40.483324
## [115]  41.341771  32.907839  59.604911  20.989279  46.886734  53.931163
## [121]  44.662468  43.125045  25.800244  22.833920  51.397357  34.775922
## [127]  50.922532  36.430258  32.975690  37.659017  48.006323  49.901919
## [133]  20.619643  24.895206   4.682543  17.049461  45.618543  28.288209
## [139]  50.446258  34.983971  38.847283  32.301493  46.044574  21.739473
## [145]  16.457915  36.157602  35.773314  23.368300  34.220736  39.443674
## [151]  26.074044  28.599269  45.410516         NA  30.585004  23.405284
## [157]  36.949438  17.647508  22.991044  40.388899  30.654440  49.261182
## [163]  31.215505  30.462442  41.294423  28.046393  24.925970  23.934094
## [169]  41.690112  46.226476  40.741721 999.000000  26.455048  12.766929
## [175]  38.133315  61.653241  11.474054  41.835335  34.572898  59.921615
## [181]  53.072688  51.889866  43.700888  33.639492  24.988901  27.019376
## [187]  12.774882  21.001551  18.133113  48.563807  64.633075  29.362668
## [193]  45.478245  34.152629  50.573869  43.564419  57.644084  20.785408
## [199]  39.811707  51.854335  33.351763  23.242107  34.351879  28.113792
## [205]  33.952911  35.372668  38.059841  40.818440  45.768335  39.272128
## [211] 999.000000  36.665537  50.282893  34.561040  46.040830  39.936308
## [217]  39.873144  51.126396   2.683472  51.667975  41.336229  43.090450
## [223]  24.686842  52.300908  37.379943  19.043254  43.512121  54.236360
## [229]  33.557160  33.851597         NA  36.149460  32.985037  31.422766
## [235]  36.574851  29.648483  18.290954  38.154075  34.446452  12.037743
## [241]  23.581530  36.968395  50.747174  37.981389  28.693203  32.396009
## [247]  41.325484  30.017571  14.818111  45.403854
```

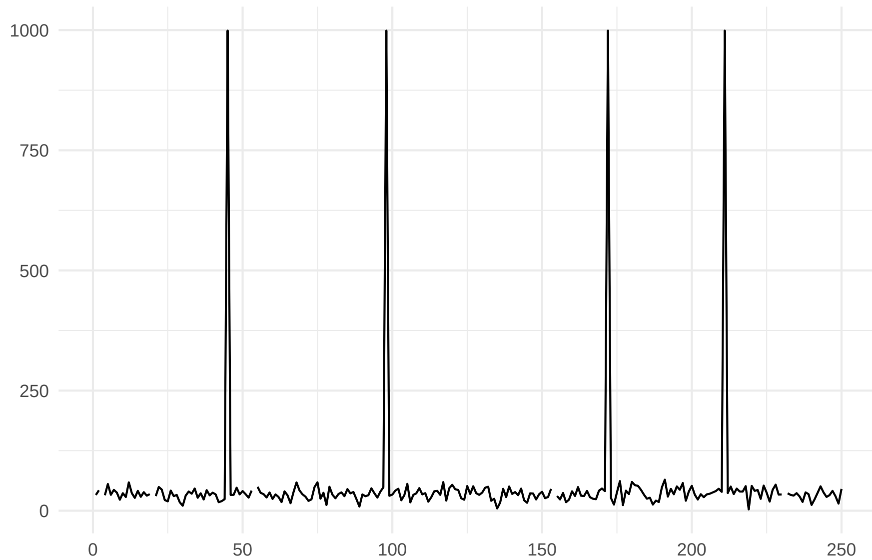
```r
# Checking for NAs and outliers
summary(myts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   2.683  28.078  34.573  50.710  42.465 999.000       5
```

```
plot(myts)
```



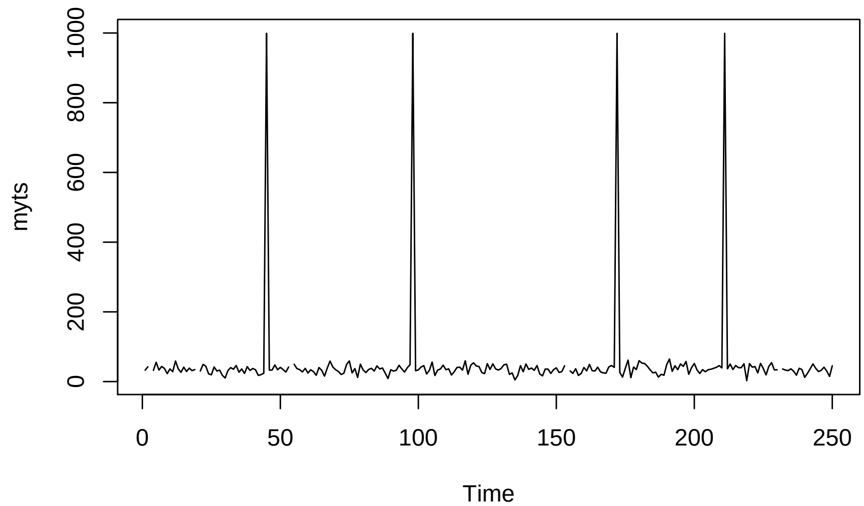

```
autoplot(myts) +
  xlab("") +
  ylab("")
```



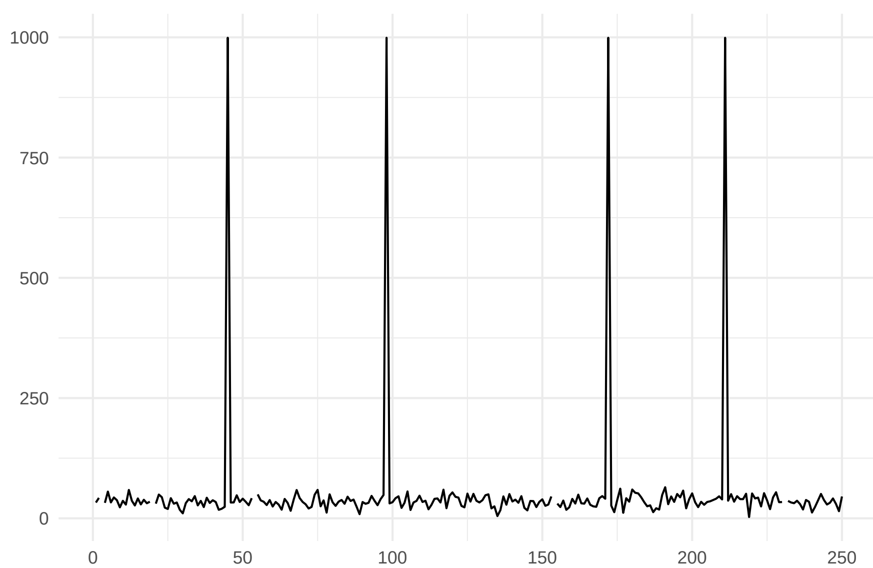

```r
# Automatic detection of outliers
myts1 = tsoutliers(myts)
myts1
```

```
## $index
## [1]  45  98 172 211
##
## $replacements
## [1] 28.41242 39.78616 33.59838 37.96883
```

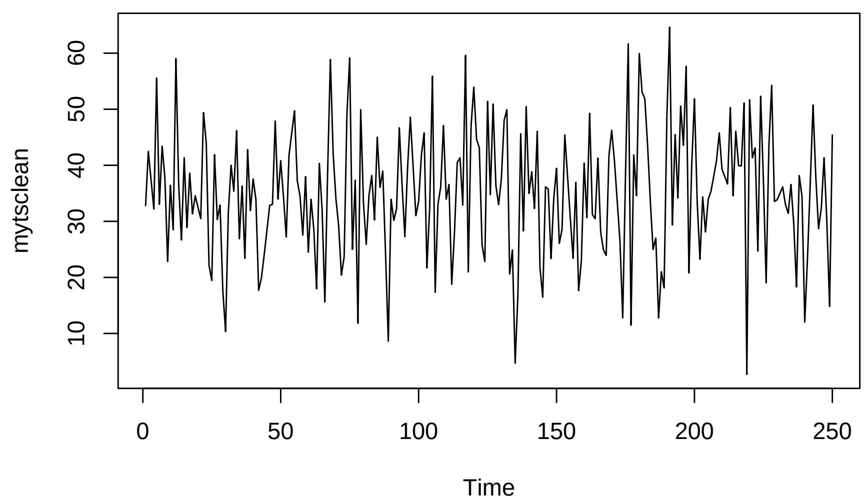
```r
plot(myts)
```



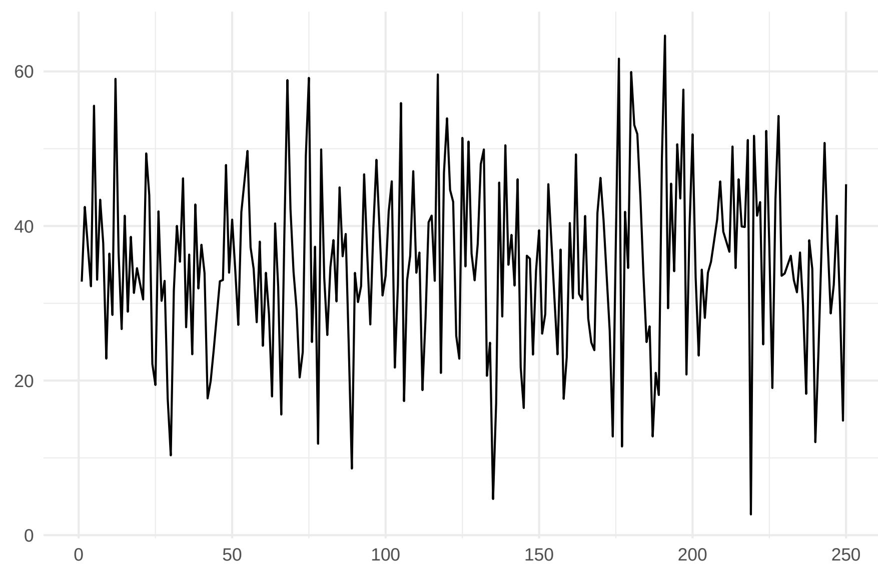

```
autoplot(myts) +
  xlab("") +
  ylab("")
```



```
# Missing data handling with zoo
myts.NAlocf = na.locf(myts)
myts.NAfill = na.fill(myts, 33)

# Tip: na.trim to get rid of NAs at the beginning or end of dataset
# Standard NA method in package forecast
myts.NAinterp = na.interp(myts)

# Cleaning NA and outliers with forecast package
mytsclean = tsclean(myts)
plot(mytsclean)
```




```
autoplot(mytsclean) +
  xlab("") +
  ylab("")
```




```
summary(mytsclean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.683  28.157  34.567  35.025  41.830  64.633
```

Chapter 4

Statistical Background For TS Analysis & Forecasting

```
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section4")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
require(lmtest)
```

```
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section4")
#loading the data
lynx
```

```
## Time Series:
## Start = 1821
## End = 1934
## Frequency = 1
```

```
##   [1]  269  321  585  871 1475 2821 3928 5943 4950 2577  523   98  184  279
##  [15]  409 2285 2685 3409 1824  409  151   45   68  213  546 1033 2129 2536
##  [29]  957  361  377  225  360  731 1638 2725 2871 2119  684  299  236  245
##  [43]  552 1623 3311 6721 4254  687  255  473  358  784 1594 1676 2251 1426
##  [57]  756  299  201  229  469  736 2042 2811 4431 2511  389   73   39   49
##  [71]   59  188  377 1292 4031 3495  587  105  153  387  758 1307 3465 6991
##  [85] 6313 3794 1836  345  382  808 1388 2713 3800 3091 2985 3790  674   81
##  [99]   80  108  229  399 1132 2432 3574 2935 1537  529  485  662 1000 1590
## [113] 2657 3396
```

```r
#looking at the data
time(lynx) #to look at the time stamps of the time series data
```

```
## Time Series:
## Start = 1821
## End = 1934
## Frequency = 1
##   [1] 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834
##  [15] 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848
##  [29] 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862
##  [43] 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876
##  [57] 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890
##  [71] 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904
##  [85] 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918
##  [99] 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932
## [113] 1933 1934
```

```r
length(lynx) #number of observations in the time series data
```

```
## [1] 114
```

```r
tail(lynx) # last 6 observations
```

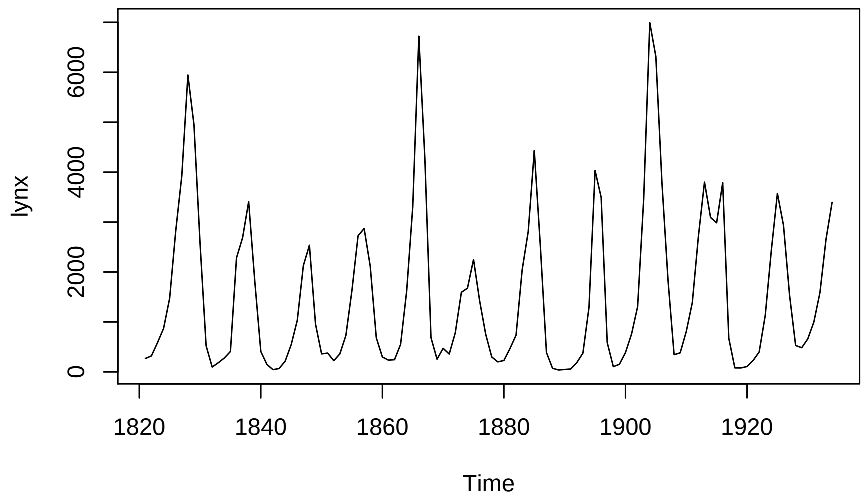
```
## Time Series:
## Start = 1929
## End = 1934
## Frequency = 1
## [1]  485  662 1000 1590 2657 3396
```

```r
mean(lynx); median(lynx) #simple descriptive statistics of the data
```

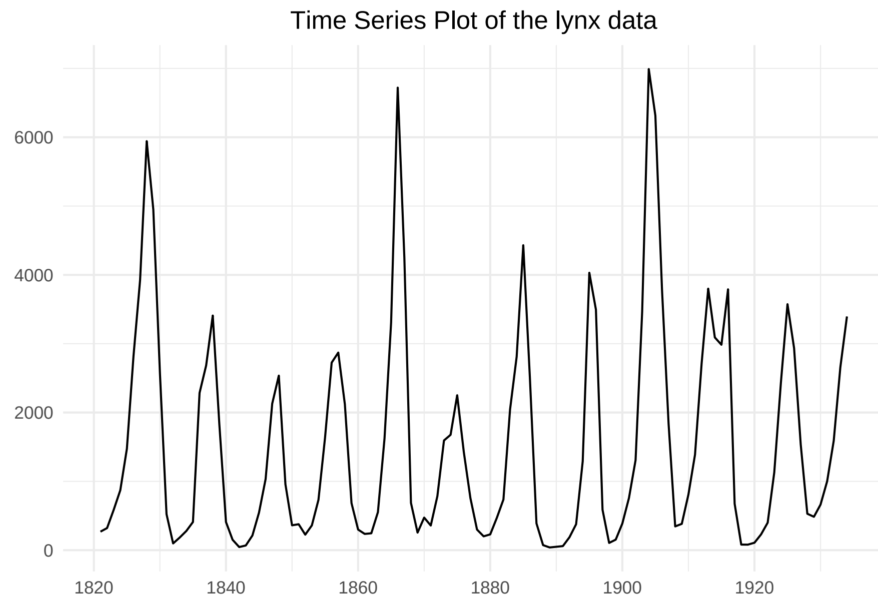
```
## [1] 1538.018
```

```
## [1] 771
```

```
plot(lynx) #see how time series is behaving
```




```
theme_set(theme_minimal())
autoplot(lynx) +
  xlab("") + ylab("") +
  ggtitle("Time Series Plot of the lynx data") +
  theme(plot.title = element_text(hjust = 0.5))  #for centering the text
```




```r
sort(lynx)
```

```
##   [1]   39   45   49   59   68   73   80   81   98  105  108  151  153  184
##  [15]  188  201  213  225  229  229  236  245  255  269  279  299  299  321
##  [29]  345  358  360  361  377  377  382  387  389  399  409  409  469  473
##  [43]  485  523  529  546  552  585  587  662  674  684  687  731  736  756
##  [57]  758  784  808  871  957 1000 1033 1132 1292 1307 1388 1426 1475 1537
##  [71] 1590 1594 1623 1638 1676 1824 1836 2042 2119 2129 2251 2285 2432 2511
##  [85] 2536 2577 2657 2685 2713 2725 2811 2821 2871 2935 2985 3091 3311 3396
##  [99] 3409 3465 3495 3574 3790 3794 3800 3928 4031 4254 4431 4950 5943 6313
## [113] 6721 6991
```

```r
sort(lynx)[c(57,58)]
```

```
## [1] 758 784
```

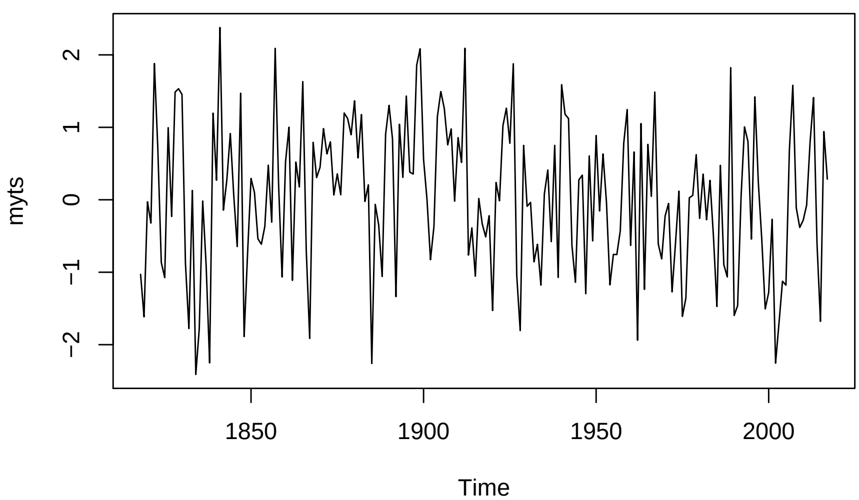
```r
quantile(lynx) #to give the quantiles of the time series
```

```
##      0%     25%     50%     75%    100%
##   39.00  348.25  771.00 2566.75 6991.00
```

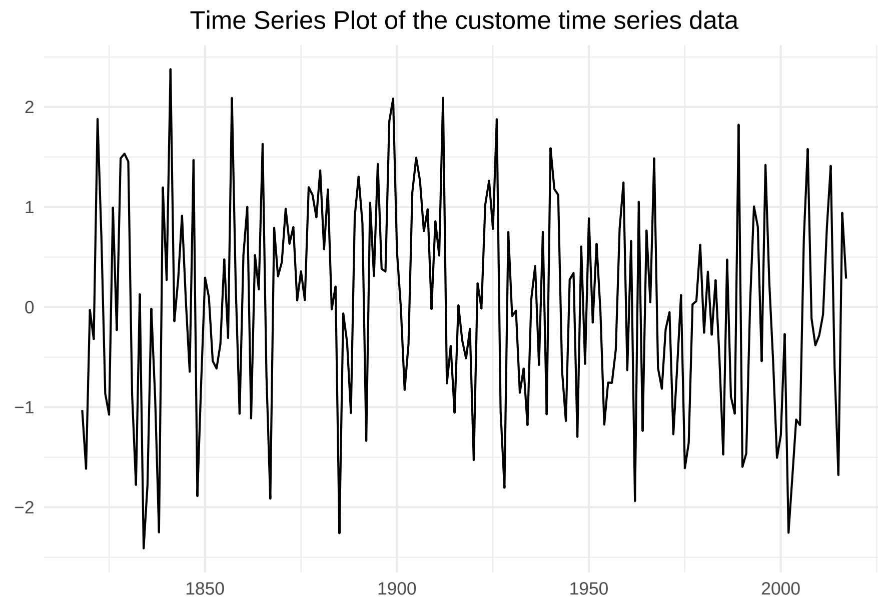
```
quantile(lynx,
         prob = seq(0, 1, length = 11), #number of categories
         type = 5) #deciles
```

```
##     0%    10%    20%    30%    40%    50%    60%    70%    80%    90%
##   39.0  146.7  259.2  380.5  546.6  771.0 1470.1 2165.6 2818.0 3790.4
##   100%
## 6991.0
```

```
### simple forecast methods

set.seed(95)
myts <- ts(rnorm(200), start = (1818))
plot(myts)
```




```
autoplot(myts) +
  xlab("") + ylab("") +
  ggtitle("Time Series Plot of the custome time series data") +
  theme(plot.title = element_text(hjust = 0.5))  #for centering the text
```




```r
meanm <- meanf(myts, h=20) #forecast by taking the mean of the values
naivem <- naive(myts, h=20) #forecast by taking the last observation forward
driftm <- rwf(myts, h=20, drift = T) #forecast by drift model

plot(meanm, plot.conf = F, main = "")
```

```
## Warning in plot.window(xlim, ylim, log, ...): "plot.conf" is not a
## graphical parameter
```

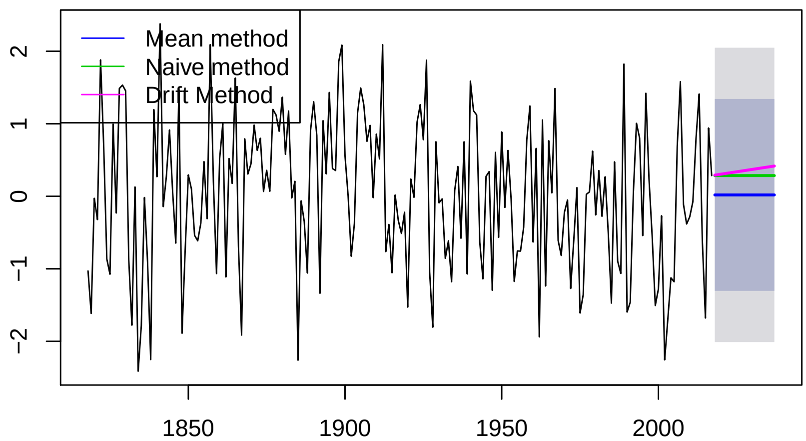
```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "plot.conf"
## is not a graphical parameter
```

```
## Warning in axis(1, ...): "plot.conf" is not a graphical parameter
```

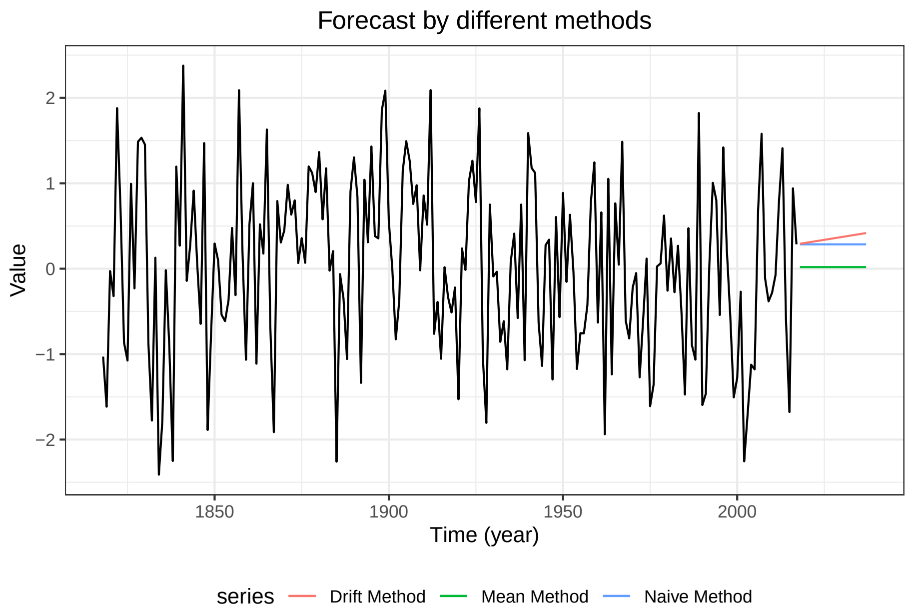
```
## Warning in axis(2, ...): "plot.conf" is not a graphical parameter
```

```
## Warning in box(...): "plot.conf" is not a graphical parameter
```

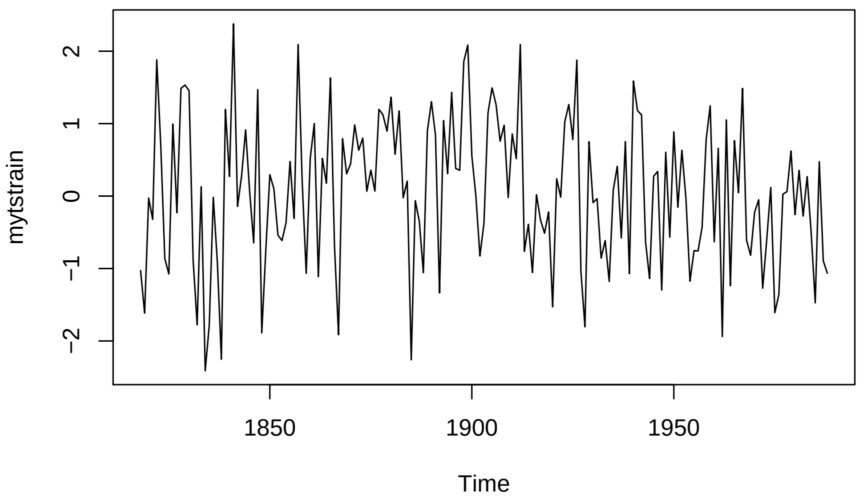
```r
lines(naivem$mean, col=123, lwd = 2)
lines(driftm$mean, col=22, lwd = 2)
legend("topleft",lty=1,col=c(4,123,22),
       legend=c("Mean method","Naive method","Drift Method"))
```



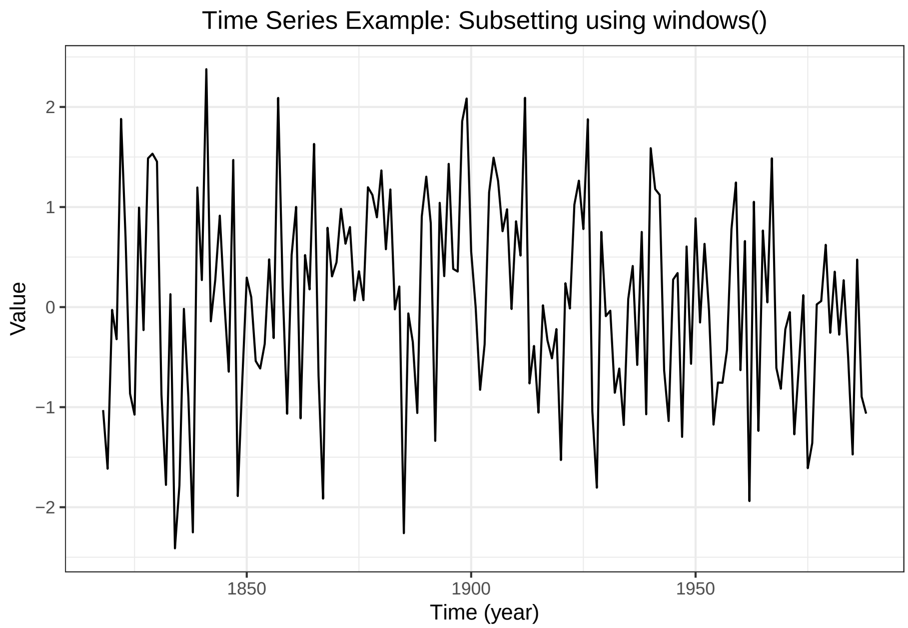

```r
autoplot(myts) +
  autolayer(meanm$mean, series = "Mean Method") +
  autolayer(naivem$mean, series = "Naive Method") +
  autolayer(driftm$mean, series = "Drift Method") +
  ggtitle("Forecast by different methods") +
  xlab("Time (year)") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
###### accuracy and model comparison
#subset the time series. Divide into training and the testing set. Forecast on the tra
set.seed(95)
myts <- ts(rnorm(200), start = (1818))
mytstrain <- window(myts, start = 1818, end = 1988) #subset the time series using wind
plot(mytstrain)
```




```
autoplot(mytstrain) +
  ggtitle("Time Series Example: Subsetting using windows()") +
  xlab("Time (year)") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
meanm <- meanf(mytstrain, h=30)
naivem <- naive(mytstrain, h=30)
driftm <- rwf(mytstrain, h=30, drift = T)

mytstest <- window(myts, start = 1988)

accuracy(meanm, mytstest) #accuracy function allows one to compare the performance of
```

```
##                         ME     RMSE       MAE       MPE     MAPE      MASE
## Training set  1.407307e-17 1.003956 0.8164571  77.65393 133.4892 0.7702074
## Test set     -2.459828e-01 1.138760 0.9627571 100.70356 102.7884 0.9082199
##                   ACF1 Theil's U
## Training set 0.1293488        NA
## Test set     0.2415939  0.981051
```

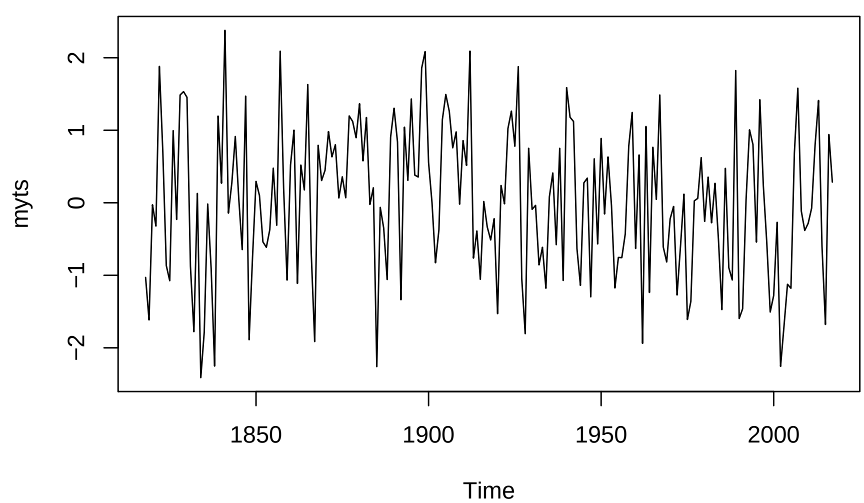
```
accuracy(naivem, mytstest)
```

```
##                         ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -0.0002083116 1.323311 1.060048 -152.73569 730.9655 1.000000
## Test set      0.8731935861 1.413766 1.162537   86.29346 307.9891 1.096683
##                    ACF1 Theil's U
## Training set -0.4953144        NA
## Test set      0.2415939  2.031079
```

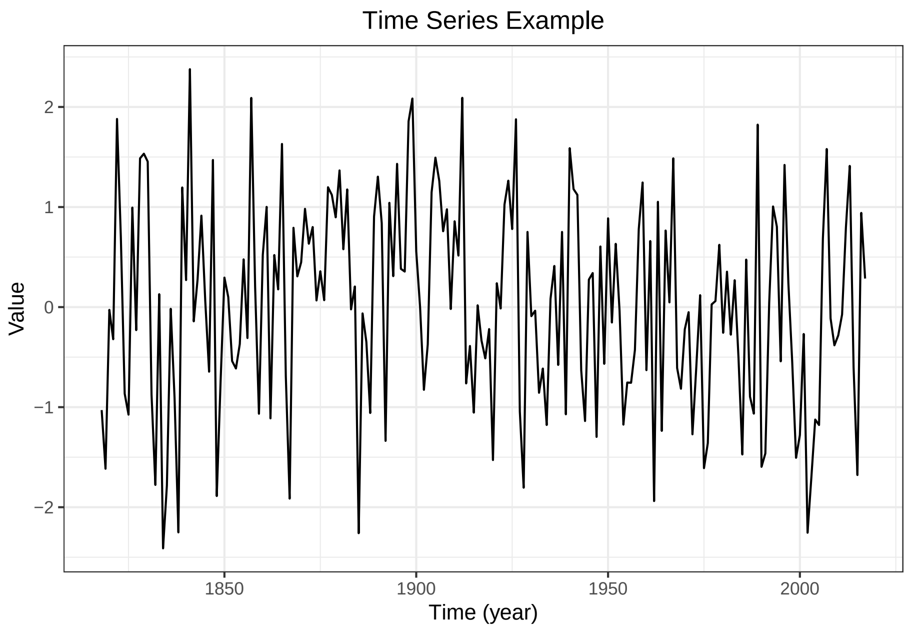
```r
accuracy(driftm, mytstest)
```

```
##                         ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -1.957854e-17 1.323311 1.060041 -152.64988 730.8626 0.9999931
## Test set      8.763183e-01 1.415768 1.163981   85.96496 308.7329 1.0980447
##                    ACF1 Theil's U
## Training set -0.4953144        NA
## Test set      0.2418493   2.03317
```

```r
###### Residuals

set.seed(95)
myts <- ts(rnorm(200), start = (1818))
plot(myts)
```




```r
autoplot(myts) +
  ggtitle("Time Series Example") +
  xlab("Time (year)") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
meanm <- meanf(myts, h=20)
naivem <- naive(myts, h=20)
driftm <- rwf(myts, h=20, drift = T)

var(meanm$residuals)
```

```
## [1] 1.053807
```

```
mean(meanm$residuals)
```

```
## [1] -5.95498e-18
```

```
mean(naivem$residuals)
```

```
## [1] NA
```

```
naivwithoutNA <- naivem$residuals
naivwithoutNA <- naivwithoutNA[2:200] #naive and drift models need one observation to
var(naivwithoutNA)
```

```
## [1] 1.798592
```

```
mean(naivwithoutNA)
```

```
## [1] 0.006605028
```

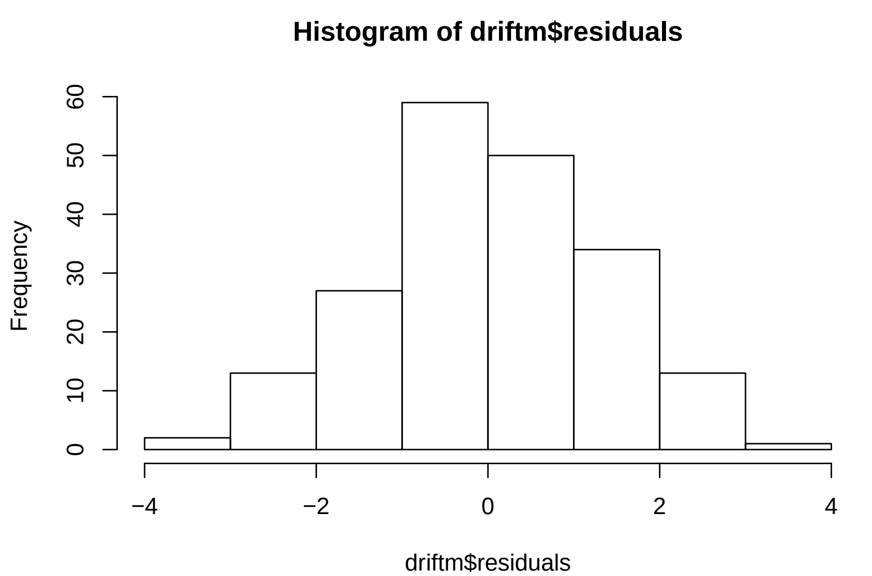
```
driftwithoutNA <- driftm$residuals
driftwithoutNA <- driftwithoutNA[2:200]
var(driftwithoutNA)
```

```
## [1] 1.798592
```

```
mean(driftwithoutNA)
```

```
## [1] -4.502054e-17
```

```
hist(driftm$residuals)
```

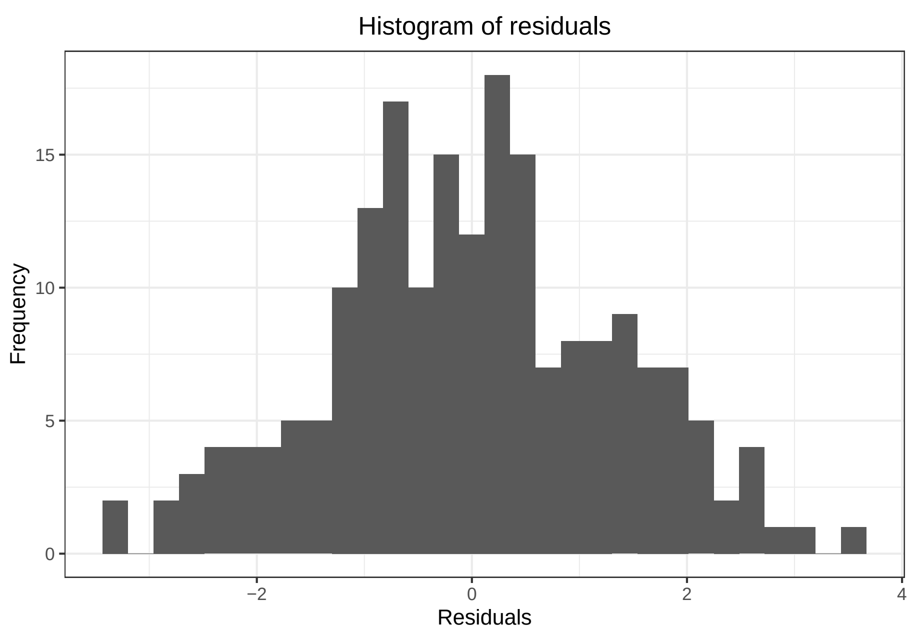



```
ggplot() +
  geom_histogram(aes(x = driftm$residuals)) +
  ggtitle("Histogram of residuals") +
  xlab("Residuals") + ylab("Frequency") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

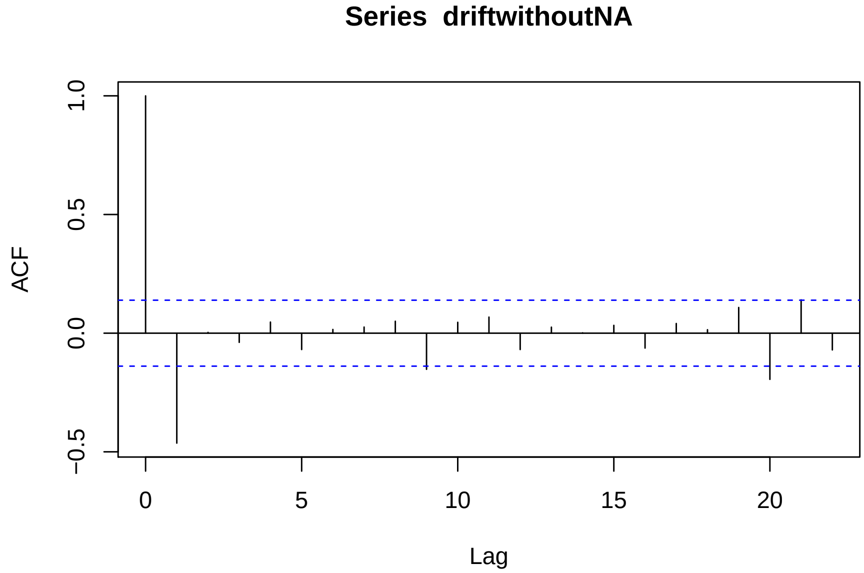
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to con
```

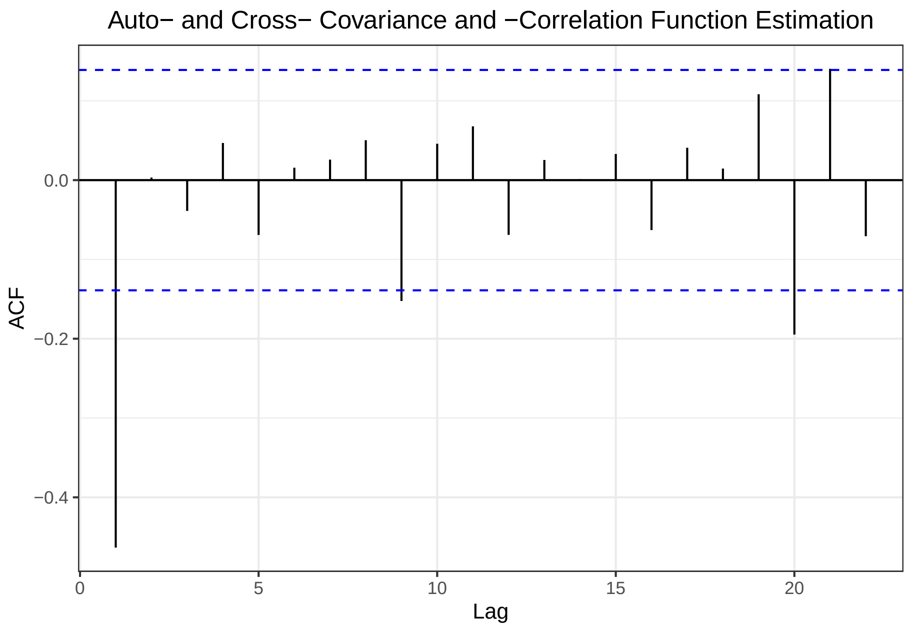
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```




```
acf(driftwithoutNA) #acf is used to identify the moving average part of the ARIMA mode
autoplot(acf(driftwithoutNA)) +
  ggtitle("Auto- and Cross- Covariance and -Correlation Function Estimation") +
  xlab("Lag") + ylab("ACF") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```



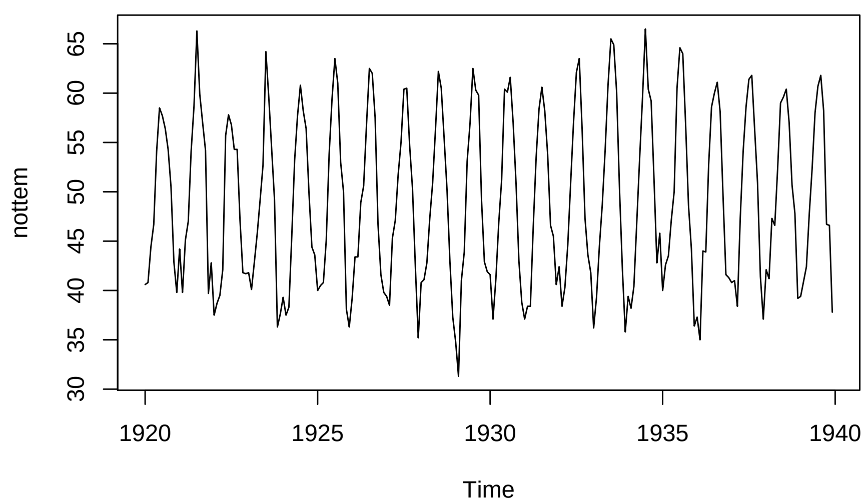



```
### Stationarity
set.seed(2019)
x <- rnorm(1000) # no unit-root, stationary

adf.test(x) # augmented Dickey Fuller Test Augmented Dickey-Fuller test removes autocorrelation a
```

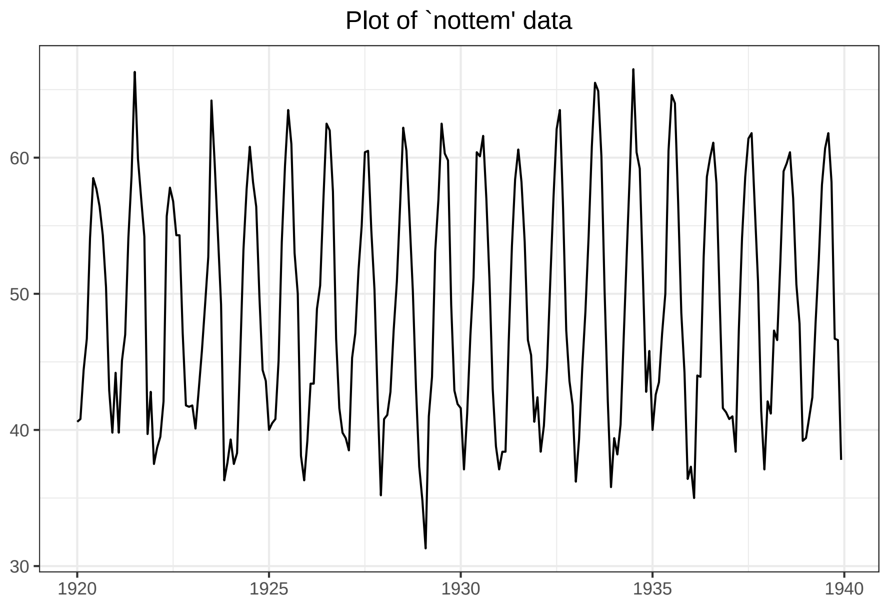
```
## Warning in adf.test(x): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  x
## Dickey-Fuller = -10.647, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

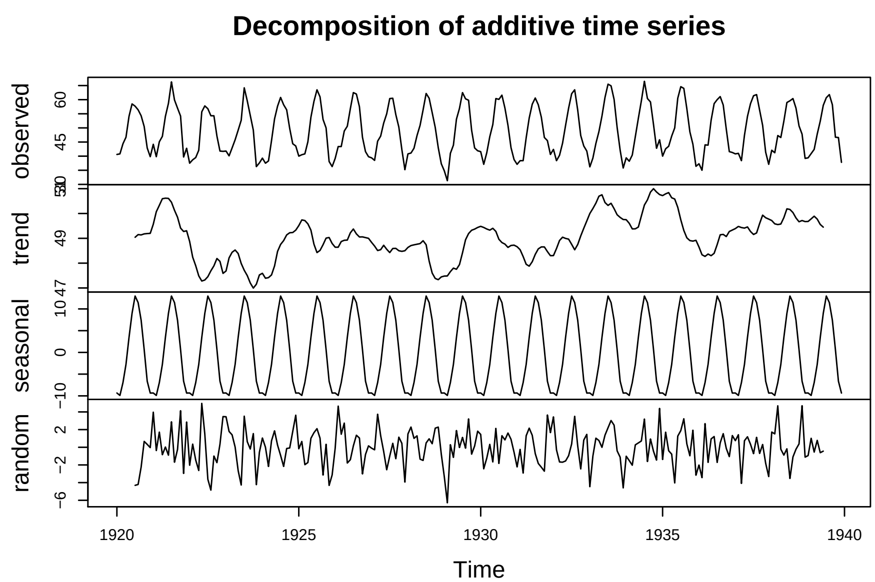
```
plot(nottem) # Let s see the nottem dataset
```



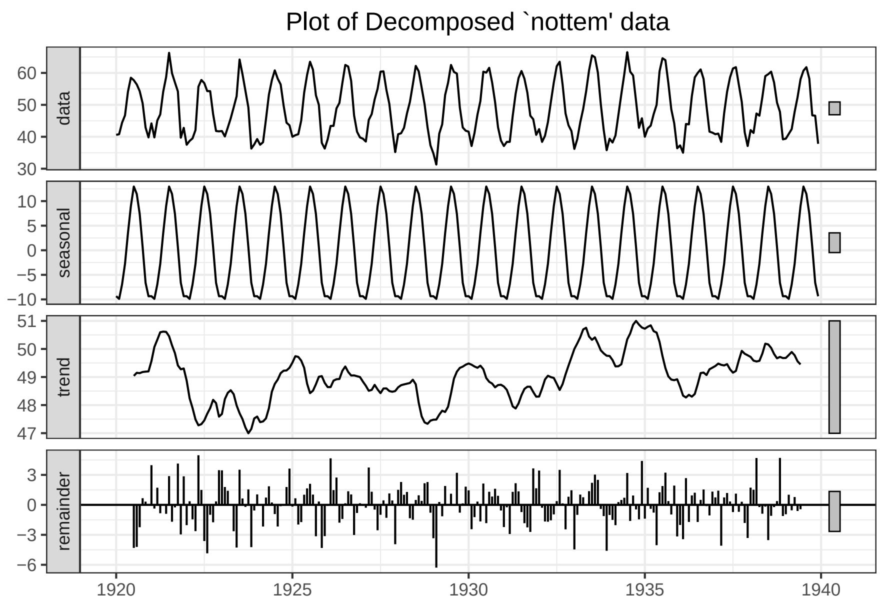

```
autoplot(nottem) +
  ggtitle("Plot of `nottem' data") +
  xlab("") + ylab("") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
plot(decompose(nottem))
```



```
autoplot(decompose(nottem)) +
  ggtitle("Plot of Decomposed `nottem' data") +
  xlab("") + ylab("") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

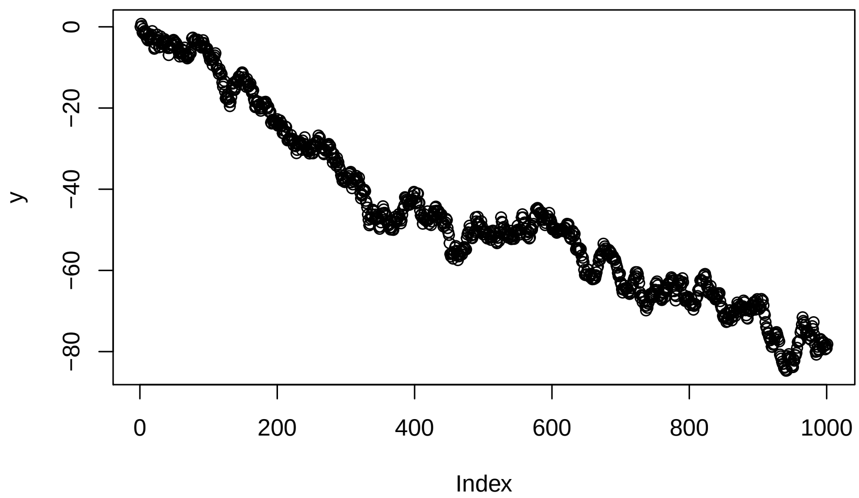



```
adf.test(nottem)
```

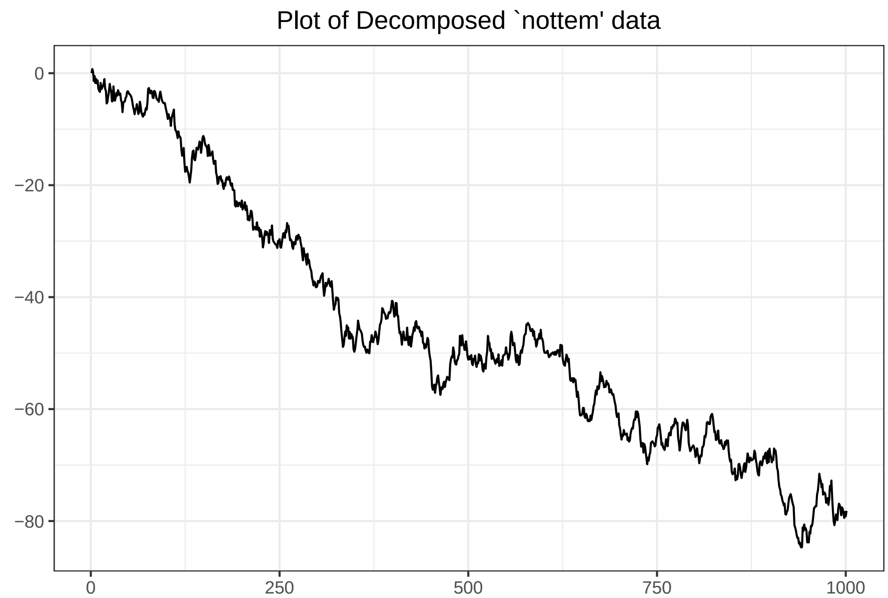
```
## Warning in adf.test(nottem): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  nottem
## Dickey-Fuller = -12.998, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
y <- diffinv(x) # non-stationary

plot(y)
```




```
ggplot() +
  geom_line(aes(y = y, x= index(y))) +
  ggtitle("Plot of Decomposed `nottem' data") +
  xlab("") + ylab("") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
adf.test(y)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y
## Dickey-Fuller = -2.6432, Lag order = 9, p-value = 0.3061
## alternative hypothesis: stationary
```

```
### Autocorrelation

# Durbin Watson test for autocorrelation
length(lynx); head(lynx); head(lynx[-1]); head(lynx[-114]) # check the required traits
```

```
## [1] 114
```

```
## Time Series:
## Start = 1821
## End = 1826
## Frequency = 1
## [1]  269  321  585  871 1475 2821
```

```
## [1]  321  585  871 1475 2821 3928
```

```
## [1]  269  321  585  871 1475 2821
```

```r
dwtest(lynx[-114] ~ lynx[-1])
```

```
##
##  Durbin-Watson test
##
## data:  lynx[-114] ~ lynx[-1]
## DW = 1.1296, p-value = 1.148e-06
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
set.seed(2019)
x = rnorm(700) # Lets take a look at random numbers
dwtest(x[-700] ~ x[-1])
```

```
##
##  Durbin-Watson test
##
## data:  x[-700] ~ x[-1]
## DW = 1.996, p-value = 0.4789
## alternative hypothesis: true autocorrelation is greater than 0
```

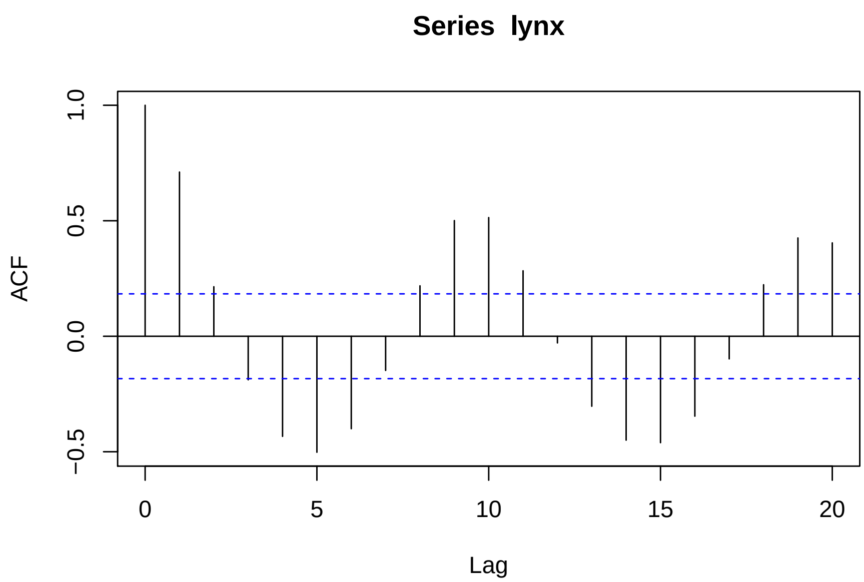
```r
length(nottem) # and the nottem dataset
```

```
## [1] 240
```

```r
dwtest(nottem[-240] ~ nottem[-1])
```

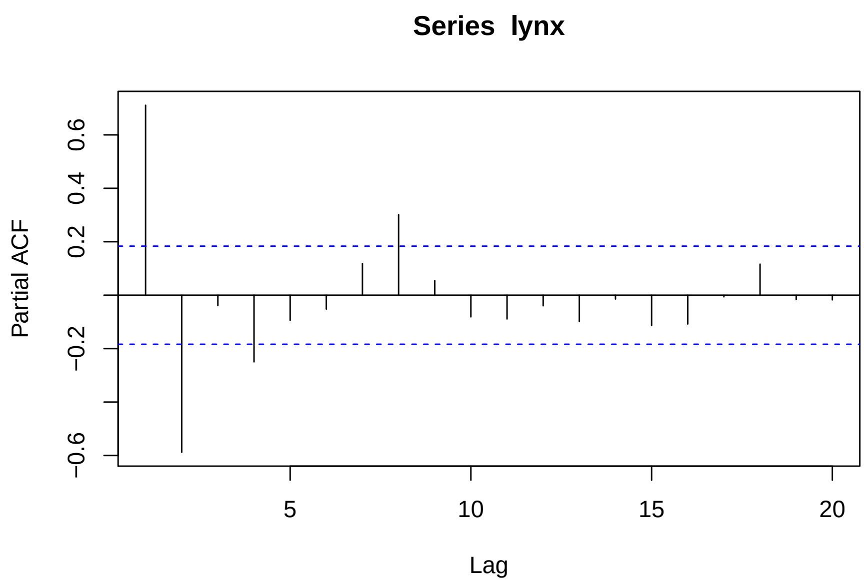
```
##
##  Durbin-Watson test
##
## data:  nottem[-240] ~ nottem[-1]
## DW = 1.0093, p-value = 5.097e-15
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
### ACF and PACF
acf(lynx, lag.max = 20); pacf(lynx, lag.max =20, plot = FALSE)
```

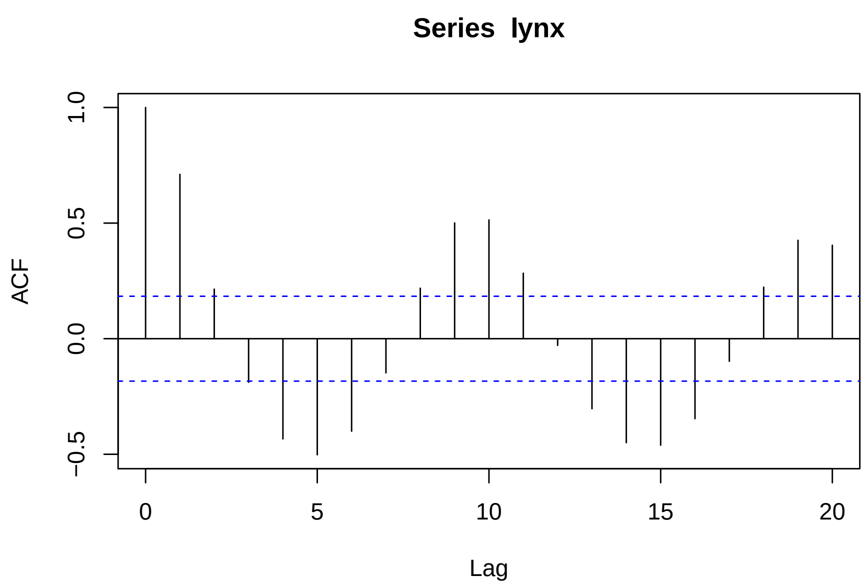


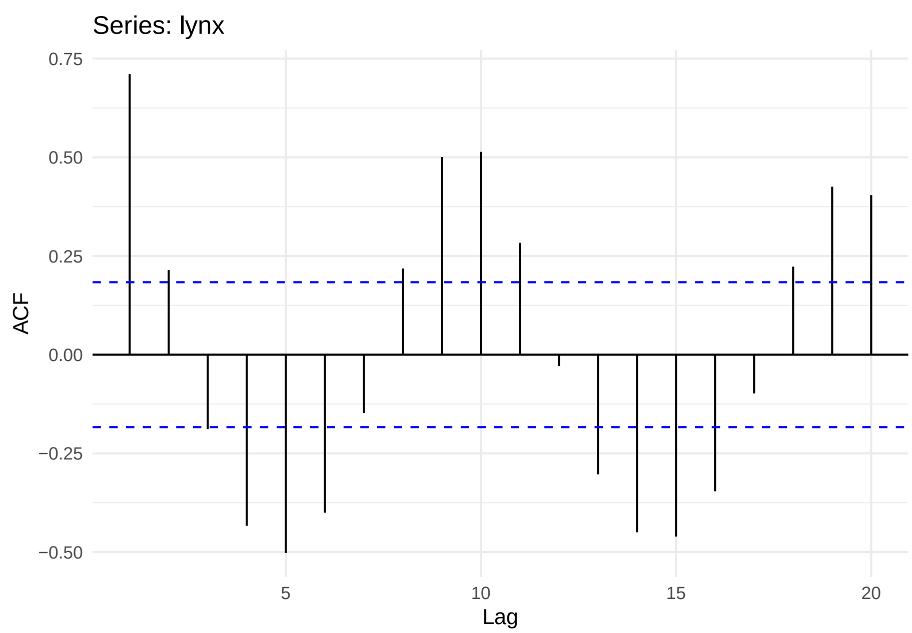

```
##
## Partial autocorrelations of series 'lynx', by lag
##
##      1      2      3      4      5      6      7      8      9     10
##  0.711 -0.588 -0.039 -0.250 -0.094 -0.052  0.119  0.301  0.055 -0.081
##     11     12     13     14     15     16     17     18     19     20
## -0.089 -0.040 -0.099 -0.014 -0.113 -0.108 -0.006  0.116 -0.016 -0.018
```

```r
pacf(lynx, lag.max =20, plot = TRUE)
```

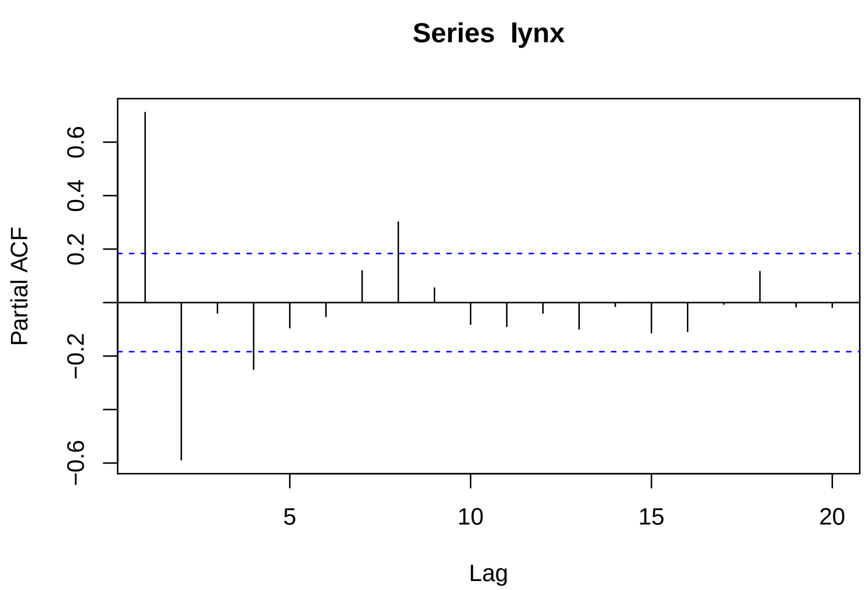


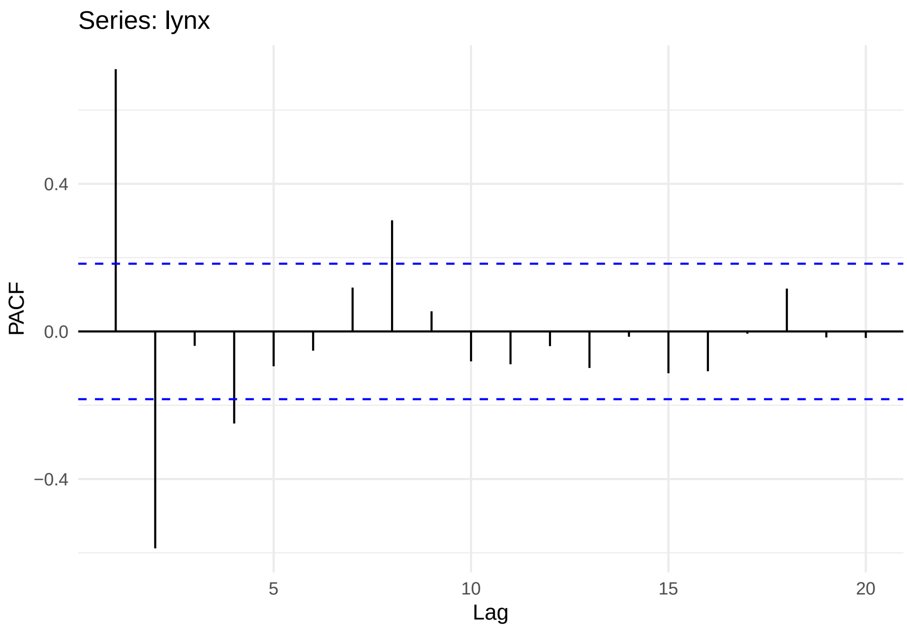

```r
autoplot(acf(lynx, lag.max = 20))
```



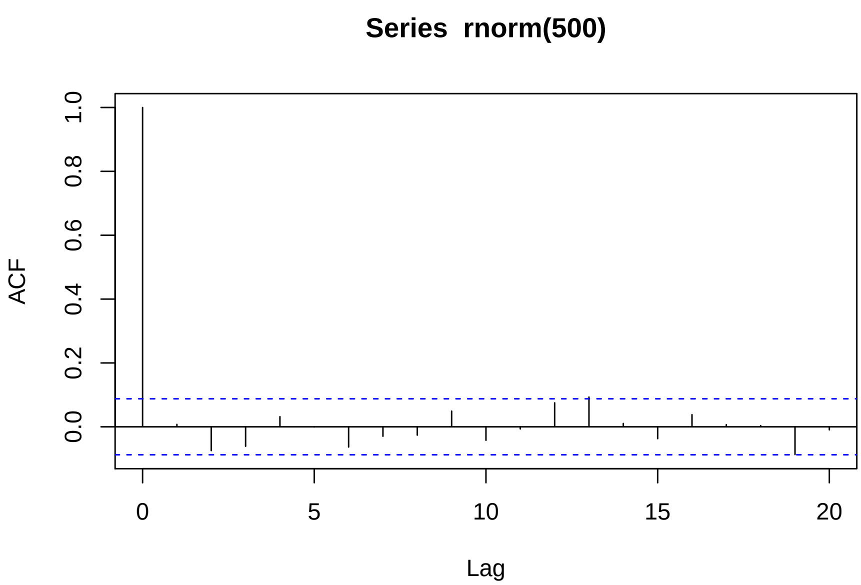



```
autoplot(pacf(lynx, lag.max =20)) #in acf() first correlation is with itself, so we ca
```

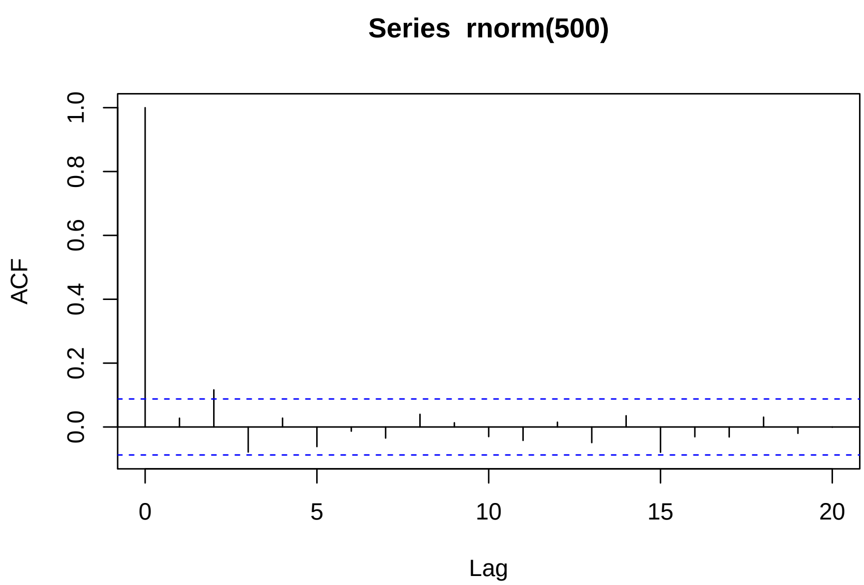


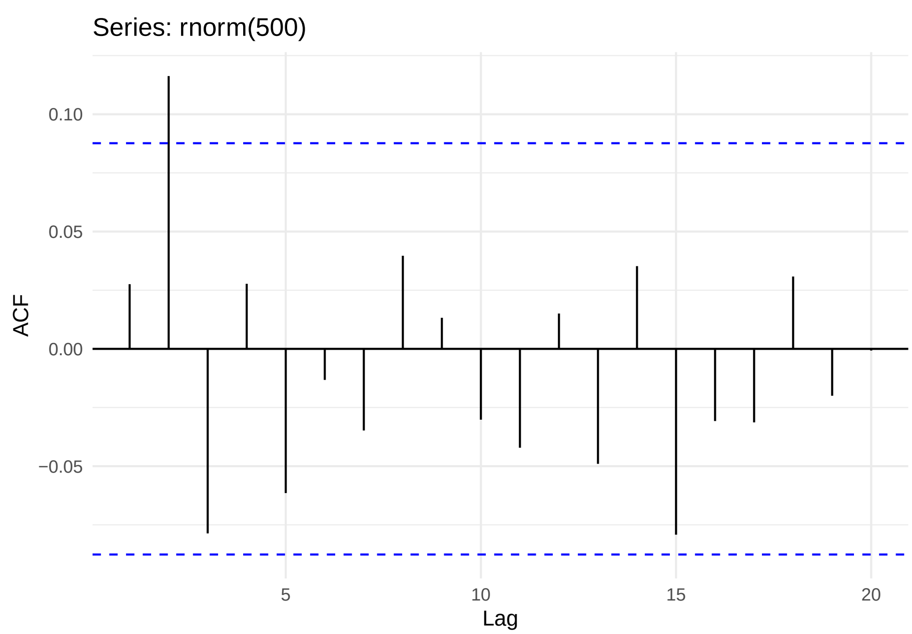



```
# lag.max for numbers of lags to be calculated
# plot = F to suppress plotting
set.seed(2019)
acf(rnorm(500), lag.max = 20)
```

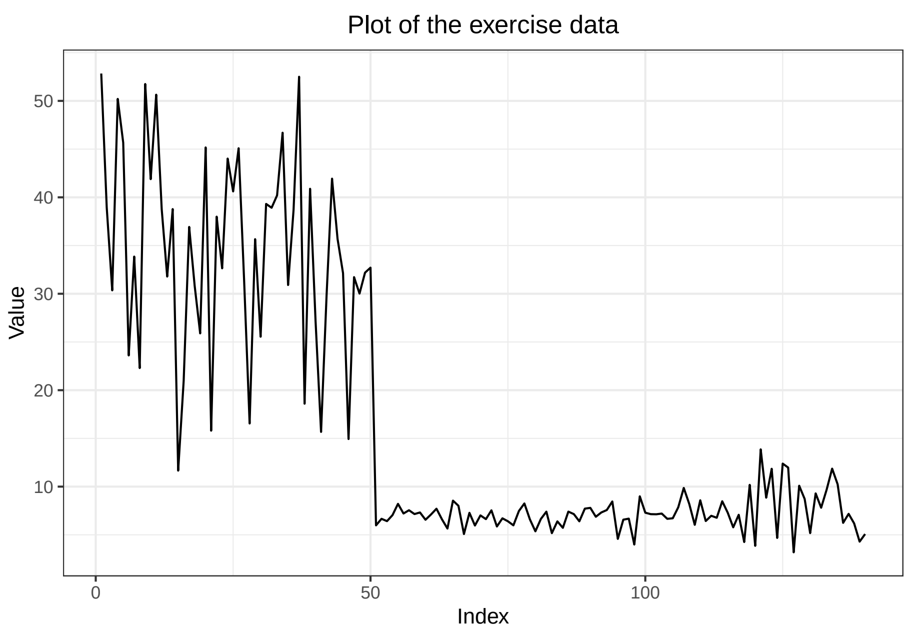


```
autoplot(acf(rnorm(500), lag.max = 20))
```





```r
## Exercise messy data
set.seed(54)
myts <- ts(c(rnorm(50, 34, 10),
             rnorm(67, 7, 1),
             runif(23, 3, 14)))

#1. Plot the data and examine it.
autoplot(myts) +
  ggtitle("Plot of the exercise data") +
  xlab("Index") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```r
#2. three forecasting models
meanm <- meanf(myts, h = 10)
naivem <- naive(myts, h= 10)
driftm <- rwf(myts, h =10, drift = TRUE)

#3. plot forecasts from three forecasting models
forecast::autoplot(myts,
         PI = TRUE,
         flwd = 2) +
  autolayer(meanm$mean, series = "Mean Method", PI = TRUE) +
```

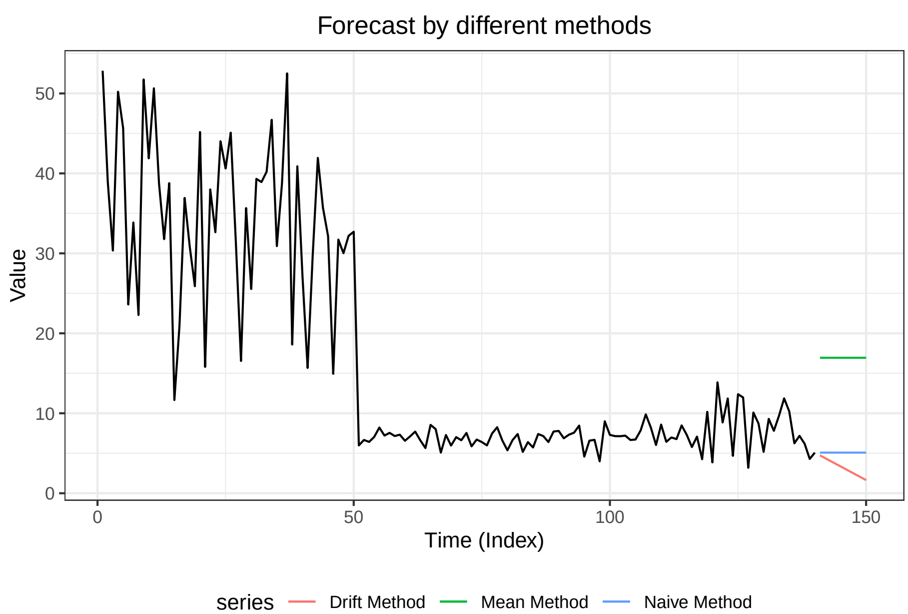
```
  autolayer(naivem$mean, series = "Naive Method", PI = TRUE) +
  autolayer(driftm$mean, series = "Drift Method", PI = TRUE) +
  ggtitle("Forecast by different methods") +
  xlab("Time (Index)") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

```
## Warning: Ignoring unknown parameters: PI, flwd
```

```
## Warning: Ignoring unknown parameters: PI
```

```
## Warning: Ignoring unknown parameters: PI
```

```
## Warning: Ignoring unknown parameters: PI
```




```
#4. Which model looks most promising

#5. Get the error measures and compare them
var(meanm$residuals)
```

```
## [1] 211.0364
```

```
mean(meanm$residuals)
```

```
## [1] -1.591522e-15
```

```
mean(naivem$residuals)
```

```
## [1] NA
```

```
naivwithoutNA <- naivem$residuals
naivwithoutNA <- naivwithoutNA[2:140] #naive and drift models need one observation to start with.
var(naivwithoutNA)
```

```
## [1] 84.53723
```

```
mean(naivwithoutNA)
```

```
## [1] -0.3435748
```

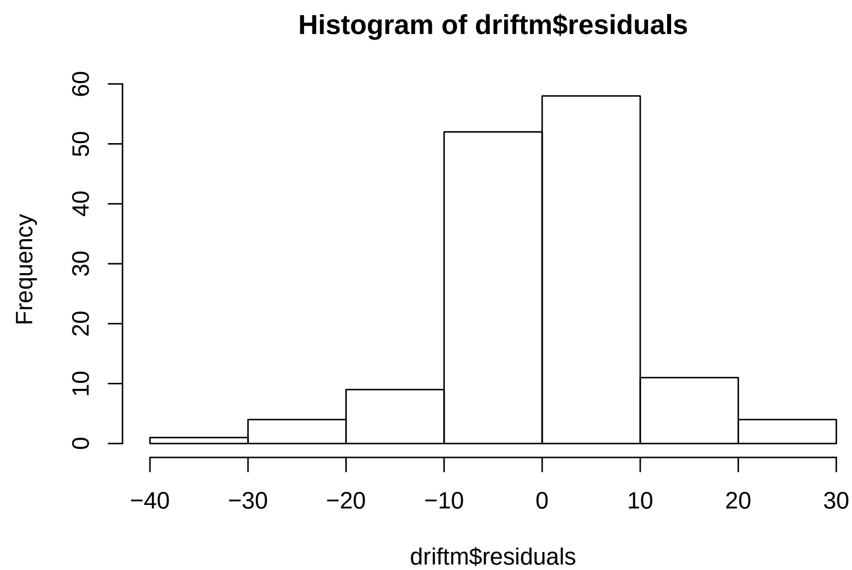
```
driftwithoutNA <- driftm$residuals
driftwithoutNA <- driftwithoutNA[2:140]
var(driftwithoutNA)
```

```
## [1] 84.53723
```

```
mean(driftwithoutNA)
```

```
## [1] 2.648343e-15
```

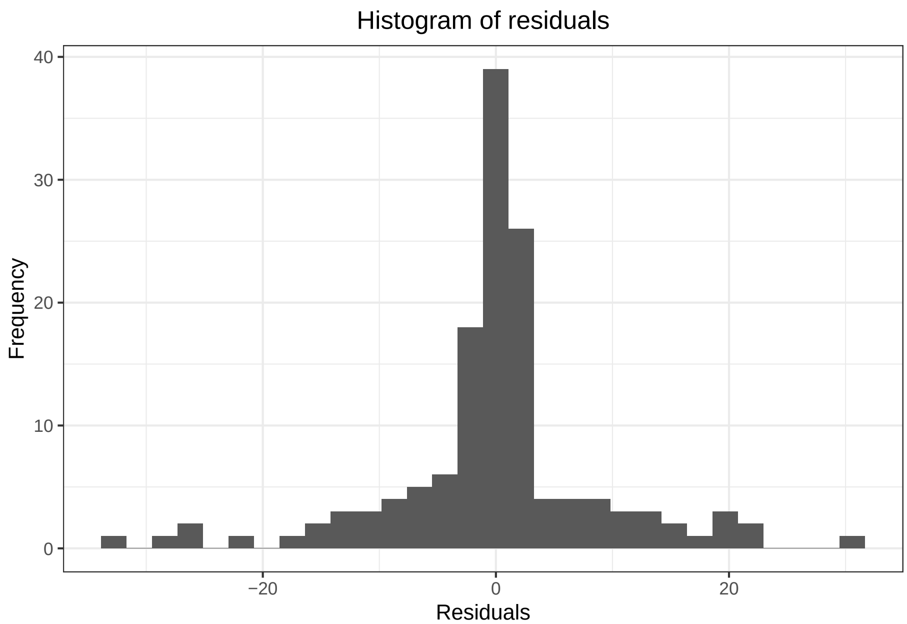
```
hist(driftm$residuals)
```



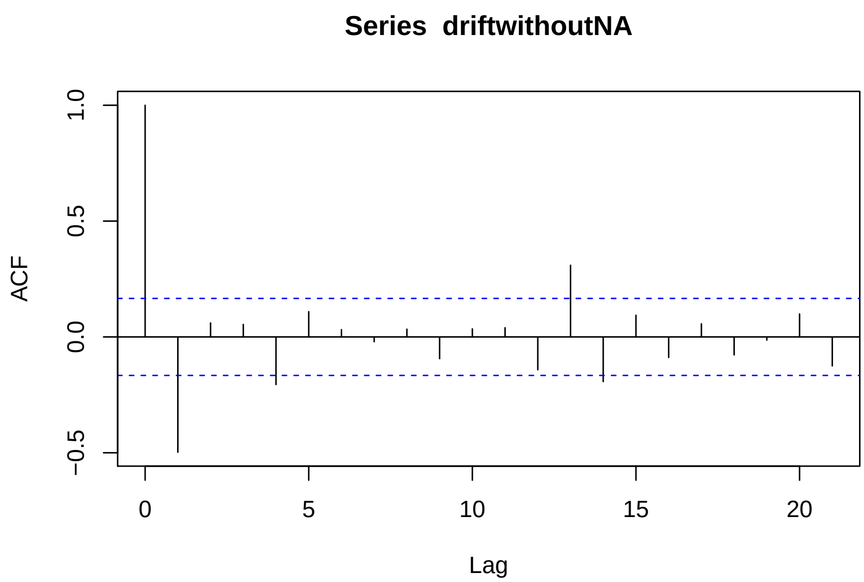

```
ggplot() +
  geom_histogram(aes(x = driftm$residuals)) +
  ggtitle("Histogram of residuals") +
  xlab("Residuals") + ylab("Frequency") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

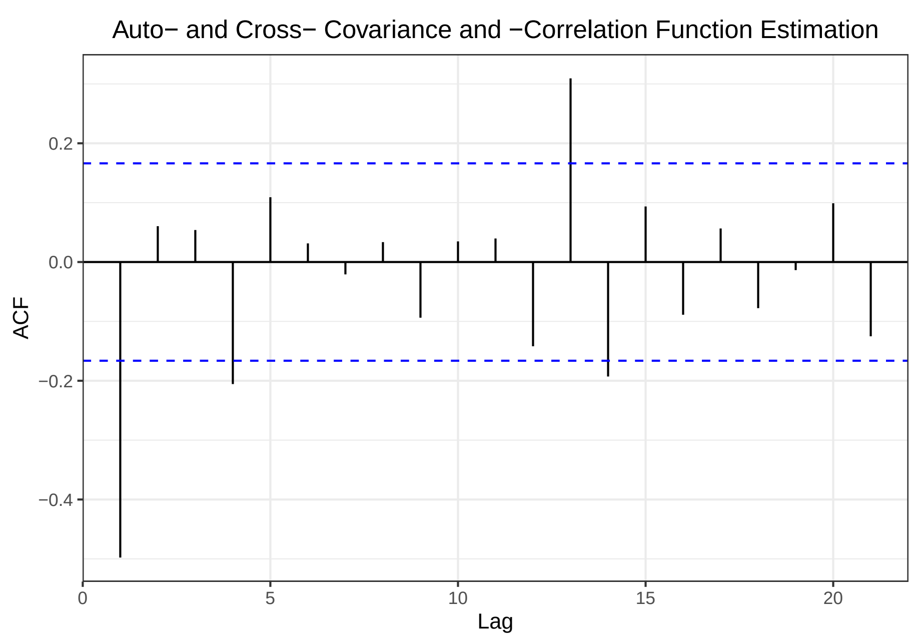
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to con
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```




```
acf(driftwithoutNA)
autoplot(acf(driftwithoutNA)) +
  ggtitle("Auto- and Cross- Covariance and -Correlation Function Estimation") +
  xlab("Lag") + ylab("ACF") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```






```
#6 check all relevant statistical traits
mytstrain <- window(myts, start = 1, end = 112 )
mytstest <- window(myts, start = 113)

meanma <- meanf(mytstrain, h=28)
```

```r
naivema <- naive(mytstrain, h=28)
driftma <- rwf(mytstrain, h=28, drift = T)
accuracy(meanma, myts	test)
```

```
##                         ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -6.408719e-16 15.31467 13.94736  -84.86989 120.4406 2.231924
## Test set     -1.125187e+01 11.61073 11.25187 -180.27778 180.2778 1.800578
##                    ACF1 Theil's U
## Training set  0.7632991        NA
## Test set     -0.1703445  2.002248
```

```r
accuracy(naivema, mytstest)
```

```
##                      ME      RMSE      MAE        MPE     MAPE      MASE
## Training set -0.4131666 10.024449 6.249032 -11.909758 33.36297 1.0000000
## Test set      0.9663084  3.022963 2.482045  -1.869095 33.51426 0.3971888
##                    ACF1 Theil's U
## Training set -0.4901263        NA
## Test set     -0.1703445 0.6497522
```

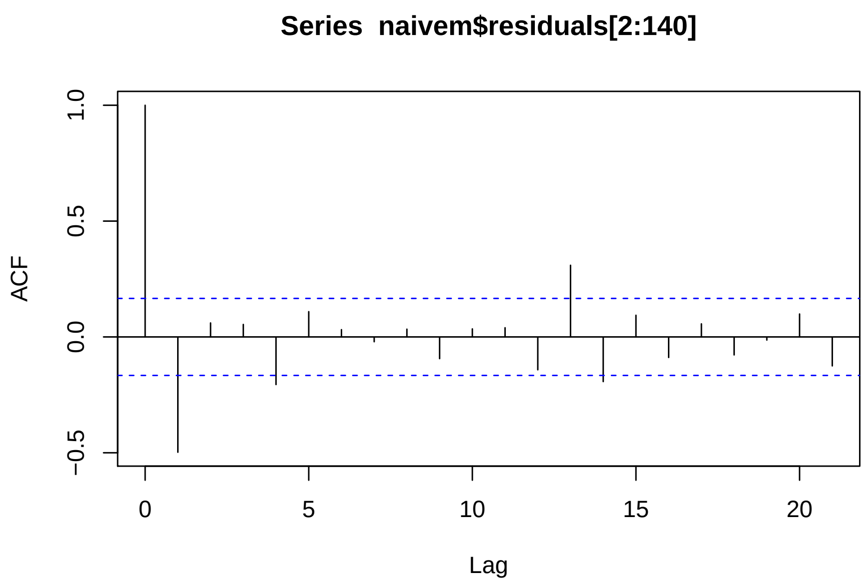
```r
accuracy(driftma, mytstest)
```

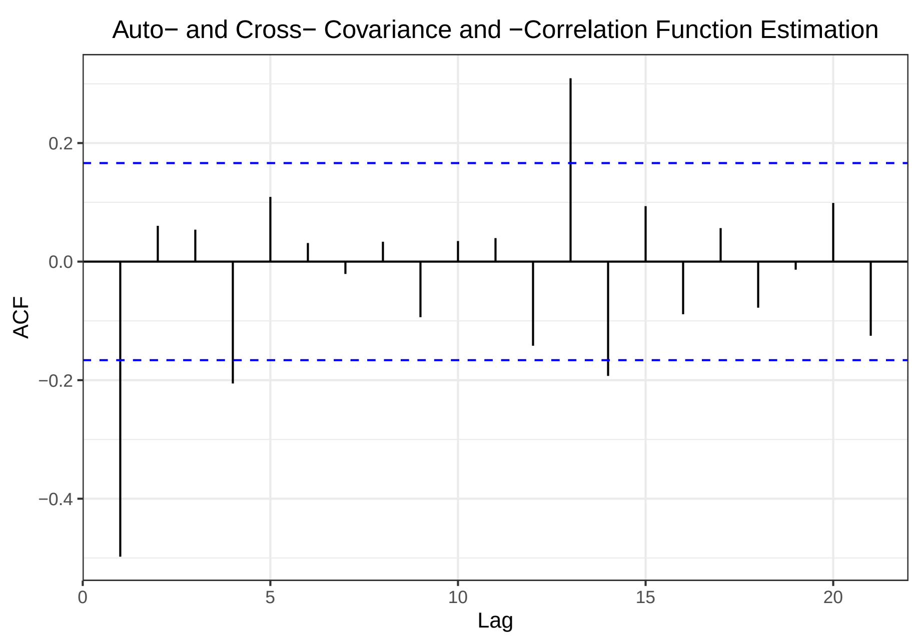
```
##                         ME      RMSE      MAE       MPE     MAPE     MASE
## Training set -1.624594e-17 10.015931 6.265918 -7.901602 33.29565 1.002702
## Test set      6.957224e+00  8.172915 6.974530 86.321252 86.72796 1.116098
##                    ACF1 Theil's U
## Training set -0.4901263        NA
## Test set      0.4327471   1.62159
```

```r
shapiro.test(naivem$residuals) # test for normal distribution, normal distr can be rejected
```

```
##
##  Shapiro-Wilk normality test
##
## data:  naivem$residuals
## W = 0.89587, p-value = 2.061e-08
```

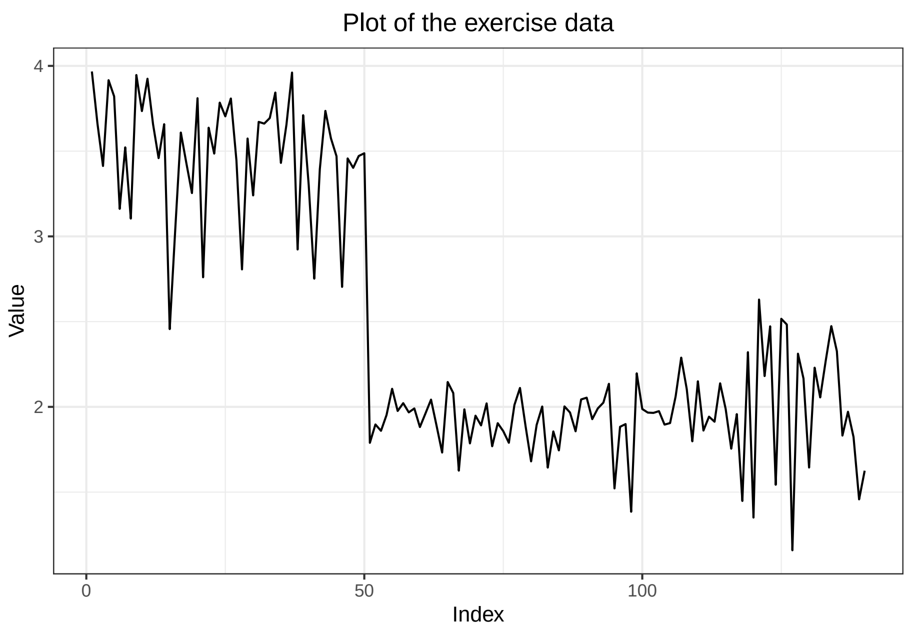
```r
acf(naivem$residuals[2:140]) # autocorrelation test, autocorrelation present
autoplot(acf(naivem$residuals[2:140])) +
  ggtitle("Auto- and Cross- Covariance and -Correlation Function Estimation") +
  xlab("Lag") + ylab("ACF") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```






```
#Repeating with the log of the data
 myts <- log(myts)
 #1. Plot the data and examine it.
 autoplot(myts) +
   ggtitle("Plot of the exercise data") +
```

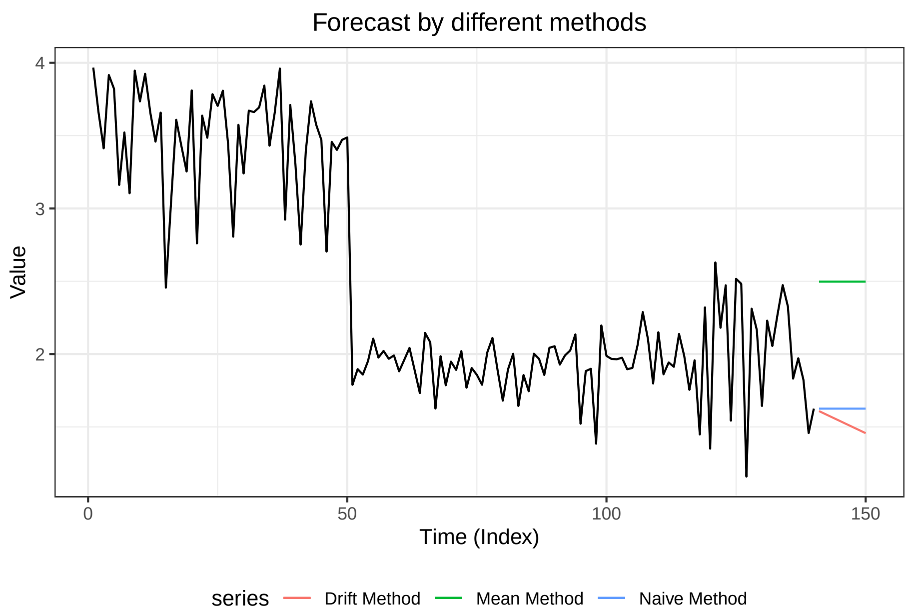
```
  xlab("Index") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```




```
#2. three forecasting models
meanm <- meanf(myts, h = 10)
naivem <- naive(myts, h= 10)
driftm <- rwf(myts, h =10, drift = TRUE)

#3. plot forecasts from three forecasting models
forecast::autoplot(myts,
                   PI = TRUE,
                   flwd = 2) +
  autolayer(meanm$mean, series = "Mean Method", PI = TRUE) +
  autolayer(naivem$mean, series = "Naive Method", PI = TRUE) +
  autolayer(driftm$mean, series = "Drift Method", PI = TRUE) +
  ggtitle("Forecast by different methods") +
  xlab("Time (Index)") + ylab("Value") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

```
## Warning: Ignoring unknown parameters: PI, flwd
```

```
## Warning: Ignoring unknown parameters: PI

## Warning: Ignoring unknown parameters: PI

## Warning: Ignoring unknown parameters: PI
```




```r
#4. Which model looks most promising

#5. Get the error measures and compare them
var(meanm$residuals)
```

```
## [1] 0.6290444
```

```r
mean(meanm$residuals)
```

```
## [1] 1.510731e-16
```

```r
mean(naivem$residuals)
```

```
## [1] NA
```

```
naivwithoutNA <- naivem$residuals
naivwithoutNA <- naivwithoutNA[2:140] #naive and drift models need one observation to start with
var(naivwithoutNA)
```

```
## [1] 0.2067372
```

```
mean(naivwithoutNA)
```

```
## [1] -0.01684688
```

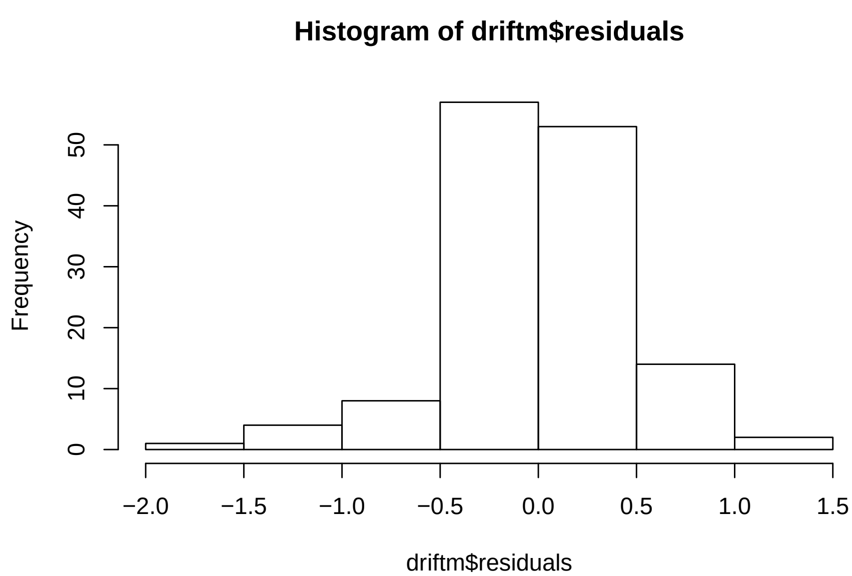
```
driftwithoutNA <- driftm$residuals
driftwithoutNA <- driftwithoutNA[2:140]
var(driftwithoutNA)
```

```
## [1] 0.2067372
```

```
mean(driftwithoutNA)
```

```
## [1] -4.472841e-17
```

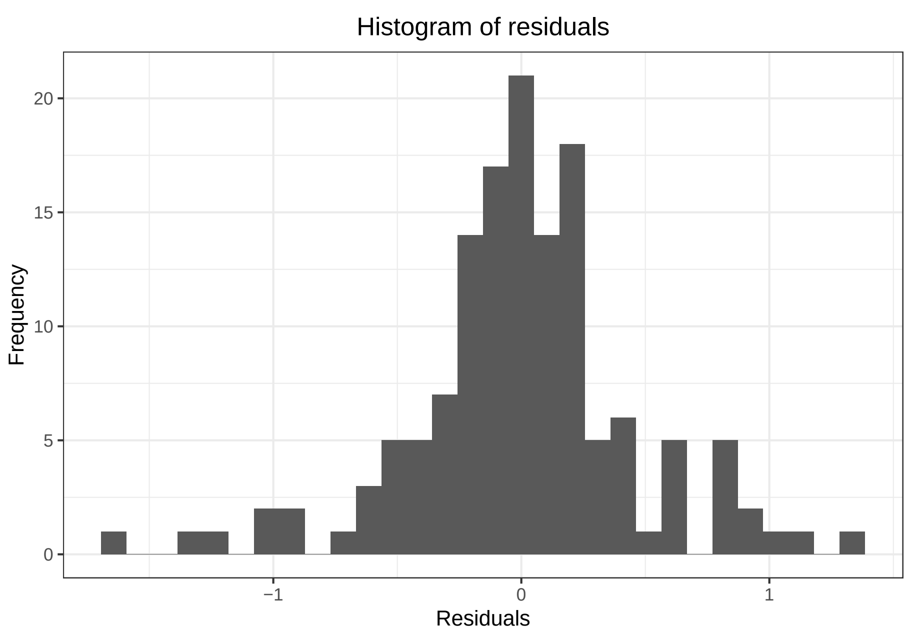
```
hist(driftm$residuals)
```



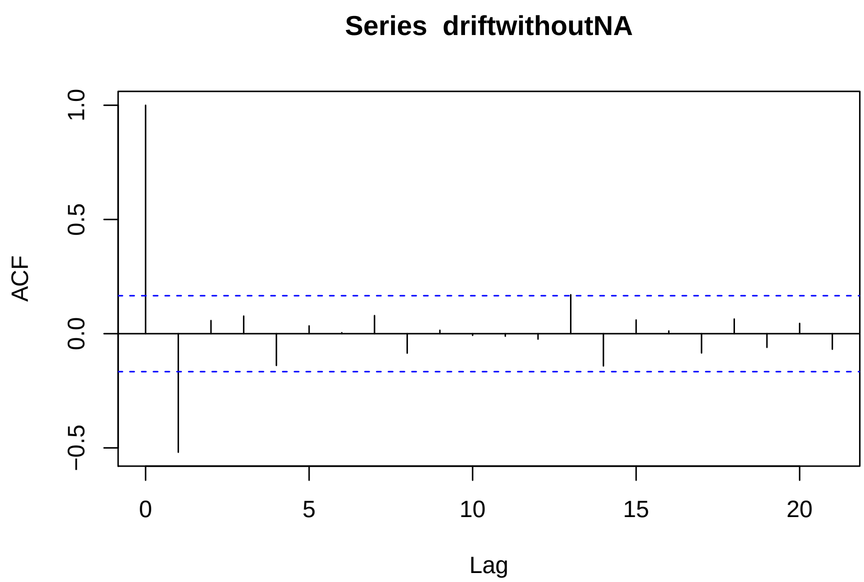
```
ggplot() +
  geom_histogram(aes(x = driftm$residuals)) +
  ggtitle("Histogram of residuals") +
  xlab("Residuals") + ylab("Frequency") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

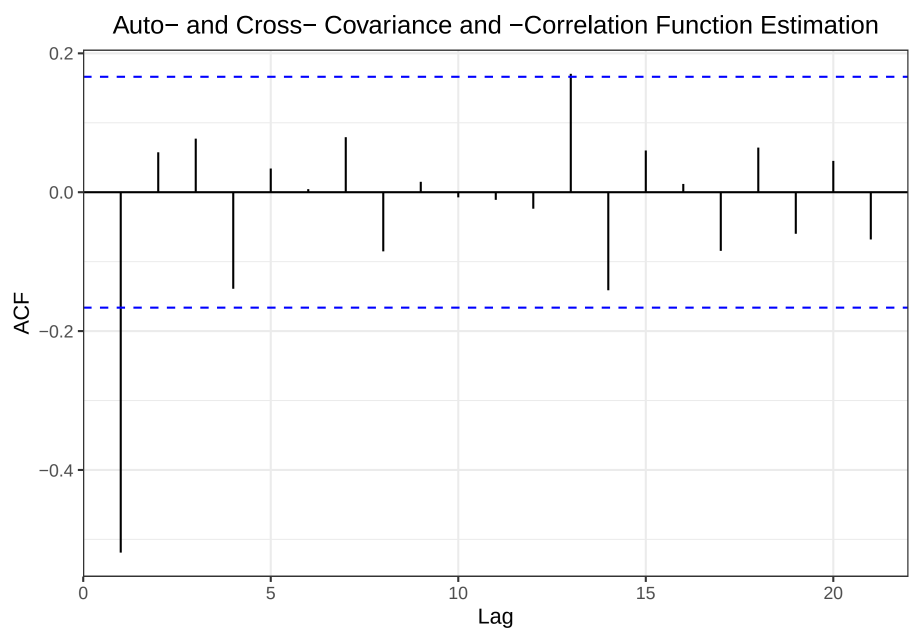
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to cont
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```




```
acf(driftwithoutNA)
autoplot(acf(driftwithoutNA)) +
  ggtitle("Auto- and Cross- Covariance and -Correlation Function Estimation") +
  xlab("Lag") + ylab("ACF") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```






```r
#6 check all relevant statistical traits
mytstrain <- window(myts, start = 1, end = 112 )
mytstest <- window(myts, start = 113)

meanma <- meanf(mytstrain, h=28)
```

```
naivema <- naive(mytstrain, h=28)
driftma <- rwf(mytstrain, h=28, drift = T)
accuracy(meanma, mytstest)
```

```
##                           ME      RMSE       MAE        MPE     MAPE
## Training set  1.846038e-16 0.8163346 0.7714004  -9.839234 31.23484
## Test set     -6.198835e-01 0.7307518 0.6204553 -36.737960 36.75971
##                  MASE       ACF1 Theil's U
## Training set 2.684607  0.8615304        NA
## Test set     2.159291 -0.2306541 0.9716032
```

```
accuracy(naivema, mytstest)
```

```
##                       ME      RMSE       MAE       MPE     MAPE     MASE
## Training set -0.01824047 0.4071185 0.2873421 -1.870232 11.33626 1.000000
## Test set      0.05893104 0.3914276 0.3316489 -1.328849 17.76316 1.154196
##                    ACF1 Theil's U
## Training set -0.4450652        NA
## Test set     -0.2306541  0.577168
```

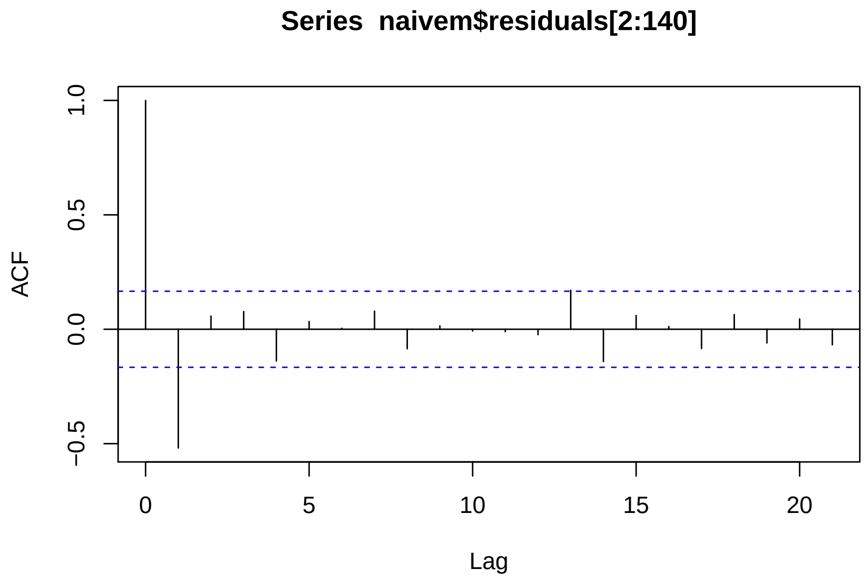
```
accuracy(driftma, mytstest)
```

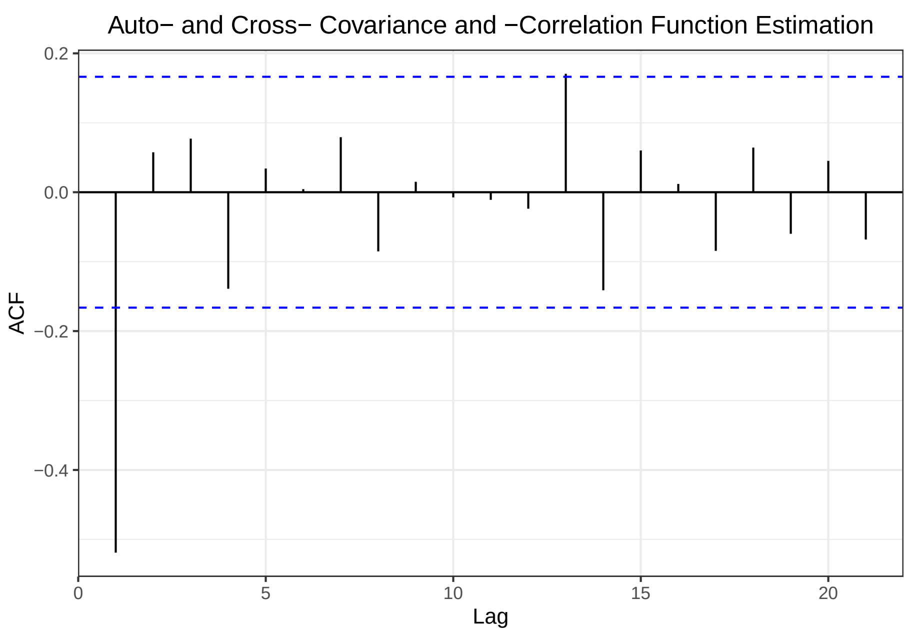
```
##                         ME      RMSE       MAE       MPE     MAPE     MASE
## Training set -9.802537e-17 0.4067096 0.2876207 -1.103181 11.31901 1.000970
## Test set      3.234179e-01 0.5206086 0.4411263 12.524858 21.27240 1.535196
##                    ACF1 Theil's U
## Training set -0.4450652        NA
## Test set     -0.1049056 0.7824451
```

```
shapiro.test(naivem$residuals) # test for normal distribution, normal distr can be re
```

```
##
##  Shapiro-Wilk normality test
##
## data:  naivem$residuals
## W = 0.961, p-value = 0.0005413
```

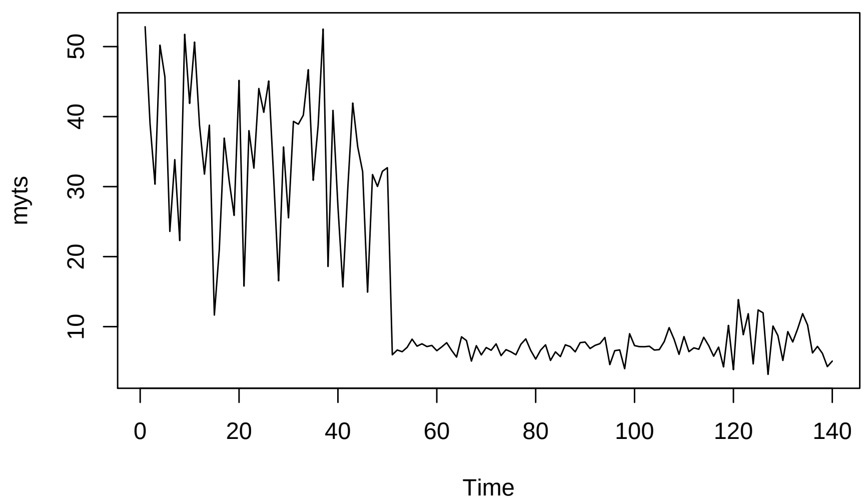
```
acf(naivem$residuals[2:140]) # autocorrelation test, autocorrelation present
autoplot(acf(naivem$residuals[2:140])) +
  ggtitle("Auto- and Cross- Covariance and -Correlation Function Estimation") +
  xlab("Lag") + ylab("ACF") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```



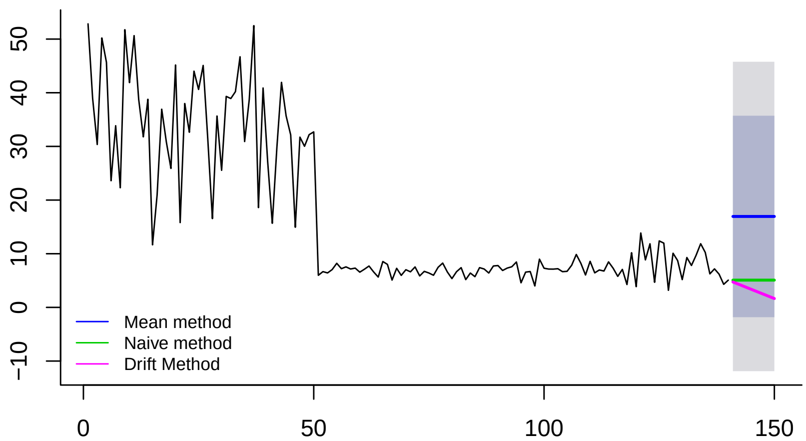



```r
#==============================================================================
set.seed(54)
myts <- ts(c(rnorm(50, 34, 10),
             rnorm(67, 7, 1),
             runif(23, 3, 14)))
```

```
plot(myts)
```




```
library(forecast)
meanm <- meanf(myts, h=10)
naivem <- naive(myts, h=10)
driftm <- rwf(myts, h=10, drift = T)

plot(meanm, main = "", bty = "l")
lines(naivem$mean, col=123, lwd = 2)
lines(driftm$mean, col=22, lwd = 2)
legend("bottomleft",lty=1,col=c(4,123,22), bty = "n", cex = 0.75,
       legend=c("Mean method","Naive method","Drift Method"))
```




```
length(myts)
```

```
## [1] 140
```

```
mytstrain <- window(myts, start = 1, end = 112 )
mytstest <- window(myts, start = 113)

meanma <- meanf(mytstrain, h=28)
naivema <- naive(mytstrain, h=28)
driftma <- rwf(mytstrain, h=28, drift = T)

accuracy(meanma, mytstest)
```

```
##                           ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -6.408719e-16 15.31467 13.94736  -84.86989 120.4406 2.231924
## Test set     -1.125187e+01 11.61073 11.25187 -180.27778 180.2778 1.800578
##                    ACF1 Theil's U
## Training set  0.7632991        NA
## Test set     -0.1703445  2.002248
```

```
accuracy(naivema, mytstest)
```

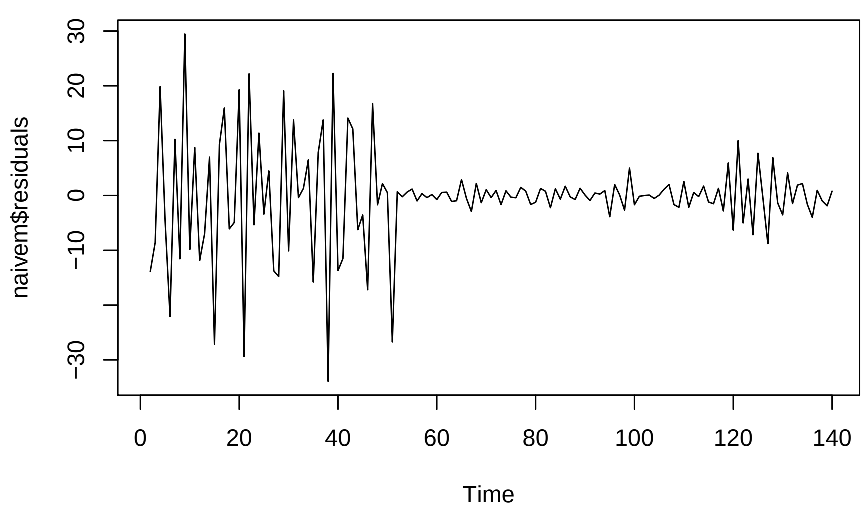
```
##                      ME       RMSE        MAE          MPE       MAPE        MASE
```

```
## Training set -0.4131666 10.024449 6.249032 -11.909758 33.36297 1.0000000
## Test set      0.9663084  3.022963 2.482045  -1.869095 33.51426 0.3971888
##                    ACF1 Theil's U
## Training set -0.4901263        NA
## Test set     -0.1703445 0.6497522
```

```
accuracy(driftma, mytstest)
```

```
##                         ME      RMSE      MAE       MPE     MAPE     MASE
## Training set -1.624594e-17 10.015931 6.265918 -7.901602 33.29565 1.002702
## Test set      6.957224e+00  8.172915 6.974530 86.321252 86.72796 1.116098
##                    ACF1 Theil's U
## Training set -0.4901263        NA
## Test set      0.4327471   1.62159
```

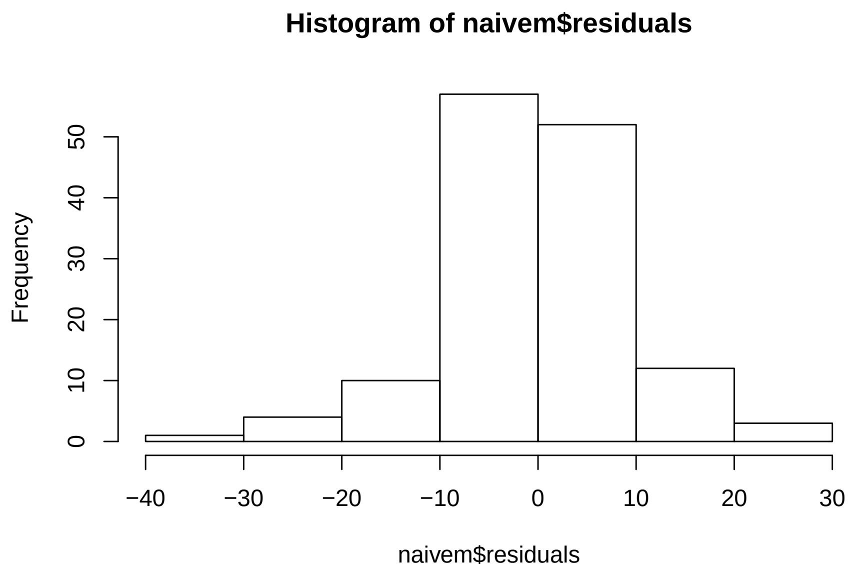
```
plot(naivem$residuals)
```




```
mean(naivem$residuals[2:140])
```

```
## [1] -0.3435748
```

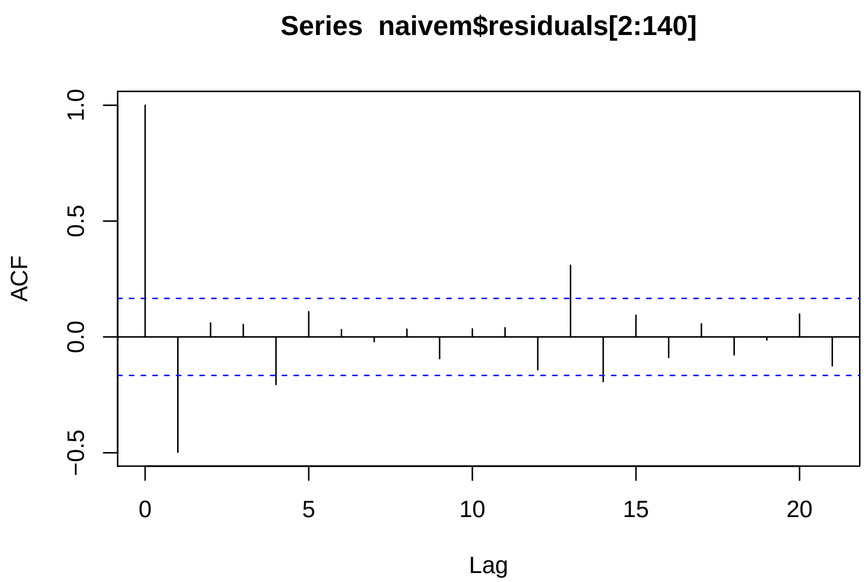
```r
hist(naivem$residuals) # normal distribution
```




```r
shapiro.test(naivem$residuals) # test for normal distribution, normal distr can be rejected
```

```
##
##  Shapiro-Wilk normality test
##
## data:  naivem$residuals
## W = 0.89587, p-value = 2.061e-08
```

```r
acf(naivem$residuals[2:140]) # autocorrelation test, autocorrelation present
```

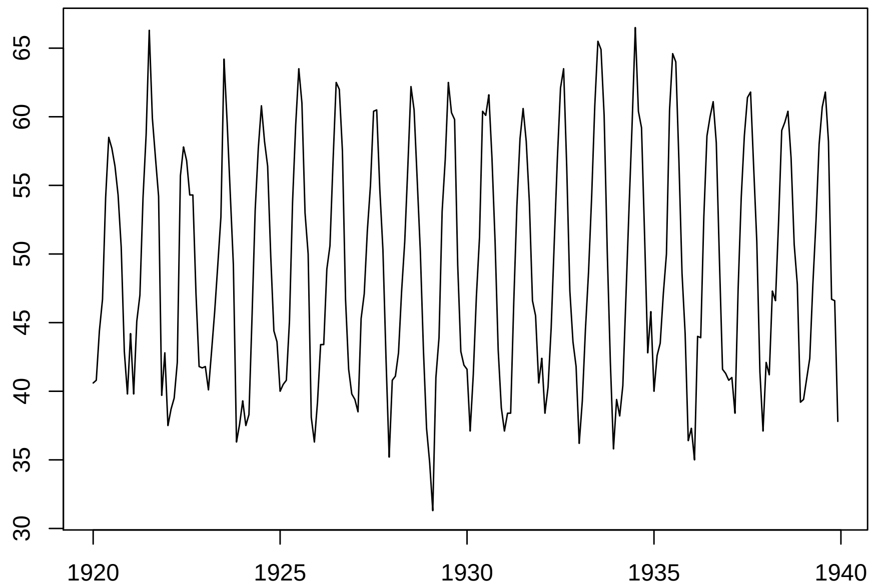

Series naivem$residuals[2:140]
ACF
1.0
0.5
0.0
−0.5
0
5
10
15
20
Lag

Chapter 5

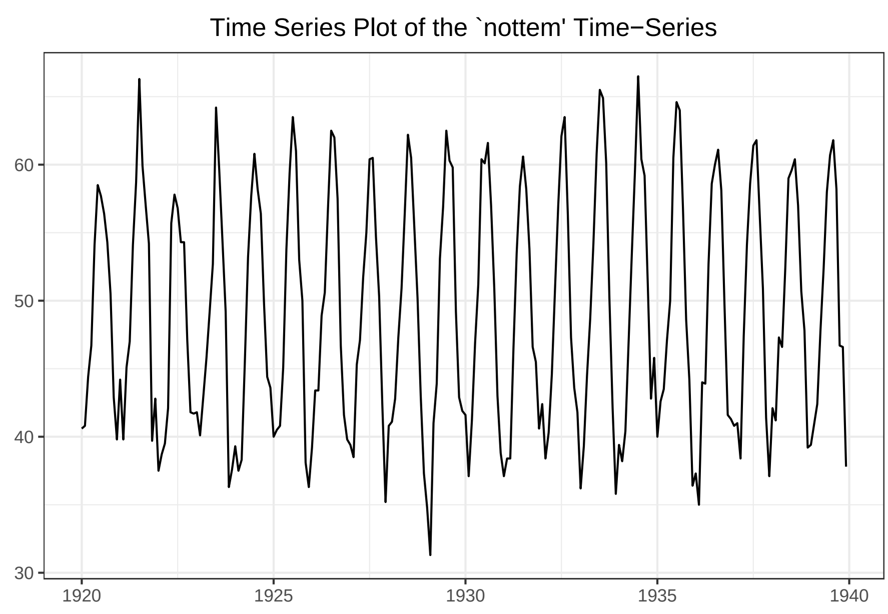
TS Analysis And Forecasting

```
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section5")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
require(lmtest)
```

```
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section5")
par(mar = rep(2, 4))
### Decomposing Time Series (U)
plot(nottem)
```




```
theme_set(theme_bw())
autoplot(nottem) + xlab("") + ylab("") + ggtitle("Time Series Plot of the `nottem' Time
    theme(plot.title = element_text(hjust = 0.5))  #for centering the text
```



```r
frequency(nottem)  #whether the data is monthly, quarterly or anything else
```

```
## [1] 12
```

```r
length(nottem)  #number of observations in the data
```

```
## [1] 240
```

```r
decompose(nottem, type = "additive")  #decompose the time series in seasonal, trend and random co
```

```
## $x
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1920 40.6 40.8 44.4 46.7 54.1 58.5 57.7 56.4 54.3 50.5 42.9 39.8
## 1921 44.2 39.8 45.1 47.0 54.1 58.7 66.3 59.9 57.0 54.2 39.7 42.8
## 1922 37.5 38.7 39.5 42.1 55.7 57.8 56.8 54.3 54.3 47.1 41.8 41.7
## 1923 41.8 40.1 42.9 45.8 49.2 52.7 64.2 59.6 54.4 49.2 36.3 37.6
## 1924 39.3 37.5 38.3 45.5 53.2 57.7 60.8 58.2 56.4 49.8 44.4 43.6
## 1925 40.0 40.5 40.8 45.1 53.8 59.4 63.5 61.0 53.0 50.0 38.1 36.3
## 1926 39.2 43.4 43.4 48.9 50.6 56.8 62.5 62.0 57.5 46.7 41.6 39.8
## 1927 39.4 38.5 45.3 47.1 51.7 55.0 60.4 60.5 54.7 50.3 42.3 35.2
## 1928 40.8 41.1 42.8 47.3 50.9 56.4 62.2 60.5 55.4 50.2 43.0 37.3
## 1929 34.8 31.3 41.0 43.9 53.1 56.9 62.5 60.3 59.8 49.2 42.9 41.9
## 1930 41.6 37.1 41.2 46.9 51.2 60.4 60.1 61.6 57.0 50.9 43.0 38.8
## 1931 37.1 38.4 38.4 46.5 53.5 58.4 60.6 58.2 53.8 46.6 45.5 40.6
## 1932 42.4 38.4 40.3 44.6 50.9 57.0 62.1 63.5 56.3 47.3 43.6 41.8
## 1933 36.2 39.3 44.5 48.7 54.2 60.8 65.5 64.9 60.1 50.2 42.1 35.8
## 1934 39.4 38.2 40.4 46.9 53.4 59.6 66.5 60.4 59.2 51.2 42.8 45.8
## 1935 40.0 42.6 43.5 47.1 50.0 60.5 64.6 64.0 56.8 48.6 44.2 36.4
## 1936 37.3 35.0 44.0 43.9 52.7 58.6 60.0 61.1 58.1 49.6 41.6 41.3
## 1937 40.8 41.0 38.4 47.4 54.1 58.6 61.4 61.8 56.3 50.9 41.4 37.1
## 1938 42.1 41.2 47.3 46.6 52.4 59.0 59.6 60.4 57.0 50.7 47.8 39.2
## 1939 39.4 40.9 42.4 47.8 52.4 58.0 60.7 61.8 58.2 46.7 46.6 37.8
##
## $seasonal
##             Jan        Feb        Mar        Apr        May        Jun
## 1920 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1921 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1922 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1923 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1924 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1925 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1926 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1927 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
```

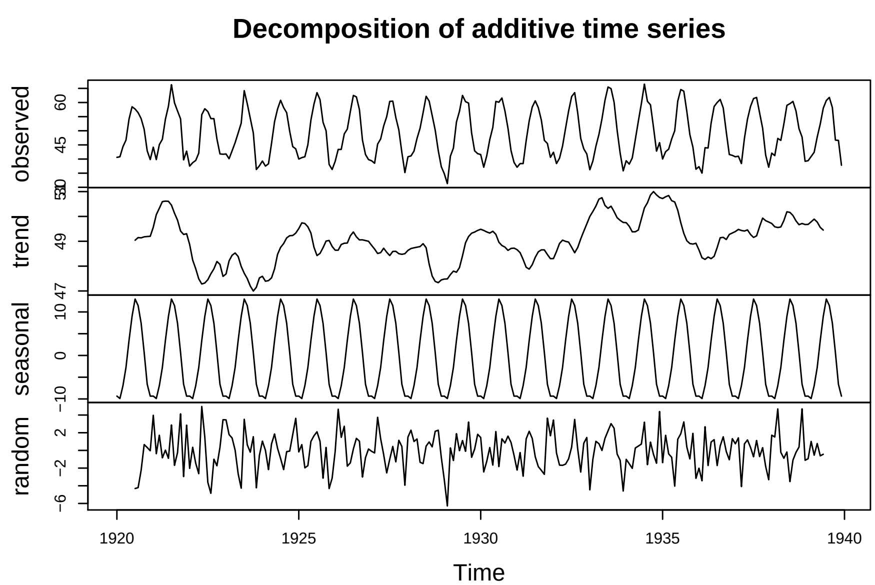
```
## 1928 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1929 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1930 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1931 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1932 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1933 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1934 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1935 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1936 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1937 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1938 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
## 1939 -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
##             Jul        Aug        Sep        Oct        Nov        Dec
## 1920 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1921 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1922 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1923 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1924 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1925 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1926 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1927 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1928 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1929 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1930 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1931 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1932 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1933 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1934 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1935 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1936 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1937 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1938 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
## 1939 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
##
## $trend
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1920       NA       NA       NA       NA       NA       NA 49.04167
## 1921 49.56667 50.07083 50.32917 50.59583 50.61667 50.60833 50.45417
## 1922 48.87083 48.24167 47.89583 47.48750 47.27917 47.32083 47.45417
## 1923 47.68333 48.21250 48.43750 48.52917 48.38750 47.98750 47.71250
## 1924 47.59167 47.39167 47.41667 47.52500 47.88750 48.47500 48.75417
## 1925 49.51250 49.74167 49.71667 49.58333 49.32917 48.76250 48.42500
## 1926 48.64167 48.64167 48.87083 48.92083 48.92917 49.22083 49.37500
## 1927 48.83750 48.68750 48.50833 48.54167 48.72083 48.55833 48.42500
## 1928 48.63333 48.70833 48.73750 48.76250 48.78750 48.90417 48.74167
## 1929 47.47917 47.48333 47.65833 47.80000 47.75417 47.94167 48.41667
```

```
## 1930 49.48333 49.43750 49.37500 49.32917 49.40417 49.27917 48.96250
## 1931 48.66250 48.54167 48.26667 47.95417 47.87917 48.05833 48.35417
## 1932 48.30417 48.58750 48.91250 49.04583 48.99583 48.96667 48.75833
## 1933 50.00000 50.20000 50.41667 50.69583 50.75417 50.44167 50.32500
## 1934 49.75000 49.60417 49.37917 49.38333 49.45417 49.90000 50.34167
## 1935 50.72083 50.79167 50.84167 50.63333 50.58333 50.25000 49.74583
## 1936 48.65000 48.33750 48.27083 48.36667 48.30000 48.39583 48.74583
## 1937 49.39167 49.47917 49.43333 49.41250 49.45833 49.27500 49.15417
## 1938 49.71667 49.58333 49.55417 49.57500 49.83333 50.18750 50.16250
## 1939 49.67917 49.78333 49.89167 49.77500 49.55833 49.45000       NA
##           Aug      Sep      Oct      Nov      Dec
## 1920 49.15000 49.13750 49.17917 49.19167 49.20000
## 1921 50.12917 49.85000 49.41250 49.27500 49.30417
## 1922 47.69167 47.89167 48.18750 48.07083 47.58750
## 1923 47.50000 47.20000 46.99583 47.15000 47.52500
## 1924 48.90833 49.13750 49.22500 49.23333 49.32917
## 1925 48.51250 48.74167 49.00833 49.03333 48.79167
## 1926 49.17917 49.05417 49.05833 49.02917 49.00000
## 1927 48.59167 48.59583 48.50000 48.47500 48.50000
## 1928 48.08333 47.60000 47.38333 47.33333 47.44583
## 1929 48.94167 49.19167 49.32500 49.37083 49.43750
## 1930 48.82917 48.76667 48.63333 48.71250 48.72500
## 1931 48.57500 48.65417 48.65417 48.46667 48.30000
## 1932 48.53750 48.75000 49.09583 49.40417 49.70000
## 1933 50.41250 50.19583 49.95000 49.84167 49.75833
## 1934 50.55000 50.86250 51.00000 50.86667 50.76250
## 1935 49.31667 49.02083 48.90833 48.88750 48.92083
## 1936 49.14167 49.15833 49.07083 49.27500 49.33333
## 1937 49.21667 49.59583 49.93333 49.82917 49.77500
## 1938 50.03750 49.82083 49.66667 49.71667 49.67500
## 1939       NA       NA       NA       NA       NA
##
## $random
##               Jan          Feb          Mar          Apr          May
## 1920           NA           NA           NA           NA           NA
## 1921  3.972697368 -0.370942982  1.717434211 -0.838486842  0.029934211
## 1922 -2.031469298  0.358223684 -1.449232456 -2.630153509  4.967434211
## 1923  3.456030702  1.787390351  1.409100877  0.028179825 -2.640899123
## 1924  1.047697368  0.008223684 -2.170065789  0.732346491  1.859100877
## 1925 -0.173135965  0.658223684 -1.970065789 -1.725986842  1.017434211
## 1926 -0.102302632  4.658223684  1.475767544  2.736513158 -1.782565789
## 1927 -0.098135965 -0.287609649  3.738267544  1.315679825 -0.474232456
## 1928  1.506030702  2.291557018  1.009100877  1.294846491 -1.340899123
## 1929 -3.339802632 -6.283442982  0.288267544 -1.142653509  1.892434211
## 1930  1.456030702 -2.437609649 -1.228399123  0.328179825 -1.657565789
## 1931 -2.223135965 -0.241776316 -2.920065789  1.303179825  2.167434211
```

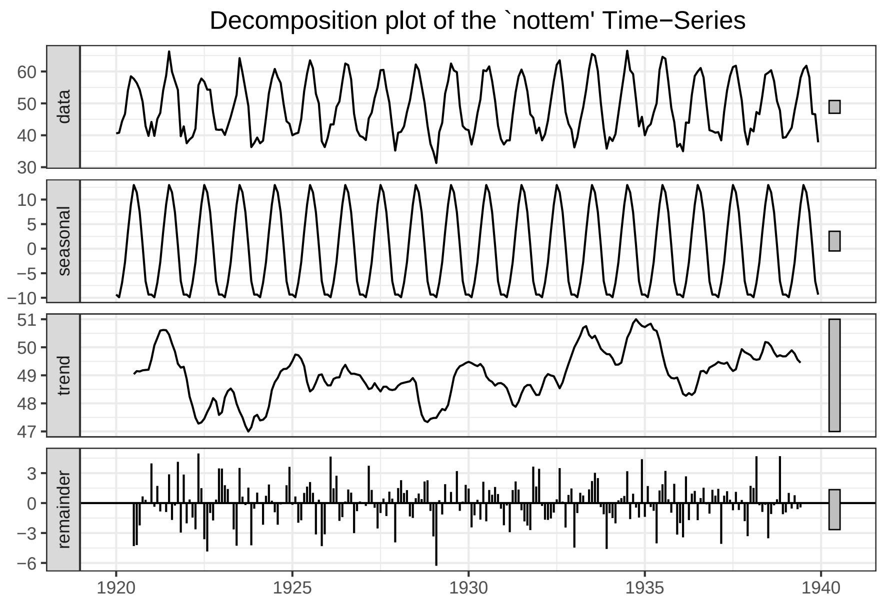
```
## 1932  3.435197368 -0.287609649 -1.665899123 -1.688486842 -1.549232456
## 1933 -4.460635965 -1.000109649  1.029934211  0.761513158 -0.007565789
## 1934 -1.010635965 -1.504276316 -2.032565789  0.274013158  0.492434211
## 1935 -1.381469298  1.708223684 -0.395065789 -0.775986842 -4.036732456
## 1936 -2.010635965 -3.437609649  2.675767544 -1.709320175  0.946600877
## 1937  0.747697368  1.420723684 -4.086732456  0.744846491  1.188267544
## 1938  1.722697368  1.516557018  4.692434211 -0.217653509 -0.886732456
## 1939 -0.939802632  1.016557018 -0.545065789  0.782346491 -0.611732456
##               Jun          Jul          Aug          Sep          Oct
## 1920           NA -4.308881579 -4.209100877 -2.237609649  0.666118421
## 1921 -0.894846491  2.878618421 -1.688267544 -0.250109649  4.132785088
## 1922  1.492653509 -3.621381579 -4.850767544 -0.991776316 -1.742214912
## 1923 -4.274013158  3.520285088  0.640899123 -0.200109649  1.549451754
## 1924  0.238486842 -0.921381579 -2.167434211 -0.137609649 -0.079714912
## 1925  1.650986842  2.107785088  1.028399123 -3.141776316  0.336951754
## 1926 -1.407346491  0.157785088  1.361732456  1.045723684 -3.013048246
## 1927 -2.544846491 -0.992214912  0.449232456 -1.295942982  1.145285088
## 1928 -1.490679825  0.491118421  0.957565789  0.399890351  2.161951754
## 1929 -0.028179825  1.116118421 -0.100767544  3.208223684 -0.779714912
## 1930  2.134320175 -1.829714912  1.311732456  0.833223684  1.611951754
## 1931  1.355153509 -0.721381579 -1.834100877 -2.254276316 -2.708881579
## 1932 -0.953179825  0.374451754  3.503399123  0.149890351 -2.450548246
## 1933  1.371820175  2.207785088  3.028399123  2.504057018 -0.404714912
## 1934  0.713486842  3.191118421 -1.609100877  0.937390351 -0.454714912
## 1935  1.263486842  1.886951754  3.224232456  0.379057018 -0.963048246
## 1936  1.217653509 -1.713048246  0.499232456  1.541557018 -0.125548246
## 1937  0.338486842 -0.721381579  1.124232456 -0.695942982  0.311951754
## 1938 -0.174013158 -3.529714912 -1.096600877 -0.220942982  0.378618421
## 1939 -0.436513158           NA           NA           NA           NA
##               Nov          Dec
## 1920  0.325986842 -0.039802632
## 1921 -2.957346491  2.856030702
## 1922  0.346820175  3.472697368
## 1923 -4.232346491 -0.564802632
## 1924  1.784320175  3.631030702
## 1925 -4.315679825 -3.131469298
## 1926 -0.811513158  0.160197368
## 1927  0.442653509 -3.939802632
## 1928  2.284320175 -0.785635965
## 1929  0.146820175  1.822697368
## 1930  0.905153509 -0.564802632
## 1931  3.650986842  1.660197368
## 1932  0.813486842  1.460197368
## 1933 -1.124013158 -4.598135965
## 1934 -1.449013158  4.397697368
## 1935  1.930153509 -3.160635965
```

```
## 1936 -1.057346491  1.326864035
## 1937 -1.811513158 -3.314802632
## 1938  4.700986842 -1.114802632
## 1939           NA           NA
##
## $figure
##  [1] -9.3393640 -9.8998904 -6.9466009 -2.7573465  3.4533991  8.9865132
##  [7] 12.9672149 11.4591009  7.4001096  0.6547149 -6.6176535 -9.3601974
##
## $type
## [1] "additive"
##
## attr(,"class")
## [1] "decomposed.ts"
```

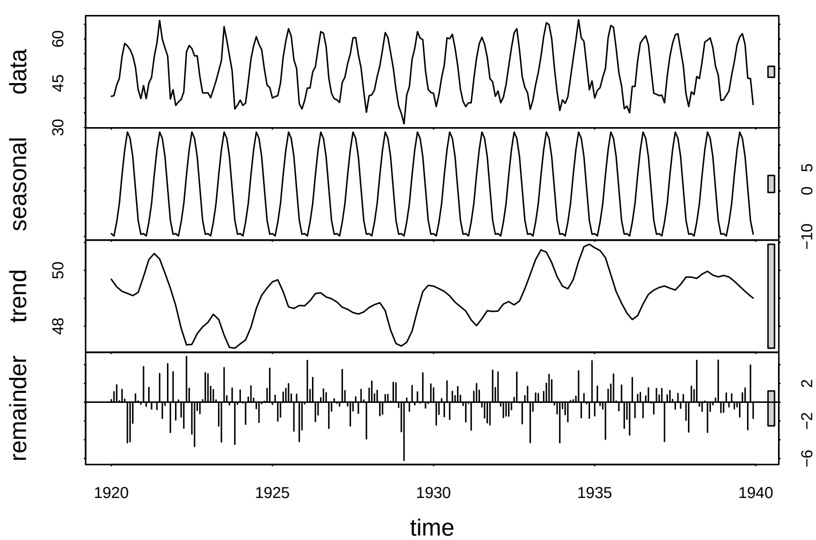
```r
plot(decompose(nottem, type = "additive"))
```



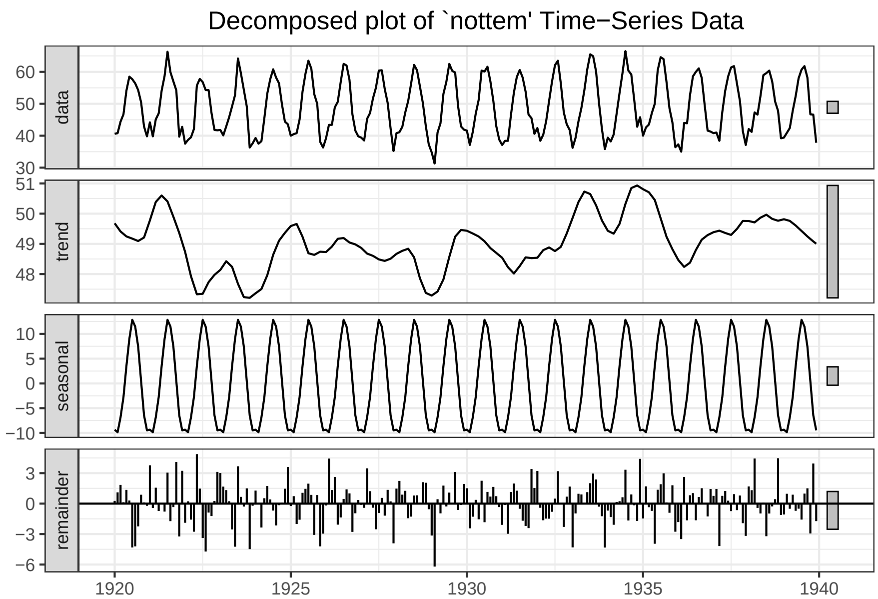

```r
autoplot(decompose(nottem, type = "additive")) + xlab("") + ylab("") +
    ggtitle("Decomposition plot of the `nottem' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5))
```




```
# alternatively the function stl could be used
plot(stl(nottem, s.window = "periodic"))
```



```r
autoplot(stl(nottem, s.window = "periodic")) + xlab("") + ylab("") +
    ggtitle("Decomposed plot of `nottem' Time-Series Data") +
    theme(plot.title = element_text(hjust = 0.5))
```




```r
stl(nottem, s.window = "periodic")
```

```
##  Call:
##  stl(x = nottem, s.window = "periodic")
##
## Components
##              seasonal    trend    remainder
## Jan 1920 -9.3471980 49.68067  0.266525379
## Feb 1920 -9.8552496 49.54552  1.109728805
## Mar 1920 -6.8533008 49.41037  1.842931803
## Apr 1920 -2.7634710 49.32862  0.134848770
## May 1920  3.5013569 49.24688  1.351767558
## Jun 1920  8.9833032 49.21027  0.306425938
## Jul 1920 12.8452501 49.17367 -4.318916345
## Aug 1920 11.4763813 49.13389 -4.210271506
## Sep 1920  7.4475114 49.09411 -2.241625601
## Oct 1920  0.4736899 49.15198  0.874331161
## Nov 1920 -6.4301309 49.20984  0.120287167
## Dec 1920 -9.4781423 49.49282 -0.214674402
```

```
## Jan 1921 -9.3471980 49.77579  3.771408356
## Feb 1921 -9.8552496 50.08132 -0.426073148
## Mar 1921 -6.8533008 50.38686  1.566444922
## Apr 1921 -2.7634710 50.49363 -0.730156884
## May 1921  3.5013569 50.60040 -0.001756869
## Jun 1921  8.9833032 50.50241 -0.785710552
## Jul 1921 12.8452501 50.40441  3.050335100
## Aug 1921 11.4763813 50.15065 -1.727027471
## Sep 1921  7.4475114 49.89688 -0.344388977
## Oct 1921  0.4736899 49.62951  4.096796830
## Nov 1921 -6.4301309 49.36215 -3.232018119
## Dec 1921 -9.4781423 49.04873  3.229417050
## Jan 1922 -9.3471980 48.73530 -1.888103454
## Feb 1922 -9.8552496 48.33172  0.223530715
## Mar 1922 -6.8533008 47.92814 -1.574835542
## Apr 1922 -2.7634710 47.62971 -2.766243150
## May 1922  3.5013569 47.33129  4.867351062
## Jun 1922  8.9833032 47.33932  1.477377256
## Jul 1922 12.8452501 47.34735 -3.392597215
## Aug 1922 11.4763813 47.54030 -4.716683254
## Sep 1922  7.4475114 47.73326 -0.880768228
## Oct 1922  0.4736899 47.85392 -1.227614475
## Nov 1922 -6.4301309 47.97459  0.255538521
## Dec 1922 -9.4781423 48.05637  3.121775401
## Jan 1923 -9.3471980 48.13814  3.009056606
## Feb 1923 -9.8552496 48.28185  1.673400681
## Mar 1923 -6.8533008 48.42556  1.327744328
## Apr 1923 -2.7634710 48.33263  0.230840836
## May 1923  3.5013569 48.23970 -2.541060836
## Jun 1923  8.9833032 47.96007 -4.243372853
## Jul 1923 12.8452501 47.68044  3.674314465
## Aug 1923 11.4763813 47.45963  0.663988147
## Sep 1923  7.4475114 47.23883 -0.286337106
## Oct 1923  0.4736899 47.22609  1.500223556
## Nov 1923 -6.4301309 47.21335 -4.483216539
## Dec 1923 -9.4781423 47.28901 -0.210865451
## Jan 1924 -9.3471980 47.36467  1.282529963
## Feb 1924 -9.8552496 47.43546 -0.080209799
## Mar 1924 -6.8533008 47.50625 -2.352949988
## Apr 1924 -2.7634710 47.73513  0.528338204
## May 1924  3.5013569 47.96401  1.734628216
## Jun 1924  8.9833032 48.30128  0.415415073
## Jul 1924 12.8452501 48.63855 -0.683798735
## Aug 1924 11.4763813 48.87239 -2.148774448
## Sep 1924  7.4475114 49.10624 -0.153749096
## Oct 1924  0.4736899 49.23554  0.090766554
```

```
## Nov 1924 -6.4301309 49.36485  1.465281448
## Dec 1924 -9.4781423 49.47714  3.601004136
## Jan 1925 -9.3471980 49.58943 -0.242228849
## Feb 1925 -9.8552496 49.62480  0.730453659
## Mar 1925 -6.8533008 49.66017 -2.006864260
## Apr 1925 -2.7634710 49.44799 -1.584518944
## May 1925  3.5013569 49.23581  1.062828193
## Jun 1925  8.9833032 48.96275  1.453949252
## Jul 1925 12.8452501 48.68968  1.965069647
## Aug 1925 11.4763813 48.66249  0.861133228
## Sep 1925  7.4475114 48.63529 -3.082802125
## Oct 1925  0.4736899 48.68778  0.838525570
## Nov 1925 -6.4301309 48.74028 -4.210147490
## Dec 1925 -9.4781423 48.73479 -2.956649644
## Jan 1926 -9.3471980 48.72931 -0.182107471
## Feb 1926 -9.8552496 48.81914  4.436114170
## Mar 1926 -6.8533008 48.90897  1.344335385
## Apr 1926 -2.7634710 49.03851  2.624958422
## May 1926  3.5013569 49.16806 -2.069416721
## Jun 1926  8.9833032 49.18067 -1.363972554
## Jul 1926 12.8452501 49.19328  0.461470949
## Aug 1926 11.4763813 49.11919  1.404424201
## Sep 1926  7.4475114 49.04511  1.007378518
## Oct 1926  0.4736899 49.01380 -2.787487357
## Nov 1926 -6.4301309 48.98248 -0.952353990
## Dec 1926 -9.4781423 48.92428  0.353859697
## Jan 1927 -9.3471980 48.86608 -0.118882290
## Feb 1927 -9.8552496 48.77246 -0.417208438
## Mar 1927 -6.8533008 48.67884  3.474464987
## Apr 1927 -2.7634710 48.64104  1.222435260
## May 1927  3.5013569 48.60324 -0.404592645
## Jun 1927  8.9833032 48.54473 -2.528029817
## Jul 1927 12.8452501 48.48622 -0.931467654
## Aug 1927 11.4763813 48.46091  0.562707318
## Sep 1927  7.4475114 48.43561 -1.183116646
## Oct 1927  0.4736899 48.47250  1.353805163
## Nov 1927 -6.4301309 48.50940  0.220726214
## Dec 1927 -9.4781423 48.58914 -3.911002529
## Jan 1928 -9.3471980 48.66888  1.478313054
## Feb 1928 -9.8552496 48.71949  2.235758285
## Mar 1928 -6.8533008 48.77010  0.883203089
## Apr 1928 -2.7634710 48.80480  1.258669662
## May 1928  3.5013569 48.83951 -1.440861943
## Jun 1928  8.9833032 48.69762 -1.280921267
## Jul 1928 12.8452501 48.55573  0.799018744
## Aug 1928 11.4763813 48.20763  0.815985704
```

```
## Sep 1928  7.4475114 47.85953  0.092953729
## Oct 1928  0.4736899 47.61843  2.107877490
## Nov 1928 -6.4301309 47.37733  2.052800495
## Dec 1928 -9.4781423 47.33335 -0.555212465
## Jan 1929 -9.3471980 47.28938 -3.142181098
## Feb 1929 -9.8552496 47.35578 -6.200531177
## Mar 1929 -6.8533008 47.42218  0.431118317
## Apr 1929 -2.7634710 47.62278 -0.959311653
## May 1929  3.5013569 47.82338  1.775260197
## Jun 1929  8.9833032 48.19582 -0.279118205
## Jul 1929 12.8452501 48.56825  1.086502728
## Aug 1929 11.4763813 48.90395 -0.080335161
## Sep 1929  7.4475114 49.23966  3.112828016
## Oct 1929  0.4736899 49.35027 -0.623963046
## Nov 1929 -6.4301309 49.46089 -0.130754863
## Dec 1929 -9.4781423 49.44736  1.930783783
## Jan 1930 -9.3471980 49.43383  1.513366755
## Feb 1930 -9.8552496 49.38822 -2.432972451
## Mar 1930 -6.8533008 49.34261 -1.289312084
## Apr 1930 -2.7634710 49.29319  0.370279597
## May 1930  3.5013569 49.24377 -1.545126902
## Jun 1930  8.9833032 49.16499  2.251710214
## Jul 1930 12.8452501 49.08620 -1.831453334
## Aug 1930 11.4763813 48.97296  1.150658078
## Sep 1930  7.4475114 48.85972  0.692770554
## Oct 1930  0.4736899 48.78043  1.645879449
## Nov 1930 -6.4301309 48.70114  0.728987588
## Dec 1930 -9.4781423 48.62085 -0.342708316
## Jan 1931 -9.3471980 48.54056 -2.093359893
## Feb 1931 -9.8552496 48.38319 -0.127942861
## Mar 1931 -6.8533008 48.22583 -2.972526256
## Apr 1931 -2.7634710 48.12257  1.140904226
## May 1931  3.5013569 48.01931  1.979336529
## Jun 1931  8.9833032 48.14283  1.273868208
## Jul 1931 12.8452501 48.26635 -0.511600778
## Aug 1931 11.4763813 48.40974 -1.686116919
## Sep 1931  7.4475114 48.55312 -2.200631996
## Oct 1931  0.4736899 48.54088 -2.414570882
## Nov 1931 -6.4301309 48.52864  3.401489476
## Dec 1931 -9.4781423 48.53430  1.543840385
## Jan 1932 -9.3471980 48.53996  3.207235620
## Feb 1932 -9.8552496 48.66617 -0.410916656
## Mar 1932 -6.8533008 48.79237 -1.639069359
## Apr 1932 -2.7634710 48.83691 -1.473441358
## May 1932  3.5013569 48.88145 -1.482811536
## Jun 1932  8.9833032 48.82173 -0.805030457
```

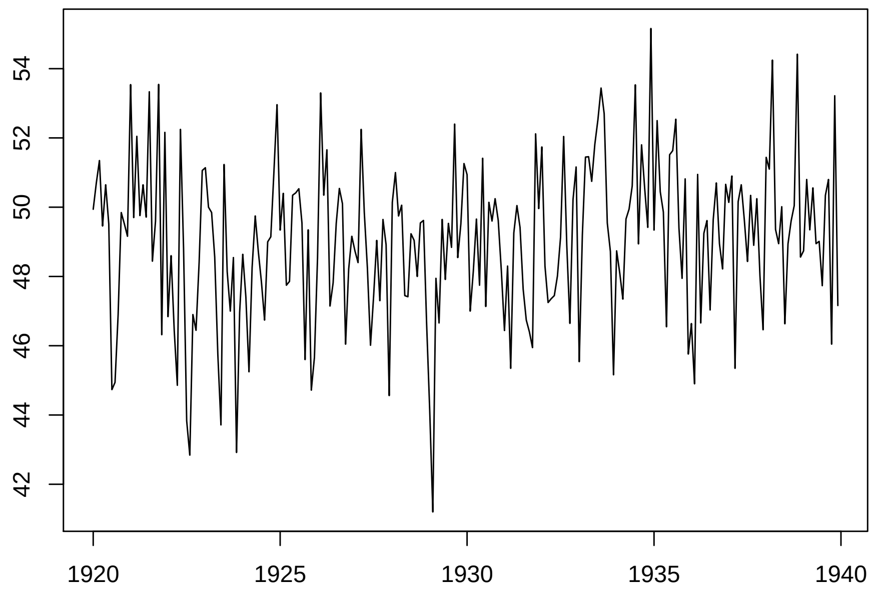
```
## Jul 1932 12.8452501 48.76200  0.492749958
## Aug 1932 11.4763813 48.83146  3.192154743
## Sep 1932  7.4475114 48.90093 -0.048439406
## Oct 1932  0.4736899 49.12176 -2.295447972
## Nov 1932 -6.4301309 49.34259  0.687542706
## Dec 1932 -9.4781423 49.59996  1.678183472
## Jan 1933 -9.3471980 49.85733 -4.310131435
## Feb 1933 -9.8552496 50.12104 -0.965788581
## Mar 1933 -6.8533008 50.38475  0.968553847
## Apr 1933 -2.7634710 50.55788  0.905594559
## May 1933  3.5013569 50.73101 -0.032362909
## Jun 1933  8.9833032 50.69026  1.126437333
## Jul 1933 12.8452501 50.64951  2.005236911
## Aug 1933 11.4763813 50.46236  2.961261164
## Sep 1933  7.4475114 50.27520  2.377286481
## Oct 1933  0.4736899 50.02482 -0.298512893
## Nov 1933 -6.4301309 49.77444 -1.244313023
## Dec 1933 -9.4781423 49.60356 -4.325422241
## Jan 1934 -9.3471980 49.43269 -0.685487132
## Feb 1934 -9.8552496 49.38494 -1.329690994
## Mar 1934 -6.8533008 49.33720 -2.083895283
## Apr 1934 -2.7634710 49.49963  0.163845195
## May 1934  3.5013569 49.66206  0.236587495
## Jun 1934  8.9833032 49.99133  0.625370019
## Jul 1934 12.8452501 50.32060  3.334151879
## Aug 1934 11.4763813 50.58387 -1.660249972
## Sep 1934  7.4475114 50.84714  0.905349242
## Oct 1934  0.4736899 50.88993 -0.163620448
## Nov 1934 -6.4301309 50.93272 -1.702590896
## Dec 1934 -9.4781423 50.87092  4.407220183
## Jan 1935 -9.3471980 50.80912 -1.461924412
## Feb 1935 -9.8552496 50.75825  1.696998728
## Mar 1935 -6.8533008 50.70738 -0.354078560
## Apr 1935 -2.7634710 50.57994 -0.716471023
## May 1935  3.5013569 50.45250 -3.953861666
## Jun 1935  8.9833032 50.14917  1.367526515
## Jul 1935 12.8452501 49.84584  1.908914032
## Aug 1935 11.4763813 49.54007  2.983549209
## Sep 1935  7.4475114 49.23430  0.118185451
## Oct 1935  0.4736899 49.02909 -0.902784869
## Nov 1935 -6.4301309 48.82389  1.806244053
## Dec 1935 -9.4781423 48.64539 -2.767248190
## Jan 1936 -9.3471980 48.46689 -1.819696106
## Feb 1936 -9.8552496 48.35225 -3.496996331
## Mar 1936 -6.8533008 48.23760  2.615703017
## Apr 1936 -2.7634710 48.30761 -1.644138274
```

```
## May 1936  3.5013569 48.37762  0.821022256
## Jun 1936  8.9833032 48.58645  1.030250876
## Jul 1936 12.8452501 48.79527 -1.640521169
## Aug 1936 11.4763813 48.96677  0.656852958
## Sep 1936  7.4475114 49.13826  1.514228150
## Oct 1936  0.4736899 49.21434 -0.088032903
## Nov 1936 -6.4301309 49.29043 -1.260294714
## Dec 1936 -9.4781423 49.33815  1.439989280
## Jan 1937 -9.3471980 49.38588  0.761317600
## Feb 1937 -9.8552496 49.41158  1.443666982
## Mar 1937 -6.8533008 49.43728 -4.183984063
## Apr 1937 -2.7634710 49.39905  0.764425401
## May 1937  3.5013569 49.36081  1.237836686
## Jun 1937  8.9833032 49.32833  0.288370135
## Jul 1937 12.8452501 49.29585 -0.741097080
## Aug 1937 11.4763813 49.39591  0.927705554
## Sep 1937  7.4475114 49.49598 -0.643490748
## Oct 1937  0.4736899 49.62722  0.799089443
## Nov 1937 -6.4301309 49.75846 -1.928331124
## Dec 1937 -9.4781423 49.75732 -3.179182522
## Jan 1938 -9.3471980 49.75619  1.691010406
## Feb 1938 -9.8552496 49.73327  1.321982868
## Mar 1938 -6.8533008 49.71035  4.442954903
## Apr 1938 -2.7634710 49.78815 -0.424682121
## May 1938  3.5013569 49.86596 -0.967317324
## Jun 1938  8.9833032 49.91577  0.100921835
## Jul 1938 12.8452501 49.96559 -3.210839669
## Aug 1938 11.4763813 49.89688 -0.973257797
## Sep 1938  7.4475114 49.82816 -0.275674861
## Oct 1938  0.4736899 49.79656  0.429748533
## Nov 1938 -6.4301309 49.76496  4.465171170
## Dec 1938 -9.4781423 49.79001 -1.111866934
## Jan 1939 -9.3471980 49.81506 -1.067860710
## Feb 1939 -9.8552496 49.78783  0.967421826
## Mar 1939 -6.8533008 49.76060 -0.507296065
## Apr 1939 -2.7634710 49.68448  0.878994770
## May 1939  3.5013569 49.60836 -0.709712574
## Jun 1939  8.9833032 49.51780 -0.501104910
## Jul 1939 12.8452501 49.42725 -1.572497911
## Aug 1939 11.4763813 49.33655  0.987068518
## Sep 1939  7.4475114 49.24585  1.506636011
## Oct 1939  0.4736899 49.16331 -2.936997128
## Nov 1939 -6.4301309 49.08076  3.949368976
## Dec 1939 -9.4781423 49.00510 -1.726954804
```

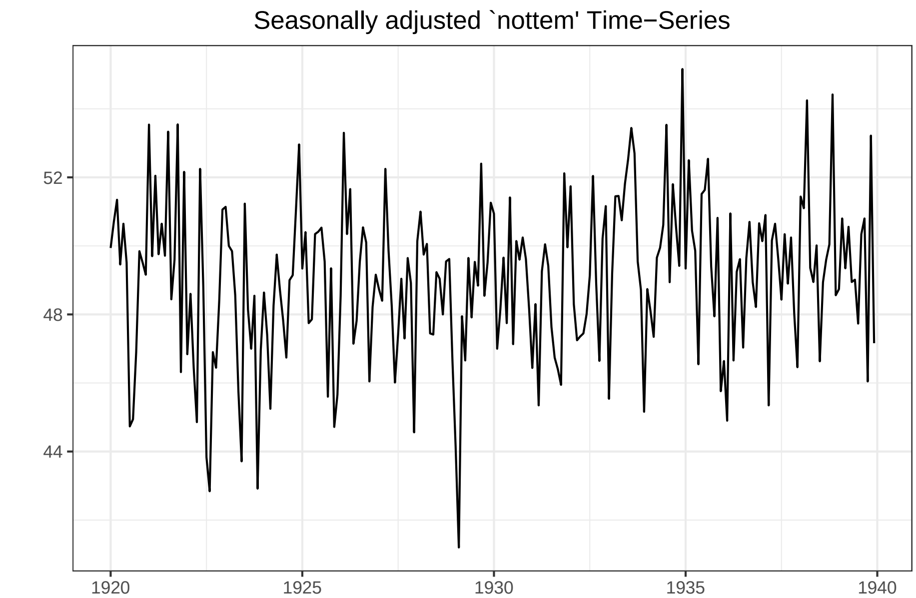
```
# seasonal adjustment
mynottem = decompose(nottem, "additive")
class(mynottem)
```

```
## [1] "decomposed.ts"
```

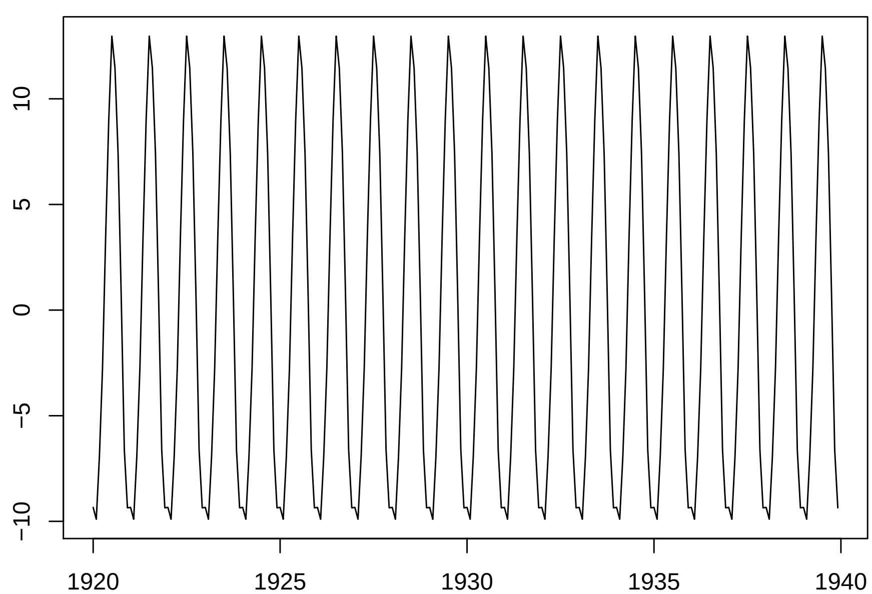
```
# we are subtracting the seasonal element
nottemadjusted = nottem - mynottem$seasonal

# getting a plot
plot(nottemadjusted)
```



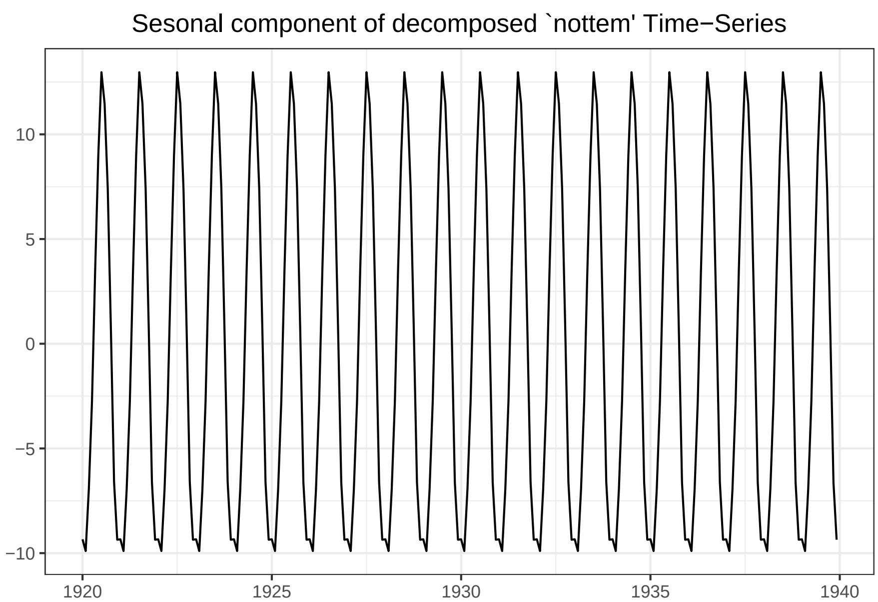

```
autoplot(nottemadjusted) + xlab("") + ylab("") + ggtitle("Seasonally adjusted `nottem' Time-Serie
    theme(plot.title = element_text(hjust = 0.5))
```



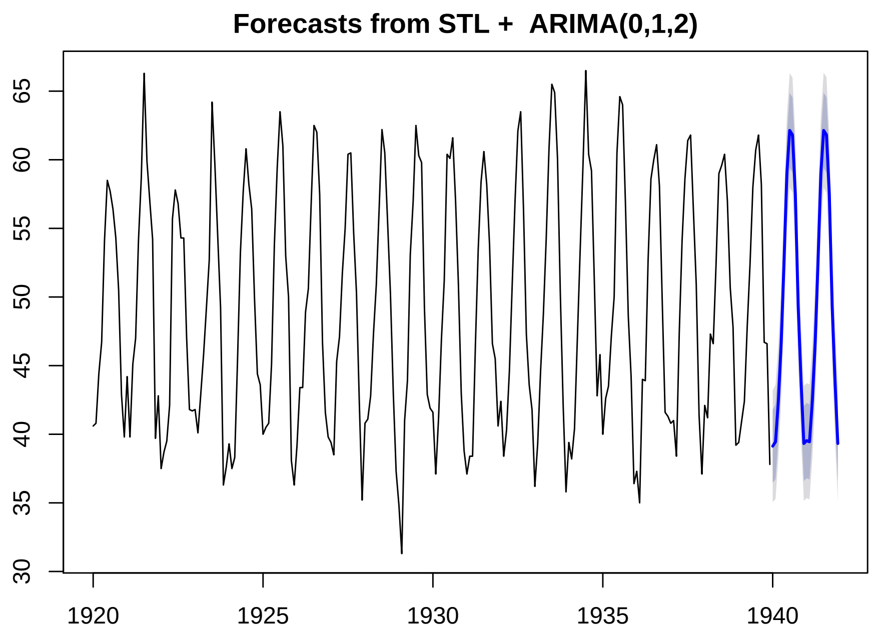

```
plot(mynottem$seasonal)
```



```
autoplot(mynottem$seasonal) + xlab("") + ylab("") + ggtitle("Sesonal component of decomposed `not
    theme(plot.title = element_text(hjust = 0.5))
```



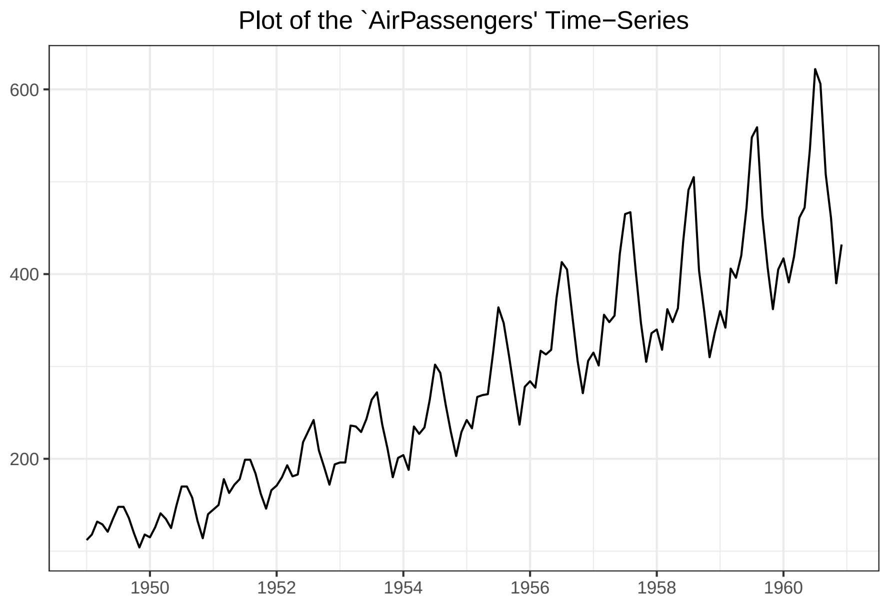

```
# a stl forecast from the package forecast
plot(stlf(nottem, method = "arima"))
```



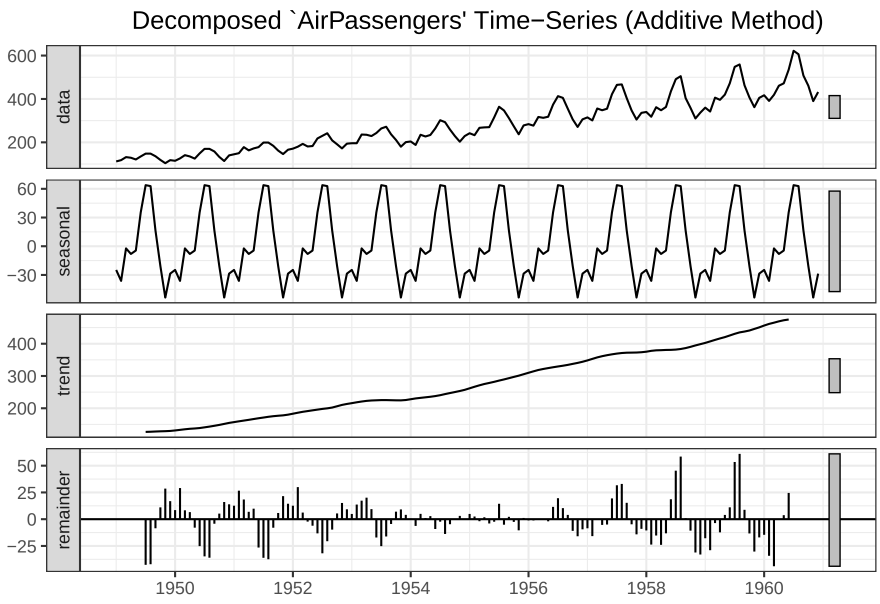

```
autoplot(stlf(nottem, method = "arima")) + xlab("") + ylab("") +
    ggtitle("Forecast of the `nottem' Time-Series using `stlf' function") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
## Error in r[i1] - r[-length(r):-(length(r) - lag + 1L)]: non-numeric argument to bina
```

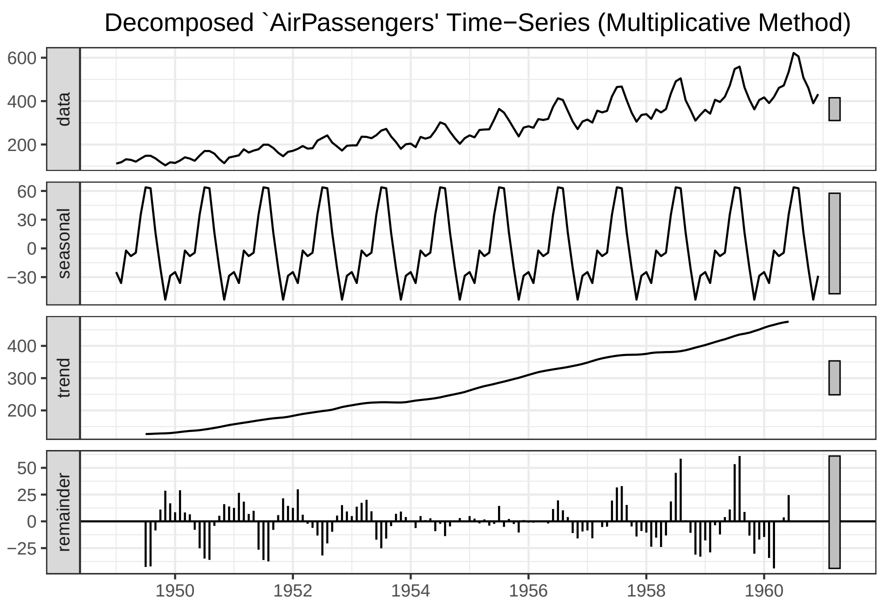
```
### Exercise Decomposition 1. Look for the problems in the Time
### Series data
autoplot(AirPassengers) + xlab("") + ylab("") + ggtitle("Plot of the `AirPassengers' Ti
    theme(plot.title = element_text(hjust = 0.5))  #Evidence of trend and seasonality
```



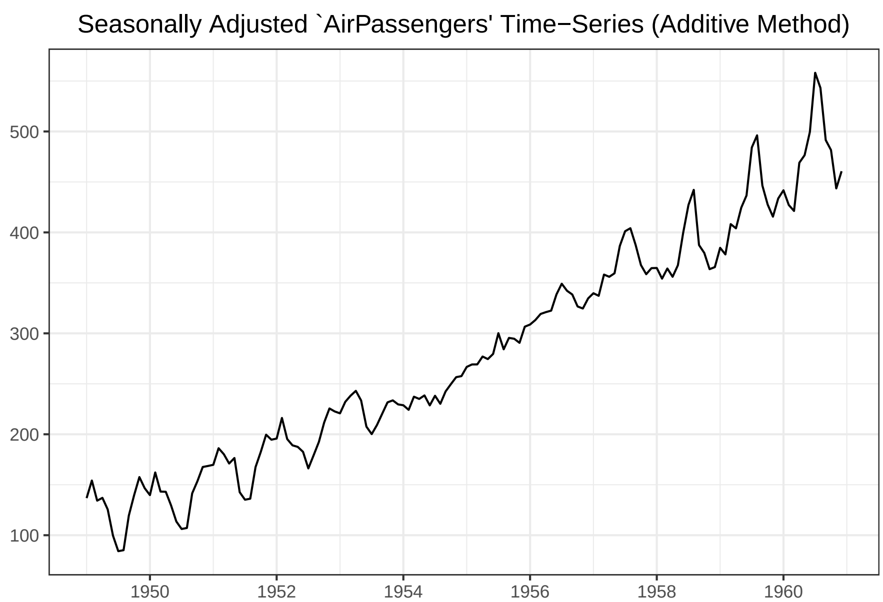

```
# 2. two models for decomposition additive and multiplicative
mymodel1 <- decompose(AirPassengers, type = "additive")
mymodel2 <- decompose(AirPassengers, type = "multiplicative")

# 3. Plot and compare the two decomposition models
autoplot(mymodel1) + xlab("") + ylab("") + ggtitle("Decomposed `AirPassengers' Time-Series (Addit
    theme(plot.title = element_text(hjust = 0.5))
```



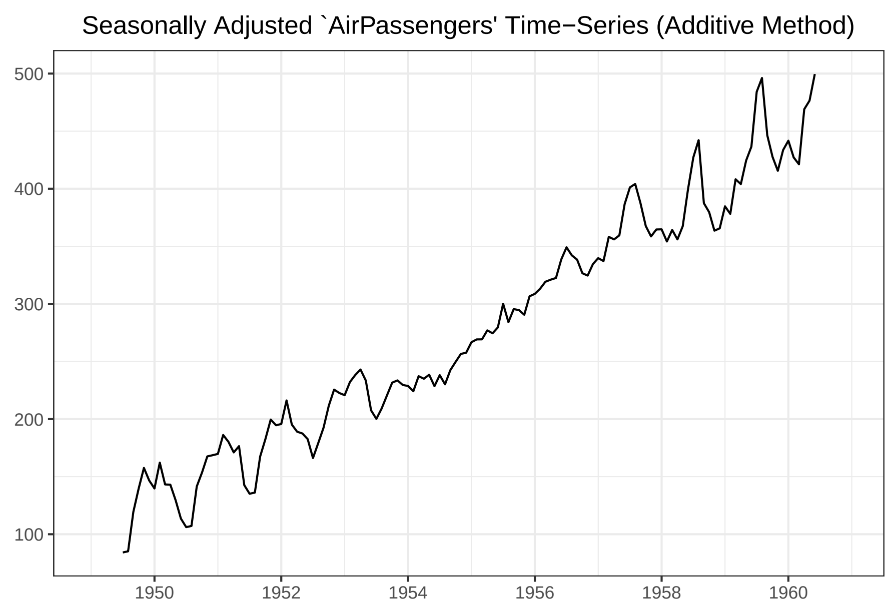

```r
autoplot(mymodel1) + xlab("") + ylab("") + ggtitle("Decomposed `AirPassengers' Time-Se
    theme(plot.title = element_text(hjust = 0.5))
```



```r
# Both the decompsition show evidence of the presence of
# trend and seasonality

# 4. plot seasonally adjusted time series for mymodel1
autoplot(AirPassengers - (mymodel1$seasonal)) + xlab("") + ylab("") +
    ggtitle("Seasonally Adjusted `AirPassengers' Time-Series (Additive Method)") +
    theme(plot.title = element_text(hjust = 0.5))
```




```r
autoplot(mymodel1$trend + mymodel1$random) + xlab("") + ylab("") +
    ggtitle("Seasonally Adjusted `AirPassengers' Time-Series (Additive Method)") +
    theme(plot.title = element_text(hjust = 0.5))
```




```
### SMOOTHING SMA in order to identify trends, we can use
### smoothers like a simple moving avg n identfies the order or
### the SMA - you can experiment with this parameter

x = c(1, 2, 3, 4, 5, 6, 7)
SMA(x, n = 3)  # SMA fro TTR package, 3rd order
```

```
## Error in SMA(x, n = 3): could not find function "SMA"
```

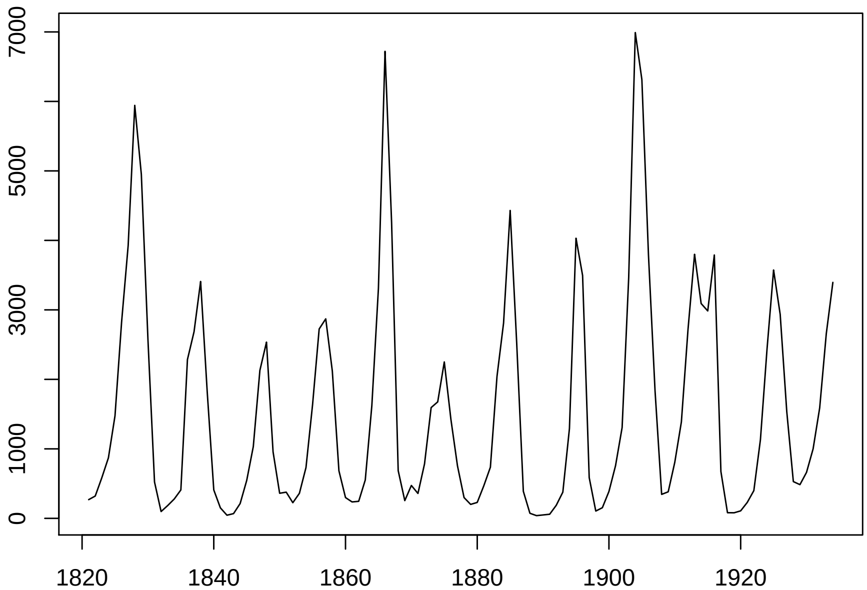
```
lynxsmoothed = SMA(lynx, n = 9)
```

```
## Error in SMA(lynx, n = 9): could not find function "SMA"
```

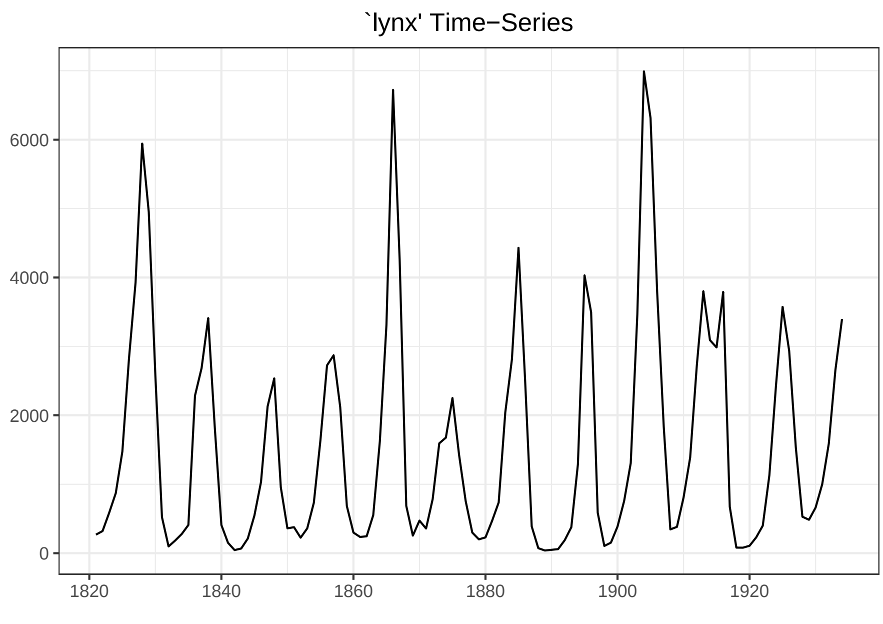
```
lynxsmoothed
```

```
## Error in eval(expr, envir, enclos): object 'lynxsmoothed' not found
```

```
# we can compare the smoothed vs the original lynx data
plot(lynx)
```




```r
autoplot(lynx) + xlab("") + ylab("") + ggtitle("`lynx' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5))
```



```
plot(lynxsmoothed)
```

```
## Error in plot(lynxsmoothed): object 'lynxsmoothed' not found
```

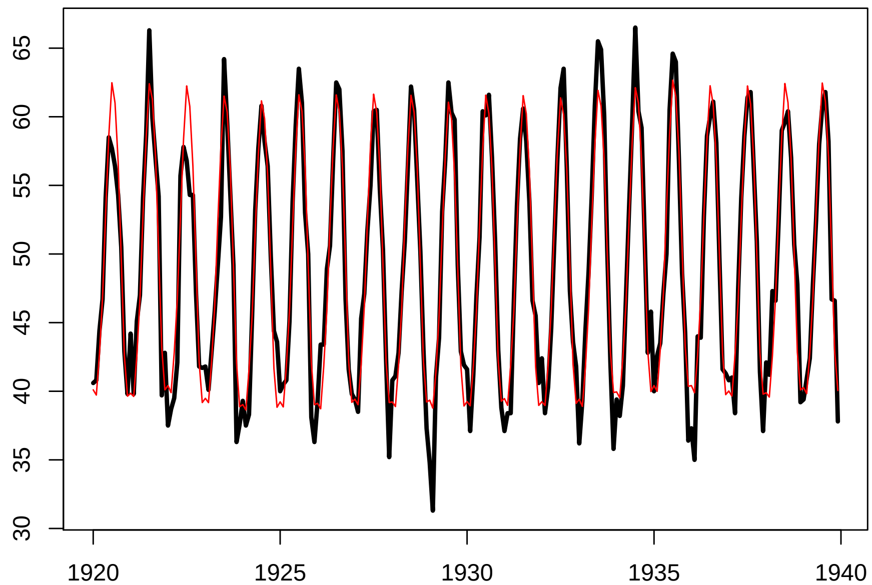
```
autoplot(lynxsmoothed) + xlab("") + ylab("") + ggtitle("Smoother `lynx' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
## Error in autoplot(lynxsmoothed): object 'lynxsmoothed' not found
```

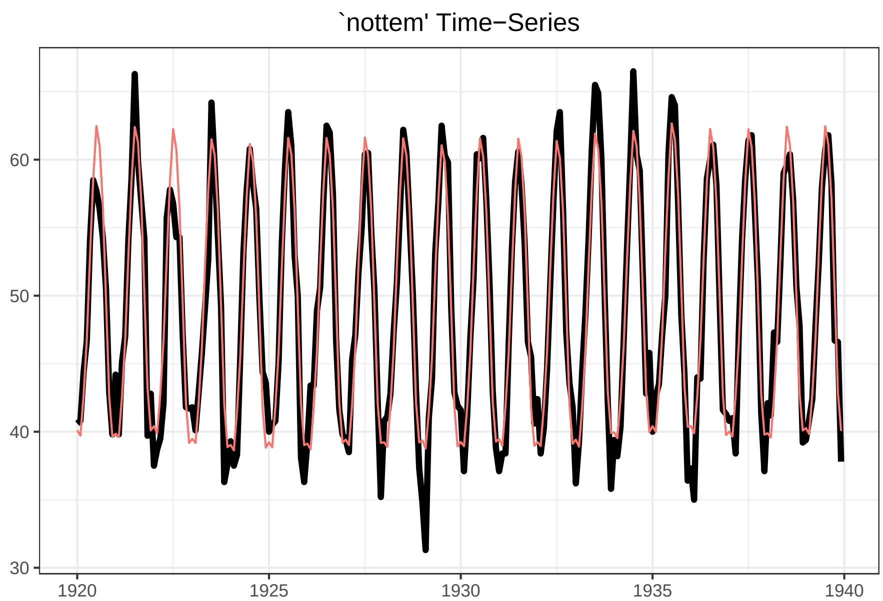
```
# Exponential Smoothing with ets ets Using function ets
etsmodel = ets(nottem)
etsmodel
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = nottem)
##
##   Smoothing parameters:
##     alpha = 0.0392
##     gamma = 1e-04
##
##   Initial states:
##     l = 49.4597
##     s = -9.5635 -6.6186 0.5447 7.4811 11.5783 12.8567
##            8.9762 3.4198 -2.7516 -6.8093 -9.7583 -9.3556
##
##   sigma:  2.3203
##
##      AIC     AICc      BIC
## 1734.944 1737.087 1787.154
```

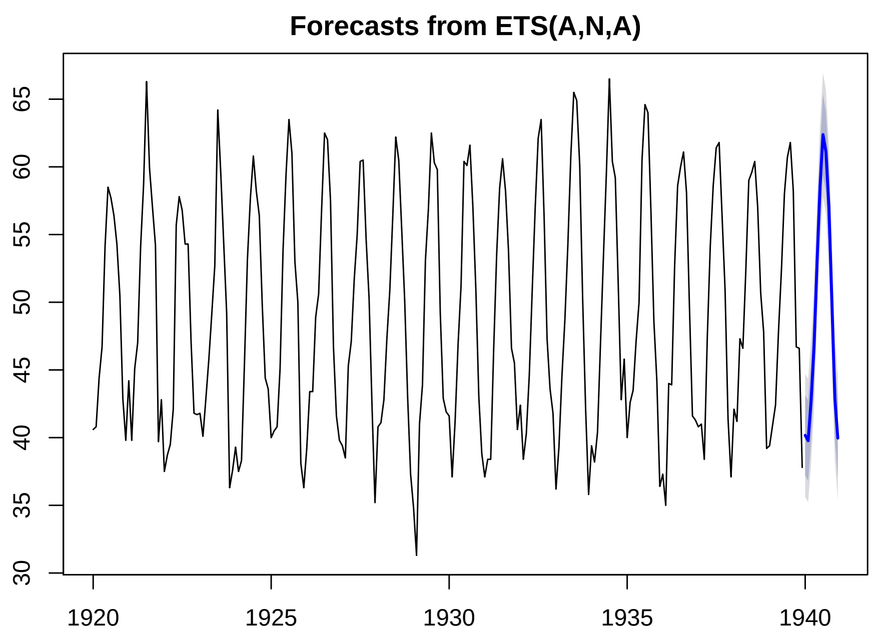
```
# Plotting the model vs original
plot(nottem, lwd = 3)
lines(etsmodel$fitted, col = "red")
```



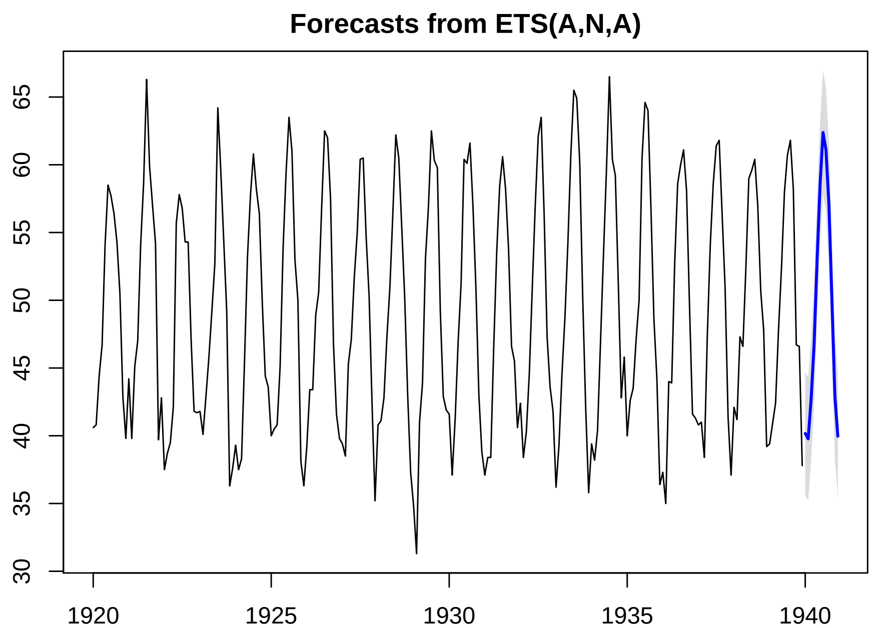

```
autoplot(nottem, size = 1.5) + xlab("") + ylab("") + ggtitle("`nottem' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5), legend.position = "none") +
    autolayer(etsmodel$fitted)
```



```r
# Plotting the forecast
plot(forecast(etsmodel, h = 12))
```

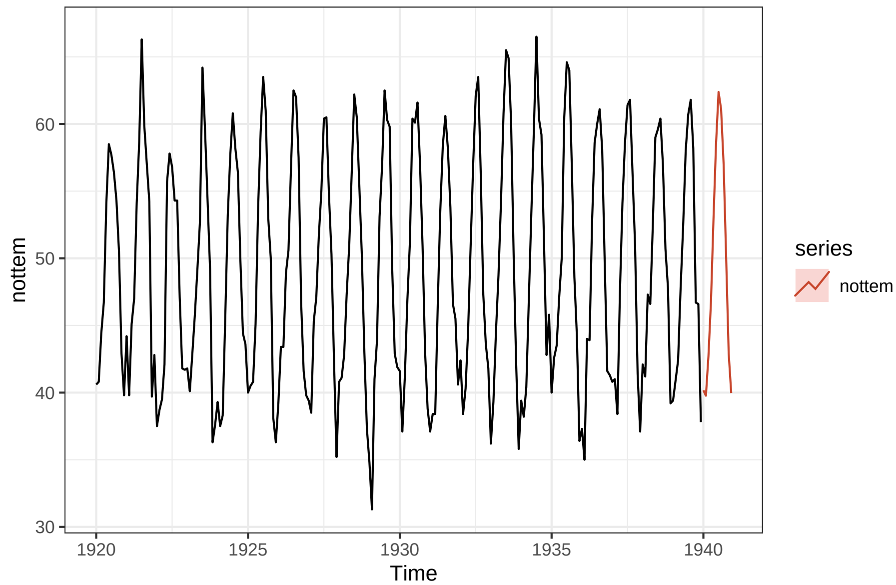



```r
# Changing the prediction interval
plot(forecast(etsmodel, h = 12, level = 95))
```




```r
autoplot(nottem, PI = TRUE, shadecols = c("#596DD5", "#D5DBFF"),
    fcol = "#0000AA", flwd = 0.5) + autolayer(forecast(etsmodel,
    h = 12), level = 95, series = "nottem")
```

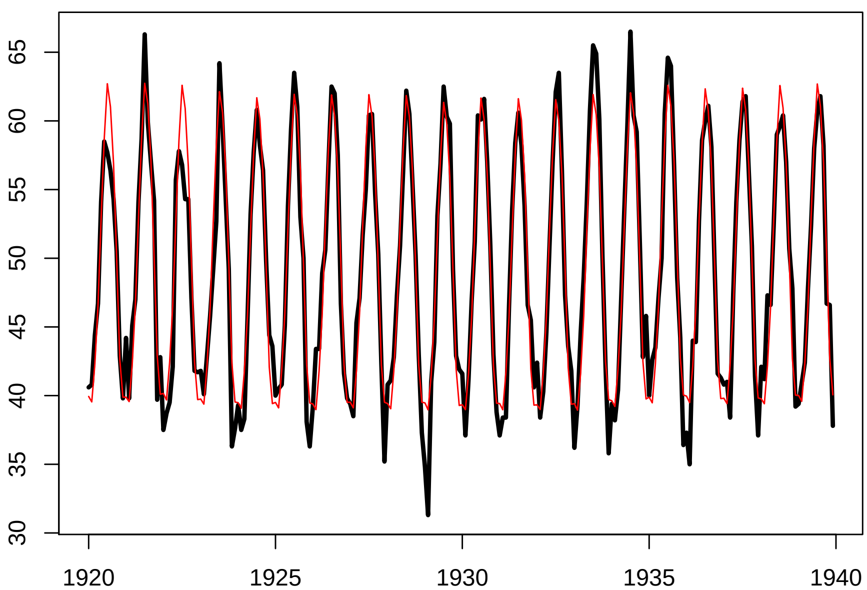
```
## Warning: Ignoring unknown parameters: PI, shadecols, fcol, flwd
```



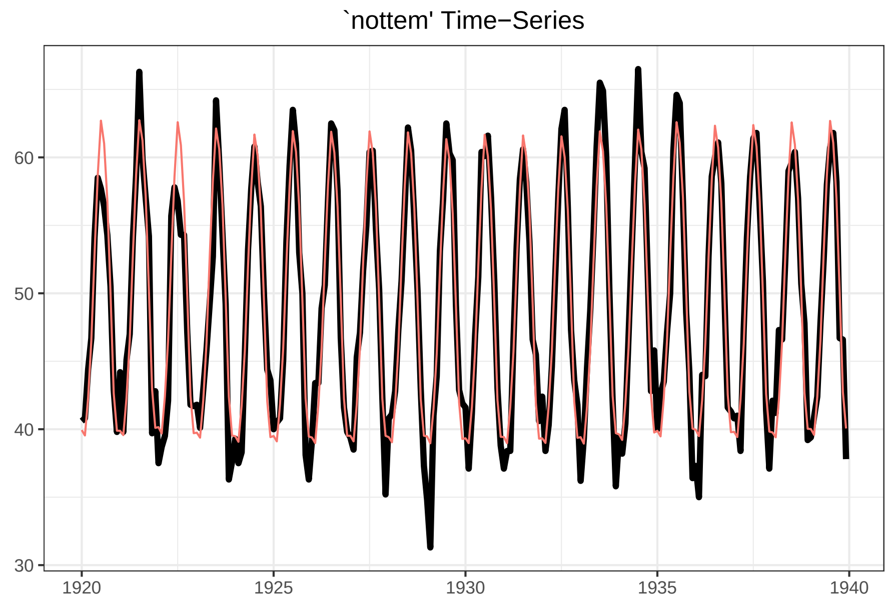

```
# Manually setting the ets model
etsmodmult = ets(nottem, model = "MZM")
etsmodmult
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = nottem, model = "MZM")
##
##   Smoothing parameters:
##     alpha = 0.0214
##     gamma = 1e-04
##
##   Initial states:
##     l = 49.3793
##     s = 0.8089 0.8647 1.0132 1.1523 1.2348 1.2666
##            1.1852 1.0684 0.9405 0.8561 0.8005 0.8088
##
##   sigma:  0.0508
##
##      AIC     AICc      BIC
## 1761.911 1764.054 1814.121
```

```r
# Plot as comparison
plot(nottem, lwd = 3)
lines(etsmodmult$fitted, col = "red")
```




```r
autoplot(nottem, size = 1.5) + xlab("") + ylab("") + ggtitle("`nottem' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5), legend.position = "none") +
    autolayer(etsmodmult$fitted)
```

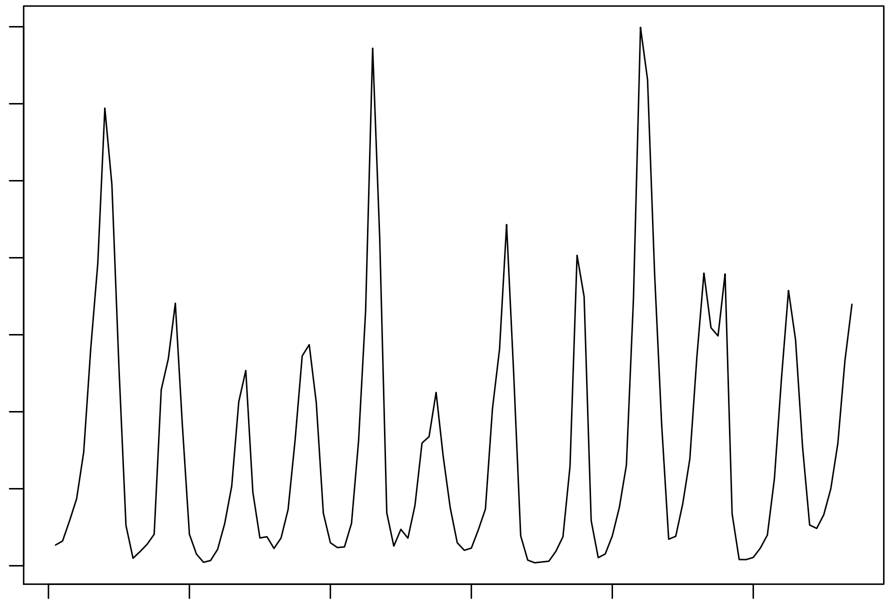

`nottem' Time−Series
60
50
40
30
1920
1925
1930
1935
1940

Chapter 6

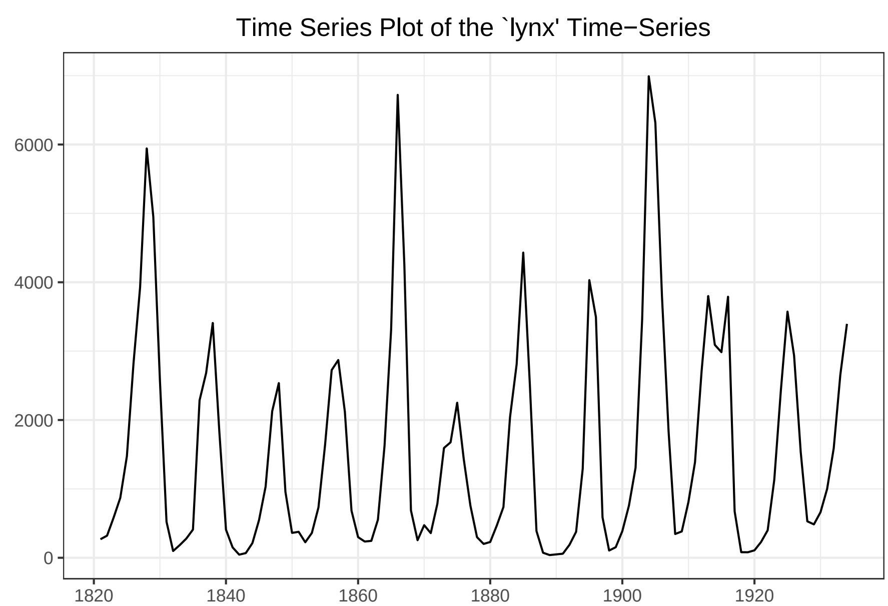
ARIMA Models

```r
# preamble setting the directories and loading packages
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section6")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
require(lmtest)
require(TTR)  #for smoothing the time series
```

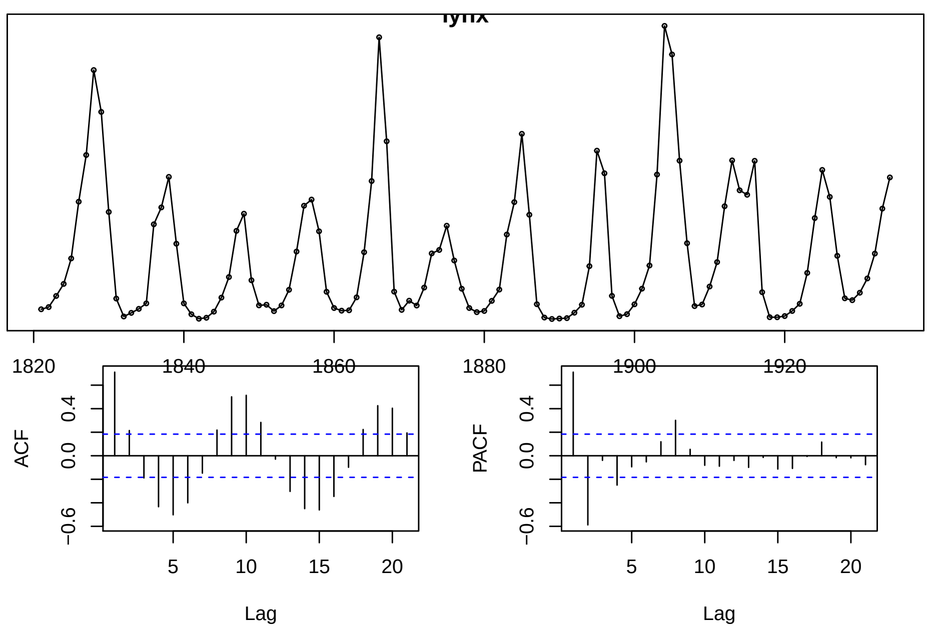
```r
#globally setting the theme for plots
theme_set(theme_bw())
par(mar = c(1,1,1,1))

#Script
### ARIMA models
## ARIMA with auto.arima

plot(lynx)
```



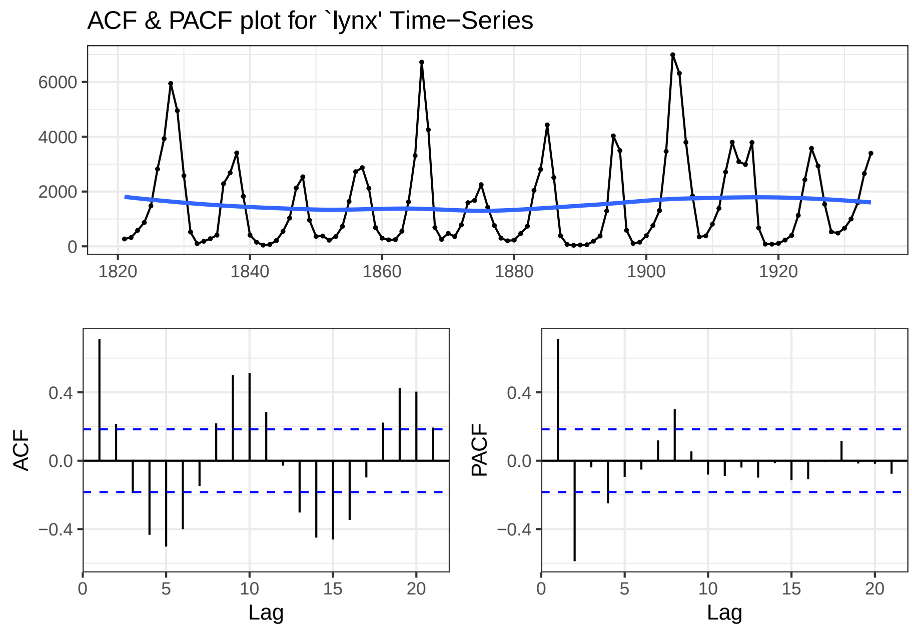
```r
autoplot(lynx) +
  xlab("") + ylab("") +
  ggtitle("Time Series Plot of the `lynx' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5)) #for centering the text
```



```r
par(mar = c(0.01,0.01,0.01,0.01))
tsdisplay(lynx) # autoregression?
```




```r
ggtsdisplay(lynx,
            plot.type = "partial",
            main = "ACF & PACF plot for `lynx' Time-Series",
            smooth = TRUE)
```




```
adf.test(lynx) #to check for the stationarity of the time series
```

```
## Warning in adf.test(lynx): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lynx
## Dickey-Fuller = -6.3068, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
auto.arima(lynx)
```

```
## Series: lynx
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2       mean
##       1.3421  -0.6738  -0.2027  -0.2564  1544.4039
## s.e.  0.0984   0.0801   0.1261   0.1097   131.9242
##
## sigma^2 estimated as 761965:  log likelihood=-932.08
## AIC=1876.17   AICc=1876.95   BIC=1892.58
```

```r
auto.arima(lynx, trace = T)
```

```
##
##  ARIMA(2,0,2) with non-zero mean : 1876.952
##  ARIMA(0,0,0) with non-zero mean : 2006.724
##  ARIMA(1,0,0) with non-zero mean : 1927.209
##  ARIMA(0,0,1) with non-zero mean : 1918.165
##  ARIMA(0,0,0) with zero mean     : 2080.721
##  ARIMA(1,0,2) with non-zero mean : 1888.757
##  ARIMA(2,0,1) with non-zero mean : 1880.014
##  ARIMA(3,0,2) with non-zero mean : 1878.603
##  ARIMA(2,0,3) with non-zero mean : Inf
##  ARIMA(1,0,1) with non-zero mean : 1891.442
##  ARIMA(1,0,3) with non-zero mean : 1890.03
##  ARIMA(3,0,1) with non-zero mean : 1881.962
##  ARIMA(3,0,3) with non-zero mean : 1881.515
##  ARIMA(2,0,2) with zero mean     : 1905.595
##
##  Best model: ARIMA(2,0,2) with non-zero mean
```

```
## Series: lynx
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2       mean
##       1.3421  -0.6738  -0.2027  -0.2564  1544.4039
## s.e.  0.0984   0.0801   0.1261   0.1097   131.9242
##
## sigma^2 estimated as 761965:  log likelihood=-932.08
## AIC=1876.17   AICc=1876.95   BIC=1892.58
```

```r
# recommended setting
auto.arima(lynx, trace = T,
           stepwise = F,
           approximation = F)
```

```
##
##  ARIMA(0,0,0) with zero mean     : 2080.721
##  ARIMA(0,0,0) with non-zero mean : 2006.724
##  ARIMA(0,0,1) with zero mean     : 1972.791
##  ARIMA(0,0,1) with non-zero mean : 1918.165
##  ARIMA(0,0,2) with zero mean     : 1925.15
##  ARIMA(0,0,2) with non-zero mean : 1890.428
##  ARIMA(0,0,3) with zero mean     : 1913.118
```

```
##  ARIMA(0,0,3) with non-zero mean : 1888.326
##  ARIMA(0,0,4) with zero mean     : 1906.524
##  ARIMA(0,0,4) with non-zero mean : 1889.064
##  ARIMA(0,0,5) with zero mean     : 1908.619
##  ARIMA(0,0,5) with non-zero mean : 1886.754
##  ARIMA(1,0,0) with zero mean     : 1934.647
##  ARIMA(1,0,0) with non-zero mean : 1927.209
##  ARIMA(1,0,1) with zero mean     : 1903.345
##  ARIMA(1,0,1) with non-zero mean : 1891.442
##  ARIMA(1,0,2) with zero mean     : 1903.567
##  ARIMA(1,0,2) with non-zero mean : 1888.757
##  ARIMA(1,0,3) with zero mean     : 1905.59
##  ARIMA(1,0,3) with non-zero mean : 1890.03
##  ARIMA(1,0,4) with zero mean     : 1907.578
##  ARIMA(1,0,4) with non-zero mean : Inf
##  ARIMA(2,0,0) with zero mean     : 1906.685
##  ARIMA(2,0,0) with non-zero mean : 1878.399
##  ARIMA(2,0,1) with zero mean     : 1903.412
##  ARIMA(2,0,1) with non-zero mean : 1880.014
##  ARIMA(2,0,2) with zero mean     : 1905.595
##  ARIMA(2,0,2) with non-zero mean : 1876.952
##  ARIMA(2,0,3) with zero mean     : 1907.963
##  ARIMA(2,0,3) with non-zero mean : Inf
##  ARIMA(3,0,0) with zero mean     : 1903.728
##  ARIMA(3,0,0) with non-zero mean : 1880.512
##  ARIMA(3,0,1) with zero mean     : 1905.587
##  ARIMA(3,0,1) with non-zero mean : 1881.962
##  ARIMA(3,0,2) with zero mean     : Inf
##  ARIMA(3,0,2) with non-zero mean : 1878.603
##  ARIMA(4,0,0) with zero mean     : 1905.899
##  ARIMA(4,0,0) with non-zero mean : 1875.007
##  ARIMA(4,0,1) with zero mean     : Inf
##  ARIMA(4,0,1) with non-zero mean : 1876.407
##  ARIMA(5,0,0) with zero mean     : 1904.543
##  ARIMA(5,0,0) with non-zero mean : 1876.332
##
##
##
##  Best model: ARIMA(4,0,0) with non-zero mean
```

```
## Series: lynx
## ARIMA(4,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4       mean
##       1.1246  -0.7174  0.2634  -0.2543  1547.3859
```

```
## s.e.  0.0903   0.1367  0.1361   0.0897   136.8501
##
## sigma^2 estimated as 748457:  log likelihood=-931.11
## AIC=1874.22   AICc=1875.01   BIC=1890.64
```

```
## ARIMA calculations
# AR(2) model
myarima = arima(lynx, order = c(2,0,0))
myarima
```

```
##
## Call:
## arima(x = lynx, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.1474  -0.5997  1545.4458
## s.e.  0.0742   0.0740   181.6736
##
## sigma^2 estimated as 768159:  log likelihood = -935.02,  aic = 1878.03
```

```
tail(lynx)
```

```
## Time Series:
## Start = 1929
## End = 1934
## Frequency = 1
## [1]  485  662 1000 1590 2657 3396
```

```
residuals(myarima)
```

```
## Time Series:
## Start = 1821
## End = 1934
## Frequency = 1
##    [1]  -711.715800  -247.179068  -321.014839  -306.751202   127.414827
##    [6]   951.890591   876.687792  2428.733153  -212.432514  -237.541926
##   [11]  -164.223204   344.415030  -313.801319  -572.372533  -499.800869
##   [16]  1284.008241  -390.614888   999.532714 -1176.312892  -338.411239
##   [21]    76.614594  -581.986383  -592.092428  -537.056449  -356.640535
##   [26]  -164.773680   572.140125    13.626146 -1375.059569    84.838236
##   [31]  -162.287575  -690.094698  -371.088497  -246.153634   316.113199
##   [36]   584.894187    27.600121  -240.002495  -724.567794    85.994521
```

```
##  [41]  -395.876984  -545.490420  -286.601293   437.533551  1080.751334
##  [46]  3196.206424 -2171.180547  -862.324669  1319.008240  -106.590562
##  [51]  -730.821550   -42.121950   210.099599  -381.832131   584.871735
##  [56]  -850.724879  -229.236651  -412.244306  -387.695328  -521.330158
##  [61]  -372.233398  -363.825181   779.748247   210.328753  1731.217687
##  [66] -1586.424626  -533.760190   433.588275  -510.481277  -650.987938
##  [71]  -672.853636  -549.330544  -502.352341   273.149380  2075.597251
##  [76] -1054.463188 -1704.735336   828.545228  -314.449678  -424.603800
##  [81]  -293.316062   -29.674453  1720.888636  3099.981788  -329.627515
##  [86]    44.035737   569.799862  -185.278565   388.247443  -122.427802
##  [91]    -9.044981   905.933577   820.433050  -341.167517  1018.287679
##  [96]  1519.696544 -2583.562459   881.643131  -307.733379  -634.234854
## [101]  -545.962808  -498.009703   112.495341   673.381411   763.327803
## [106]  -406.375377  -386.254717  -173.376326   100.795394  -276.260594
## [111]  -167.745563   140.575959   733.302579   601.838001
```

```
# Check the equation for AR(2)
(2657 - 1545.45)*1.147 - (1590 - 1545.45)*0.6 + 601.84
```

```
## [1] 1850.058
```

```
3396 - 1545.45
```

```
## [1] 1850.55
```

```
# MA(2) model
myarima = arima(lynx, order = c(0,0,2))
myarima
```

```
##
## Call:
## arima(x = lynx, order = c(0, 0, 2))
##
## Coefficients:
##          ma1     ma2  intercept
##       1.1407  0.4697  1545.3670
## s.e.  0.0776  0.0721   224.5215
##
## sigma^2 estimated as 855092:  log likelihood = -941.03,  aic = 1890.06
```

```
residuals(myarima)
```

```
## Time Series:
```

```
## Start = 1821
## End = 1934
## Frequency = 1
##   [1]  -803.732851  -316.819775  -339.796973  -153.575542   256.164758
##   [6]  1051.017490  1062.665677  2690.592373  -162.936784   -44.605977
##  [11]  -894.921151  -405.552321  -478.418368  -530.135762  -306.914693
##  [16]  1338.739662  -243.365541  1512.454318 -1332.377780  -326.856600
##  [21]  -395.701695  -895.452231  -270.030612  -603.745610  -183.818830
##  [26]   -19.103090   691.762826   210.483679 -1153.389638    32.488716
##  [31]  -663.690549  -578.528068  -213.686644  -298.876138   533.939929
##  [36]   710.925757   263.863961   -61.283359  -915.393384  -173.356483
##  [41]  -681.659497  -441.346353  -169.735507   478.553892  1299.450551
##  [46]  3468.524287 -1858.394332  -367.558957     1.794961  -901.775190
##  [51]  -159.518680  -155.841195   301.331932  -139.911234   723.702215
##  [56]  -879.208277  -126.335608  -689.293961  -498.722732  -423.698105
##  [61]  -358.791482  -201.071547   894.524832   339.653953  2078.024947
##  [66] -1564.386859  -347.838999  -340.793301  -954.233203  -247.766795
##  [71]  -755.533899  -379.124919  -381.015812   359.344984  2254.673608
##  [76]  -791.145850 -1114.876734   203.012693 -1100.303344     1.440094
##  [81]  -272.206328    71.473327  1965.953529  3169.419791   228.755266
##  [86]   499.031993  -386.077507  -994.344061   152.258905  -444.019657
##  [91]   277.629380  1059.482295   915.638387     3.497027  1005.575888
##  [96]  1095.889333 -2593.803120   979.758850 -1364.729473  -340.749646
## [101]  -286.657856  -659.317506   473.383962   656.300763  1057.619544
## [106]  -125.095552  -362.420744  -544.182680  -269.369783  -320.487877
## [111]   -53.252771   255.911083   844.717221   766.830502
```

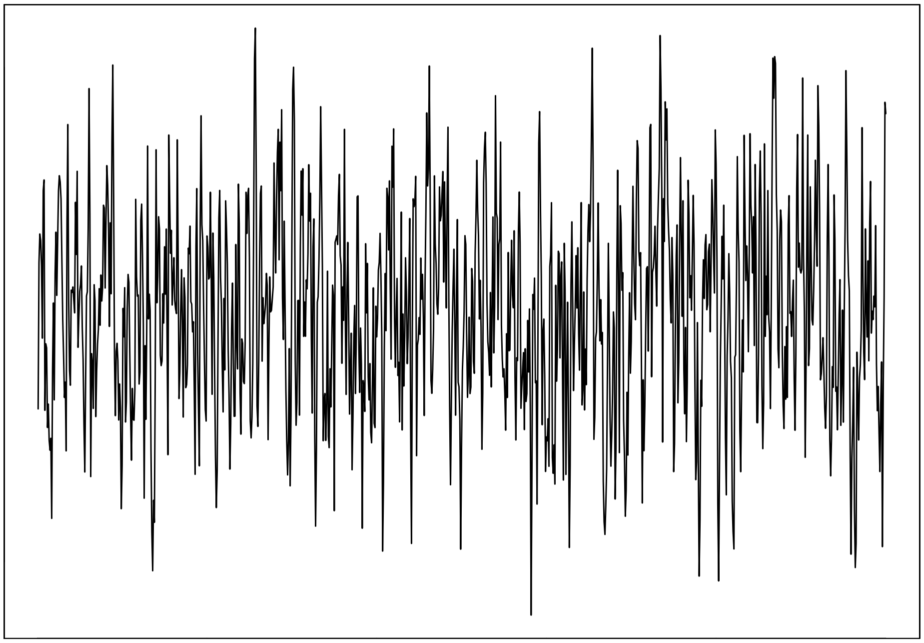
```r
# Check the equation for MA(2)
844.72*1.141 + 255.91*0.47 + 766.83
```

```
## [1] 1850.933
```

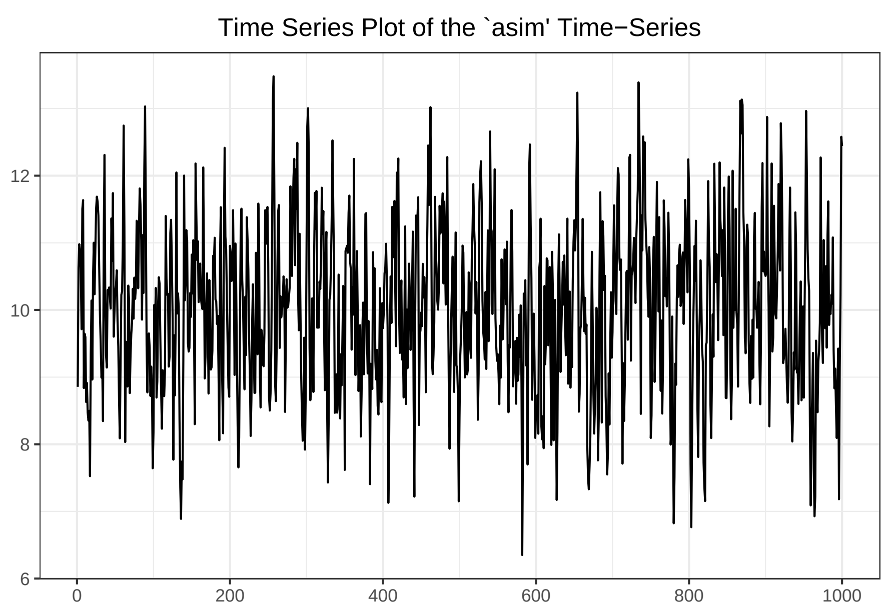
```r
3396 - 1545.37
```

```
## [1] 1850.63
```

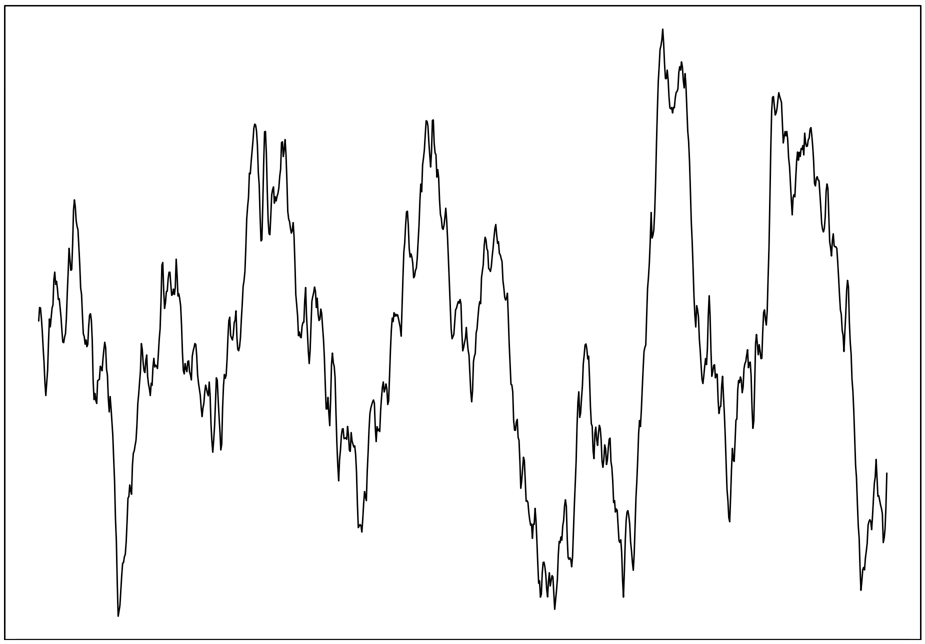
```r
## ARIMA time series simulations
set.seed(123) # for reproduction
# simulation, at least n of 1000
asim <- arima.sim(model = list(order = c(1,0,1),
                              ar = c(0.4),
                              ma = c(0.3)),
                  n = 1000) + 10
plot(asim)
```



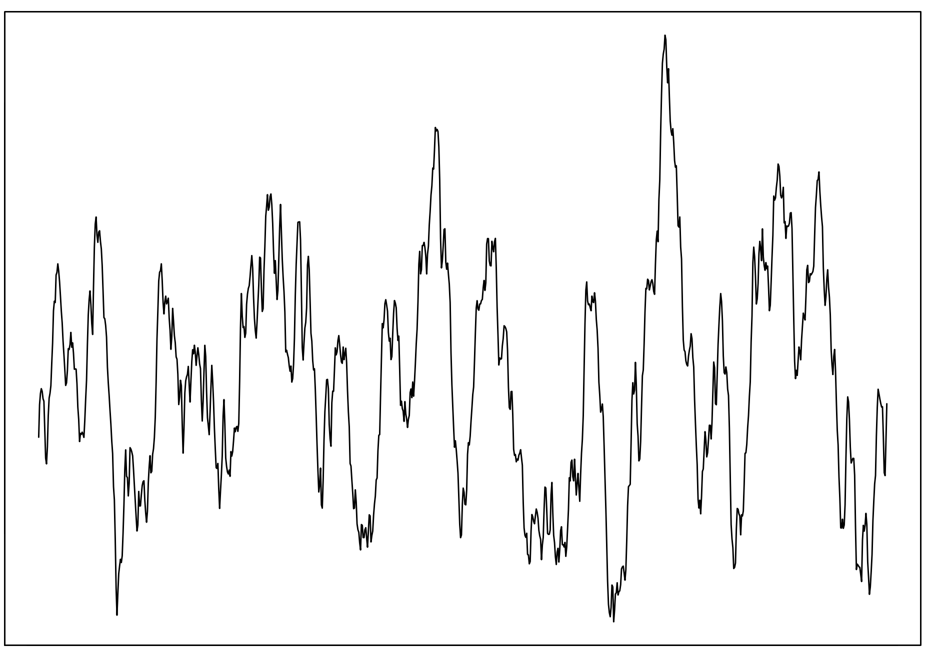
```r
autoplot(asim) +
  xlab("") + ylab("") +
  ggtitle("Time Series Plot of the `asim' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5)) #for centering the text
```



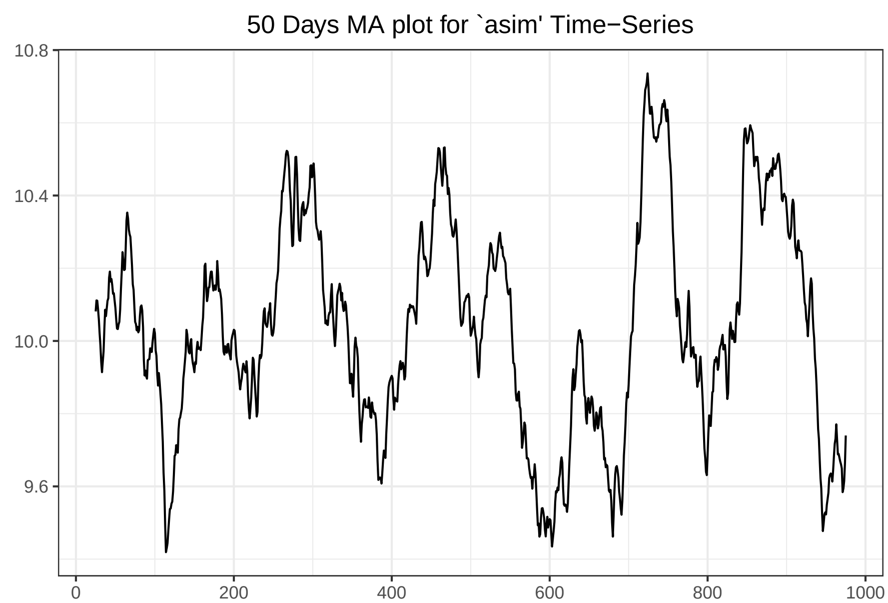
```r
plot(rollmean(asim, 50)) #50 days MA
```



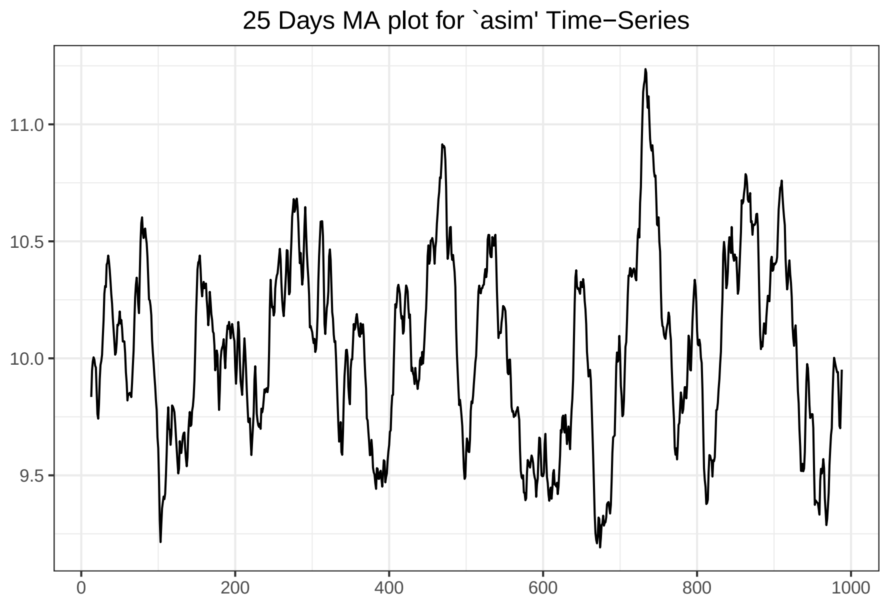
```r
plot(rollmean(asim, 25)) #25 days MA
```



```
autoplot(rollmean(asim, 50)) + #50 days MA
  xlab("") + ylab("") +
  ggtitle("50 Days MA plot for `asim' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
autoplot(rollmean(asim, 25)) + #25 days MA
xlab("") + ylab("") +
  ggtitle("25 Days MA plot for `asim' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5))
```

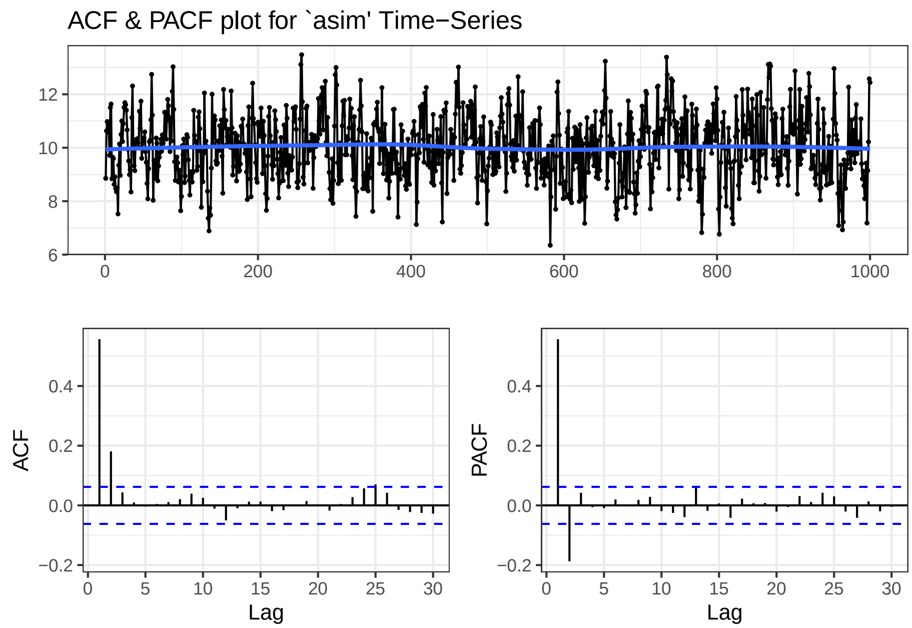



```r
# Stationarity library(tseries)
adf.test(asim)
```

```
## Warning in adf.test(asim): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  asim
## Dickey-Fuller = -9.0113, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# Autocorrelation library(forecast) tsdisplay(asim)
ggtsdisplay(asim,
            plot.type = "partial",
            main = "ACF & PACF plot for `asim' Time-Series",
            smooth = TRUE)
```




```r
auto.arima(asim,
           trace = T,
           stepwise = F,
           approximation = F)
```

```
##
##  ARIMA(0,0,0) with zero mean     : 7465.459
##  ARIMA(0,0,0) with non-zero mean : 3241.528
##  ARIMA(0,0,1) with zero mean     : 6218.948
##  ARIMA(0,0,1) with non-zero mean : 2878.74
##  ARIMA(0,0,2) with zero mean     : 5341.968
##  ARIMA(0,0,2) with non-zero mean : 2836.895
##  ARIMA(0,0,3) with zero mean     : 4809.724
##  ARIMA(0,0,3) with non-zero mean : 2837.534
##  ARIMA(0,0,4) with zero mean     : 4450.32
##  ARIMA(0,0,4) with non-zero mean : 2838.689
##  ARIMA(0,0,5) with zero mean     : 4219.275
##  ARIMA(0,0,5) with non-zero mean : 2840.557
##  ARIMA(1,0,0) with zero mean     : Inf
##  ARIMA(1,0,0) with non-zero mean : 2870.637
##  ARIMA(1,0,1) with zero mean     : Inf
##  ARIMA(1,0,1) with non-zero mean : 2836.047
##  ARIMA(1,0,2) with zero mean     : Inf
##  ARIMA(1,0,2) with non-zero mean : 2837.165
```

```
##  ARIMA(1,0,3) with zero mean     : Inf
##  ARIMA(1,0,3) with non-zero mean : 2839.088
##  ARIMA(1,0,4) with zero mean     : Inf
##  ARIMA(1,0,4) with non-zero mean : 2840.615
##  ARIMA(2,0,0) with zero mean     : Inf
##  ARIMA(2,0,0) with non-zero mean : 2836.945
##  ARIMA(2,0,1) with zero mean     : Inf
##  ARIMA(2,0,1) with non-zero mean : 2837.319
##  ARIMA(2,0,2) with zero mean     : Inf
##  ARIMA(2,0,2) with non-zero mean : 2838.849
##  ARIMA(2,0,3) with zero mean     : Inf
##  ARIMA(2,0,3) with non-zero mean : 2840.867
##  ARIMA(3,0,0) with zero mean     : Inf
##  ARIMA(3,0,0) with non-zero mean : 2837.297
##  ARIMA(3,0,1) with zero mean     : Inf
##  ARIMA(3,0,1) with non-zero mean : 2839.296
##  ARIMA(3,0,2) with zero mean     : Inf
##  ARIMA(3,0,2) with non-zero mean : 2840.86
##  ARIMA(4,0,0) with zero mean     : Inf
##  ARIMA(4,0,0) with non-zero mean : 2839.279
##  ARIMA(4,0,1) with zero mean     : Inf
##  ARIMA(4,0,1) with non-zero mean : 2841.309
##  ARIMA(5,0,0) with zero mean     : Inf
##  ARIMA(5,0,0) with non-zero mean : 2841.162
##
##
##
##  Best model: ARIMA(1,0,1) with non-zero mean
```

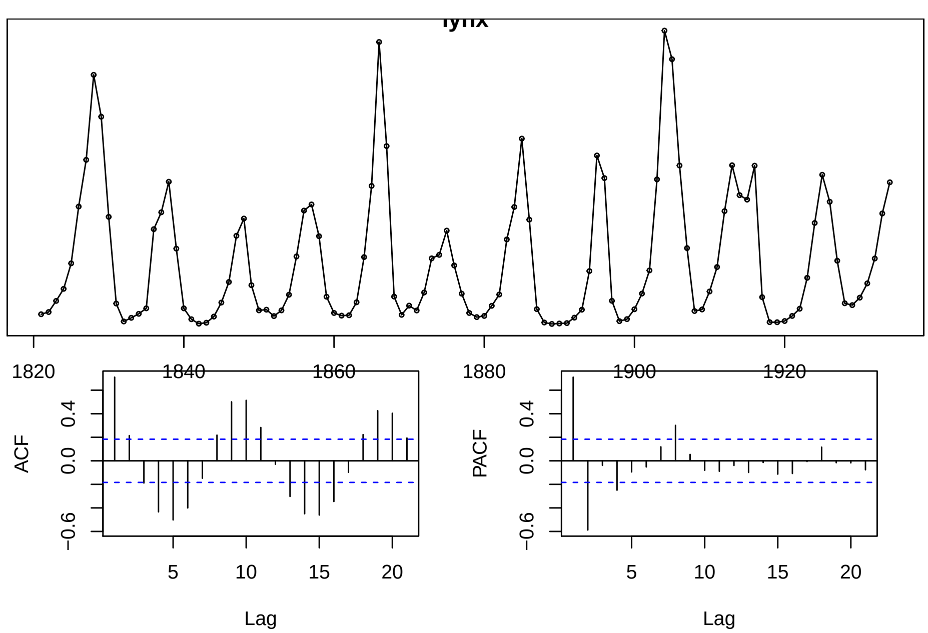
```
## Series: asim
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1     mean
##       0.3494  0.3183  10.0288
## s.e.  0.0478  0.0473   0.0637
##
## sigma^2 estimated as 0.9927:  log likelihood=-1414
## AIC=2836.01   AICc=2836.05   BIC=2855.64
```

```r
## ARIMA parameter selection
adf.test(lynx)
```

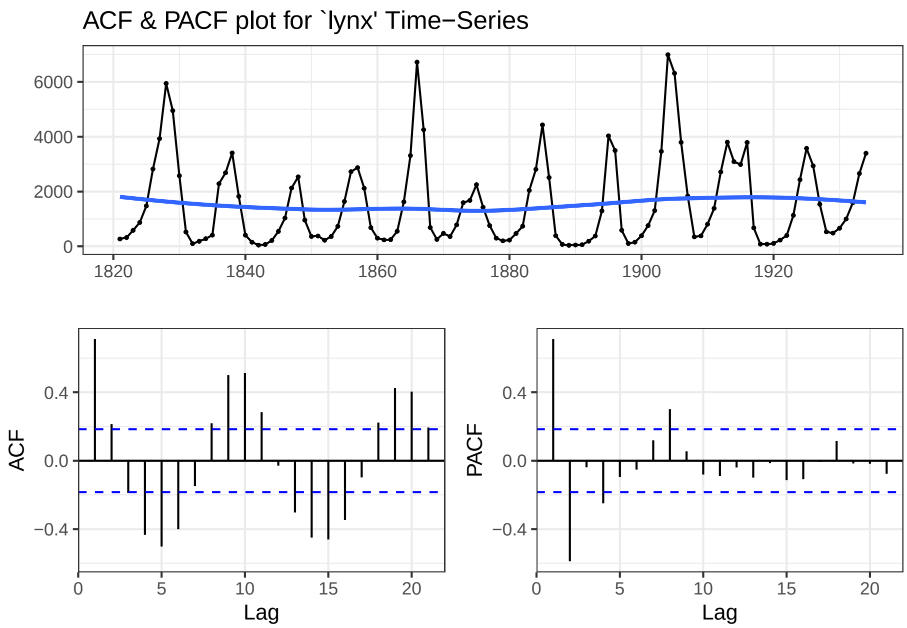
```
## Warning in adf.test(lynx): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lynx
## Dickey-Fuller = -6.3068, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

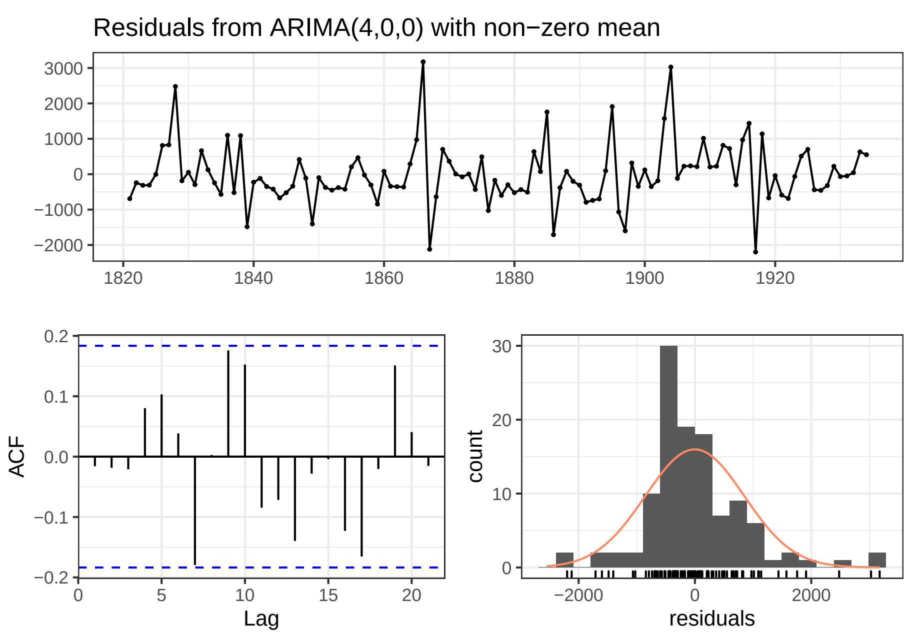
```
tsdisplay(lynx)
```




```
ggtsdisplay(lynx,
            plot.type = "partial",
            main = "ACF & PACF plot for `lynx' Time-Series",
            smooth = TRUE)
```




```r
# Arima from forecast package
myarima <- Arima(lynx,
                 order = c(4,0,0))
checkresiduals(myarima)
```



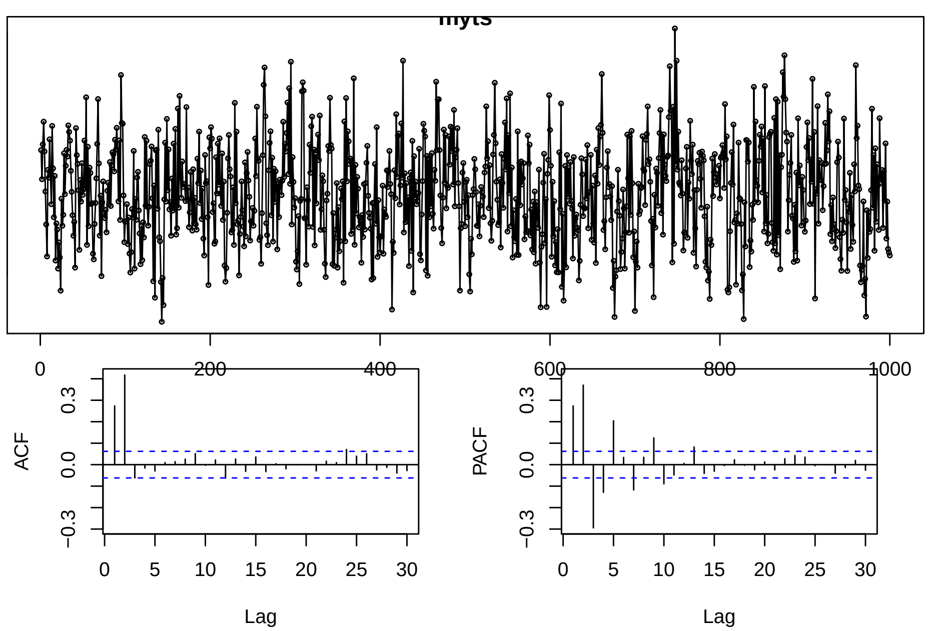
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,0) with non-zero mean
## Q* = 13.201, df = 5, p-value = 0.02157
##
## Model df: 5.   Total lags used: 10
```

```r
# Example MA time series
set.seed(123) # for reproduction
# Simulation
myts <- arima.sim(model = list(order = c(0,0,2),
                               ma = c(0.3, 0.7)), n = 1000) + 10
adf.test(myts) # Stationarity
```

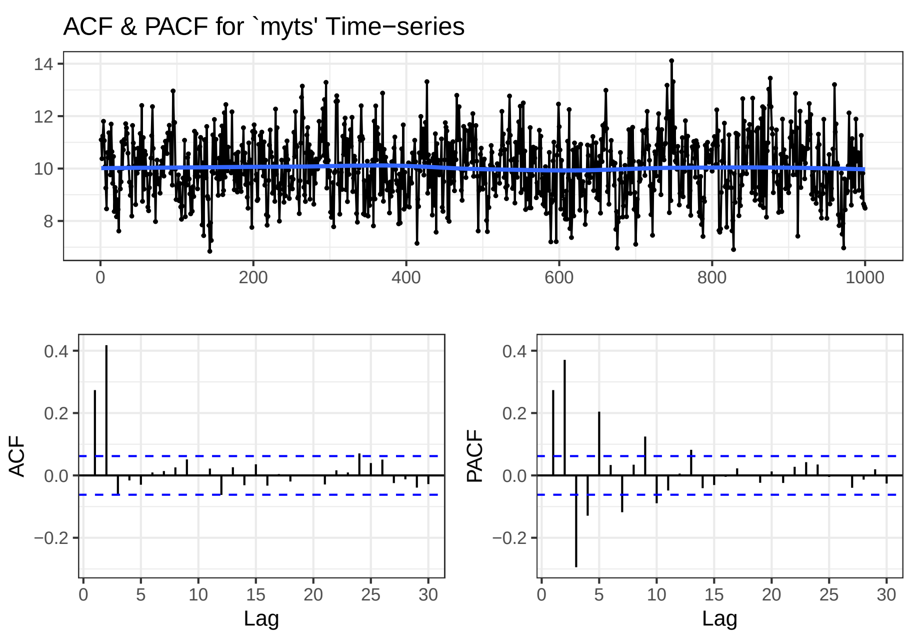
```
## Warning in adf.test(myts): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  myts
## Dickey-Fuller = -9.0469, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

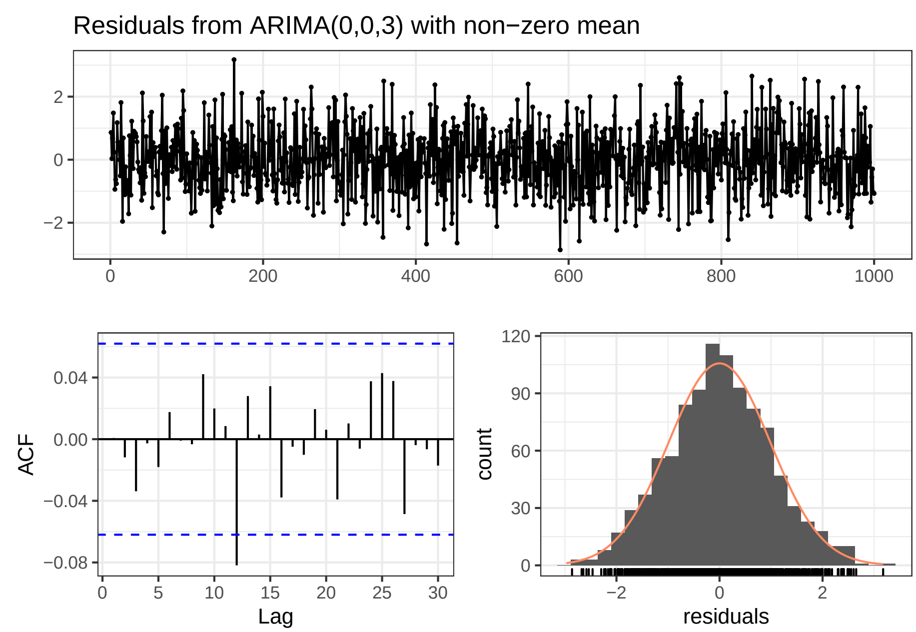
```r
tsdisplay(myts) # Autocorrelation
```



```r
ggtsdisplay(myts,
            plot.type = "partial",
            main = "ACF & PACF for `myts' Time-series",
            smooth = TRUE)
```




```r
# Arima
myarima <- Arima(myts, order = c(0,0,3))
checkresiduals(myarima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,3) with non-zero mean
## Q* = 4.1475, df = 6, p-value = 0.6567
##
## Model df: 4.   Total lags used: 10
```

```
auto.arima(myts,
           trace = T,
           stepwise = F,
           approximation = F)
```

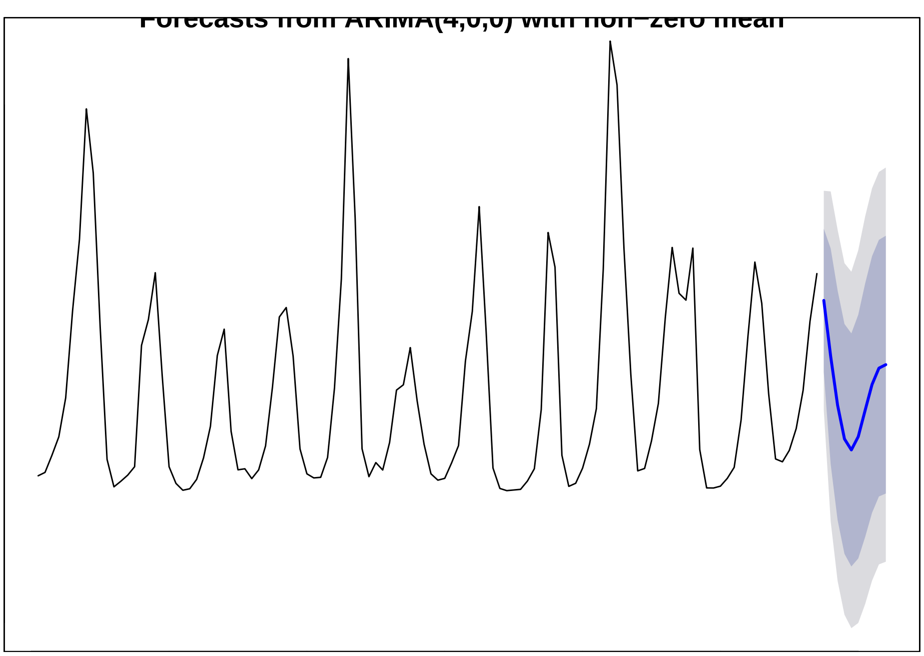
```
##
##  ARIMA(0,0,0) with zero mean     : 7465.902
##  ARIMA(0,0,0) with non-zero mean : 3239.597
##  ARIMA(0,0,1) with zero mean     : 6414.662
##  ARIMA(0,0,1) with non-zero mean : 3199.385
##  ARIMA(0,0,2) with zero mean     : 5571.943
##  ARIMA(0,0,2) with non-zero mean : 2828.282
##  ARIMA(0,0,3) with zero mean     : 4982.239
##  ARIMA(0,0,3) with non-zero mean : 2829.867
##  ARIMA(0,0,4) with zero mean     : 4556.587
##  ARIMA(0,0,4) with non-zero mean : 2831.522
```

```
##  ARIMA(0,0,5) with zero mean     : 4300.593
##  ARIMA(0,0,5) with non-zero mean : 2831.318
##  ARIMA(1,0,0) with zero mean     : 3610.918
##  ARIMA(1,0,0) with non-zero mean : 3163.665
##  ARIMA(1,0,1) with zero mean     : Inf
##  ARIMA(1,0,1) with non-zero mean : 3120.607
##  ARIMA(1,0,2) with zero mean     : Inf
##  ARIMA(1,0,2) with non-zero mean : 2829.89
##  ARIMA(1,0,3) with zero mean     : Inf
##  ARIMA(1,0,3) with non-zero mean : 2831.04
##  ARIMA(1,0,4) with zero mean     : Inf
##  ARIMA(1,0,4) with non-zero mean : 2832.859
##  ARIMA(2,0,0) with zero mean     : Inf
##  ARIMA(2,0,0) with non-zero mean : 3017.436
##  ARIMA(2,0,1) with zero mean     : Inf
##  ARIMA(2,0,1) with non-zero mean : 2977.38
##  ARIMA(2,0,2) with zero mean     : Inf
##  ARIMA(2,0,2) with non-zero mean : 2831.603
##  ARIMA(2,0,3) with zero mean     : Inf
##  ARIMA(2,0,3) with non-zero mean : 2832.823
##  ARIMA(3,0,0) with zero mean     : Inf
##  ARIMA(3,0,0) with non-zero mean : 2929.264
##  ARIMA(3,0,1) with zero mean     : Inf
##  ARIMA(3,0,1) with non-zero mean : 2924.325
##  ARIMA(3,0,2) with zero mean     : Inf
##  ARIMA(3,0,2) with non-zero mean : 2831.357
##  ARIMA(4,0,0) with zero mean     : Inf
##  ARIMA(4,0,0) with non-zero mean : 2914.331
##  ARIMA(4,0,1) with zero mean     : Inf
##  ARIMA(4,0,1) with non-zero mean : 2899.065
##  ARIMA(5,0,0) with zero mean     : Inf
##  ARIMA(5,0,0) with non-zero mean : 2873.303
##
##
##
##  Best model: ARIMA(0,0,2) with non-zero mean
```

```
## Series: myts
## ARIMA(0,0,2) with non-zero mean
##
## Coefficients:
##          ma1     ma2     mean
##       0.2878  0.6838  10.0297
## s.e.  0.0230  0.0231   0.0617
##
## sigma^2 estimated as 0.9842:  log likelihood=-1410.12
```

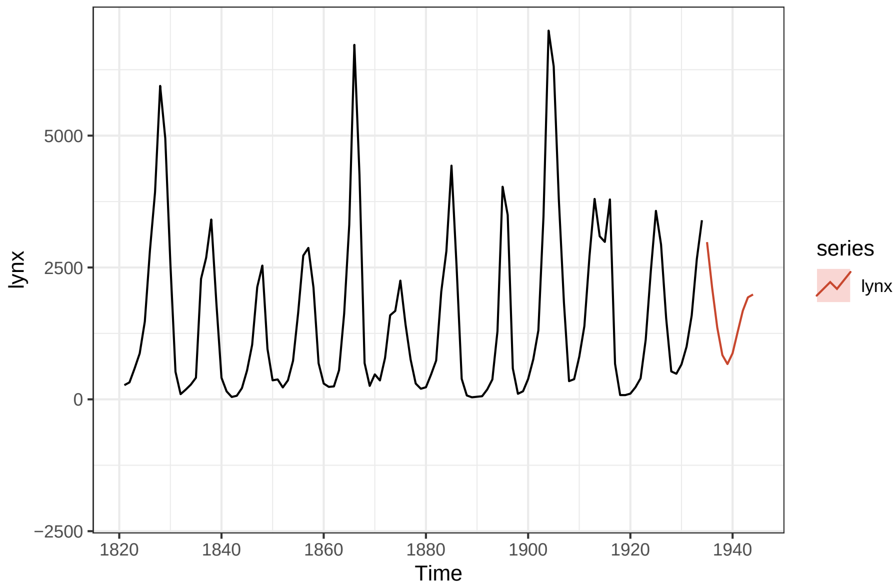
```
## AIC=2828.24   AICc=2828.28   BIC=2847.87
```

```
## ARIMA Forecasting
# Our model
myarima <- auto.arima(lynx,
                      stepwise = F,
                      approximation = F)

# Forecast of 10 years
arimafore <- forecast(myarima, h = 10)
plot(arimafore)
```

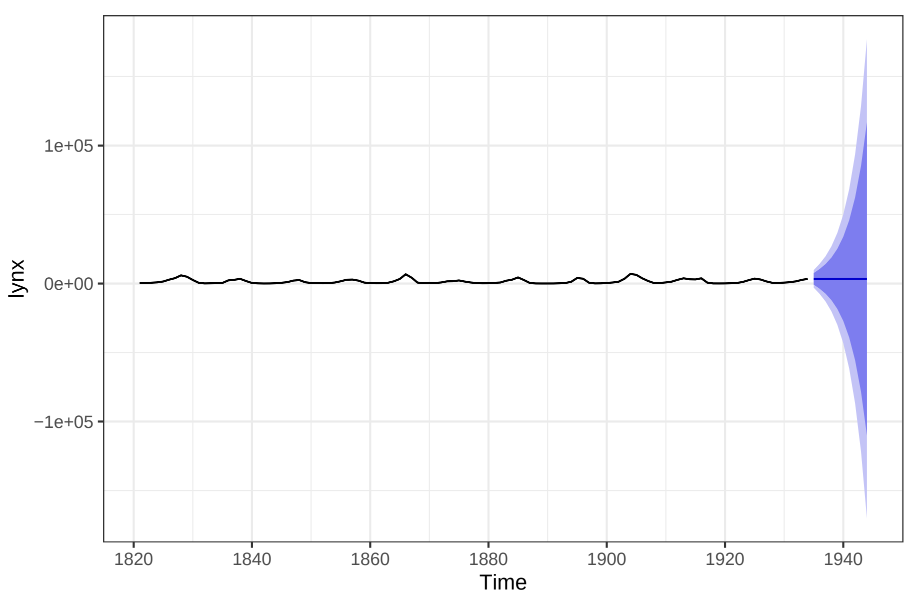



```
autoplot(lynx, PI = TRUE,
         shadecols = c("#596DD5", "#D5DBFF"), fcol = "#0000AA", flwd = 0.5) +
  autolayer(forecast(myarima, h = 10), level = 95, series = "lynx")
```

```
## Warning: Ignoring unknown parameters: PI, shadecols, fcol, flwd
```



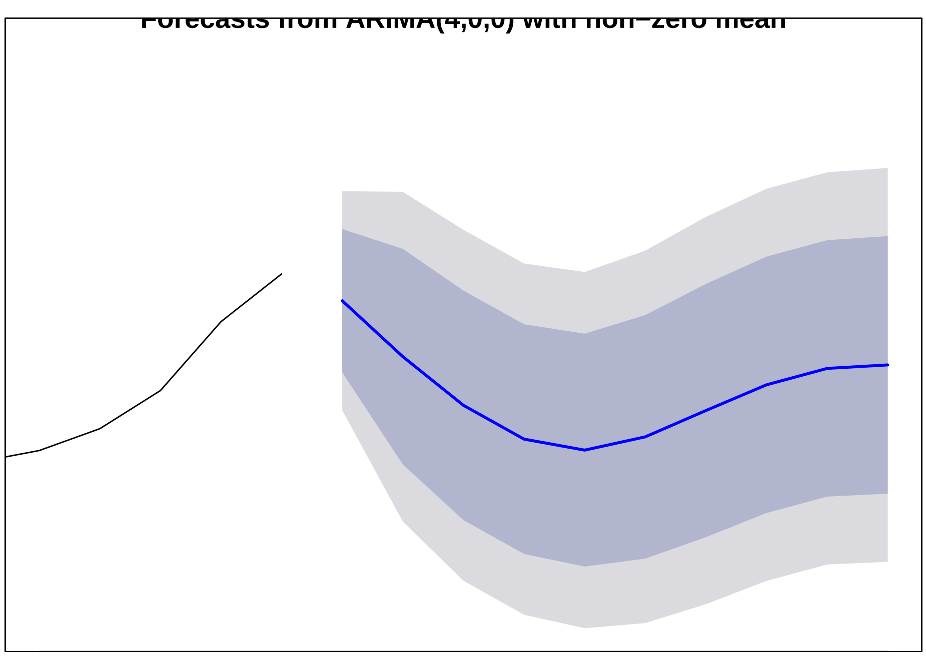

```r
autoplot(lynx) + geom_forecast(h = 10)
```



```r
# See the forecasted values
arimafore$mean
```

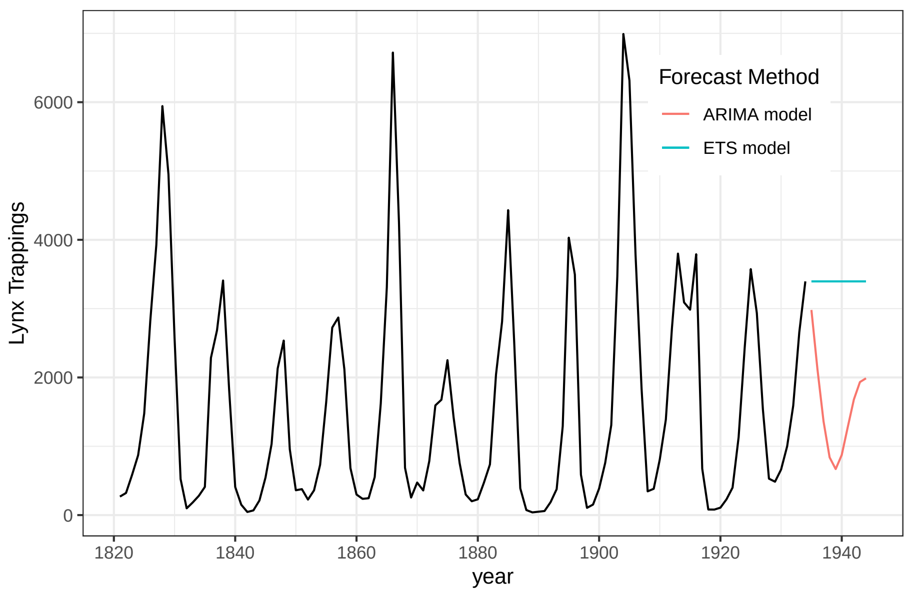
```
## Time Series:
## Start = 1935
## End = 1944
## Frequency = 1
##  [1] 2980.7782 2114.6447 1361.7211  839.0137  668.7873  874.3079 1281.3753
##  [8] 1679.8363 1933.3503 1987.5494
```

```r
# Plot last observations and the forecast
plot(arimafore,
     xlim = c(1930, 1944))
```




```r
# Ets for comparison
myets <- ets(lynx)
etsfore <- forecast(myets, h = 10)

# Comparison plot for 2 models
autoplot(lynx) +
  forecast::autolayer(etsfore$mean, series = 'ETS model') +
  forecast::autolayer(arimafore$mean, series = 'ARIMA model') +
  xlab('year') + ylab('Lynx Trappings') +
```

```
  guides(colour = guide_legend(title = 'Forecast Method')) +
  theme(legend.position = c(0.8, 0.8))
```

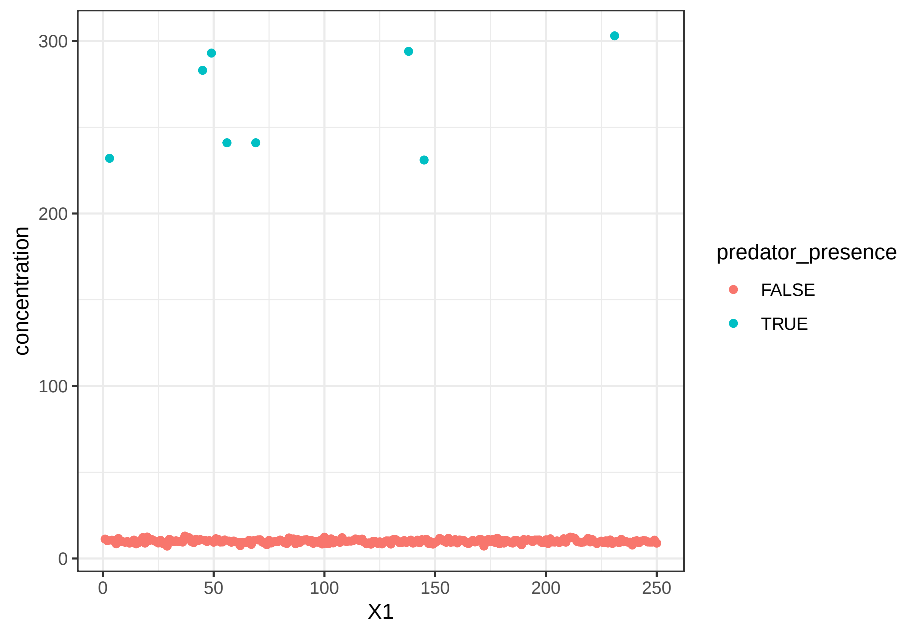



```
## ARIMA with Explanatory Variables - Dynamic Regression
## Importing the cyprinidae dataset
cyprinidae <- read_csv("cyprinidae.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   concentration = col_double(),
##   predator_presence = col_logical()
## )
```

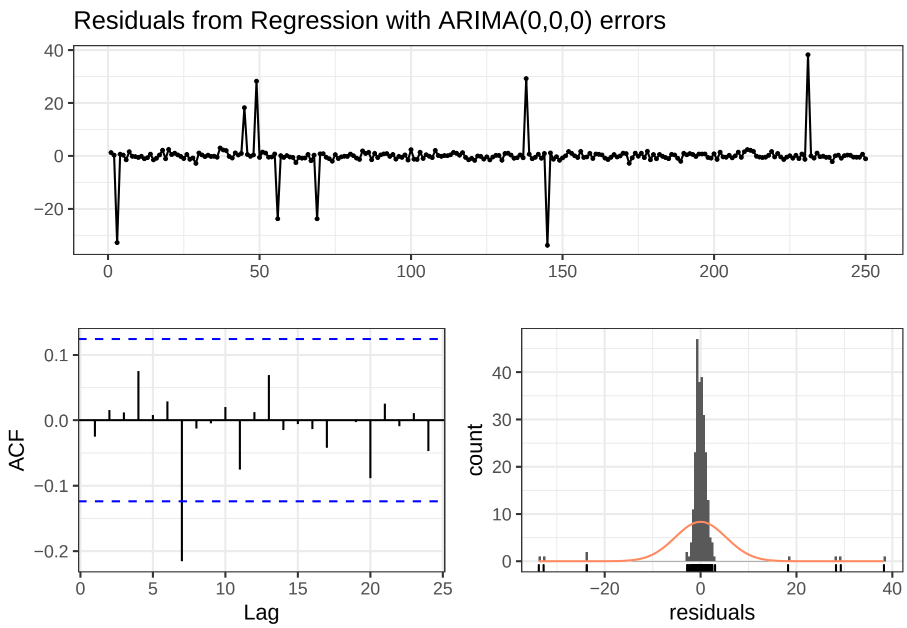
```
# Display the multivariate dataset
ggplot(cyprinidae,
       aes(y = concentration, x = X1)) +
  geom_point () +
  aes(colour = predator_presence)
```




```
# Convert the variables into time series
x = ts(cyprinidae$concentration)
y = as.numeric(cyprinidae$predator_presence)

# Arima model creation
mymodel = auto.arima(x, xreg = y,
                     stepwise = F,
                     approximation = F)
mymodel
```

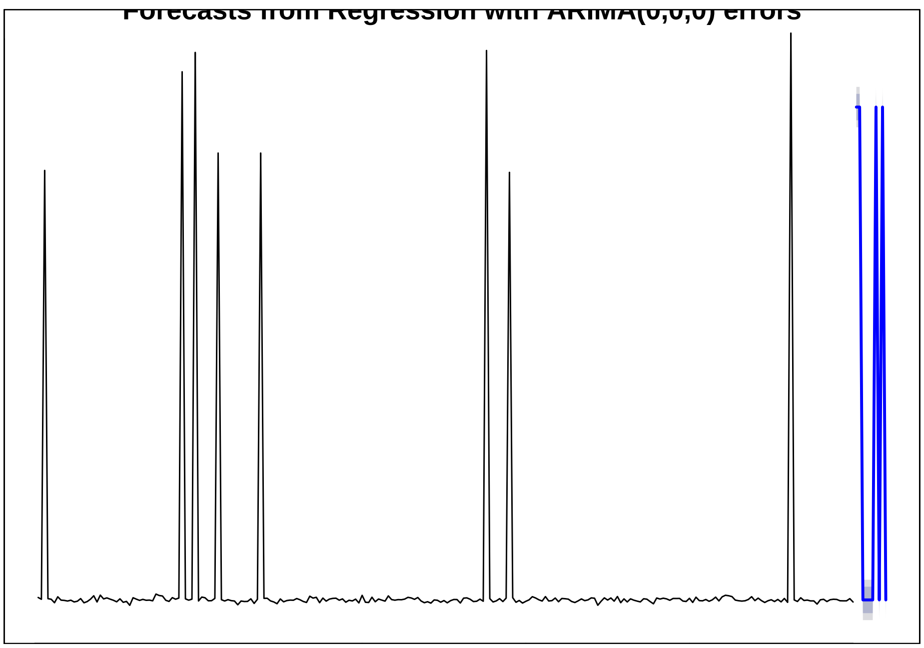
```
## Series: x
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
##       intercept      xreg
##          9.9765  254.7735
## s.e.     0.3409    1.9059
##
## sigma^2 estimated as 28.36:  log likelihood=-771.84
## AIC=1549.68   AICc=1549.77   BIC=1560.24
```

```
# Quick check of model quality
checkresiduals(mymodel)
```

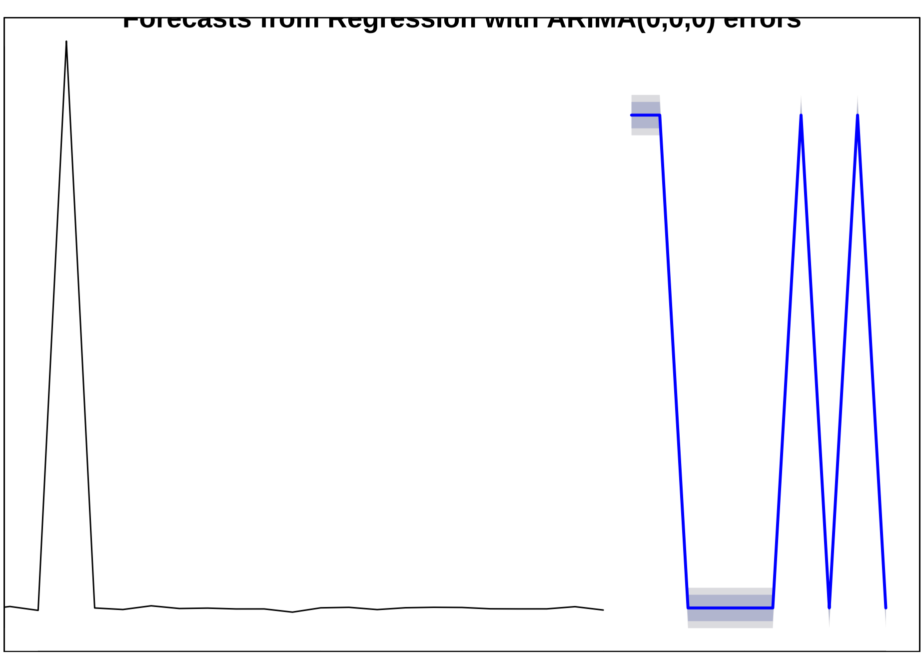



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,0) errors
## Q* = 14.122, df = 8, p-value = 0.07865
##
## Model df: 2.   Total lags used: 10
```

```
# Expected predator presence at future 10 time points
y1 = as.numeric(c(T,T,F,F,F,F,T,F,T,F))

# Getting a forecast based on future predator presence
plot(forecast(mymodel, xreg = y1))
```




```r
plot(forecast(mymodel, xreg = y1),
     xlim  = c(230,260))
```



Chapter 7

Multivariate TS Analysis

```
# preamble setting the directories and loading packages
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section7")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
require(lmtest)
require(TTR)  #for smoothing the time series
require(MTS)
require(vars)
require(fUnitRoots)
require(lattice)
require(grid)
```

```
### Multivariate Time Series Datasets
# Generating a random dataframe
set.seed(40)
x = rnorm(100, 1)
```

```
y = rnorm(100, 30)
z = rnorm(100, 500)

xyz = data.frame(x, y, z)

class(xyz)
```

```
## [1] "data.frame"
```

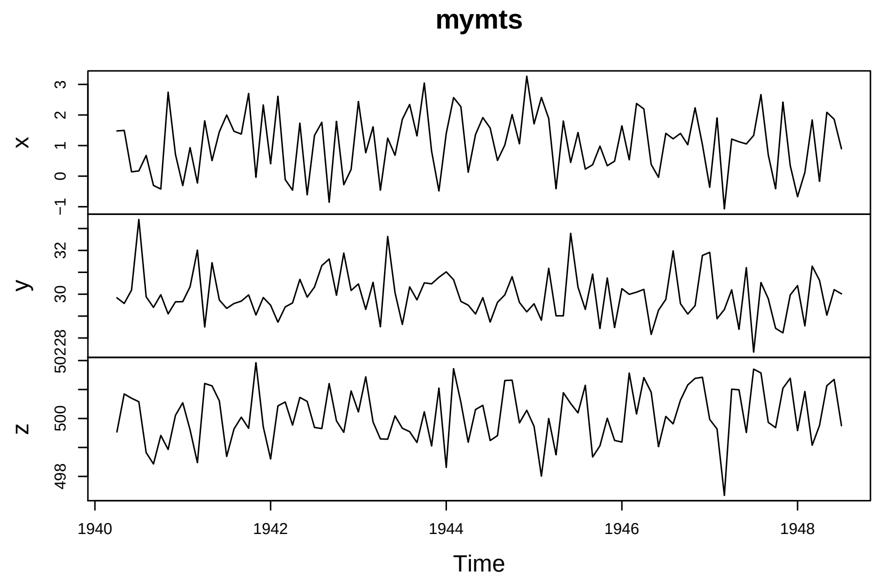
```
# Converting a data.frame into mts
mymts = ts(xyz,
           frequency = 12,
           start = c(1940, 4))
mymts
```

```
##                    x        y        z
## Apr 1940  1.47773904 29.83632 499.5348
## May 1940  1.49618282 29.57833 500.8470
## Jun 1940  0.14041570 30.18632 500.6993
## Jul 1940  0.17094004 33.41376 500.5783
## Aug 1940  0.67842692 29.88003 498.8215
## Sep 1940 -0.30377040 29.39793 498.4307
## Oct 1940 -0.42148660 29.97486 499.4152
## Nov 1940  2.74491495 29.10156 498.9308
## Dec 1940  0.71172064 29.65150 500.1091
## Jan 1941 -0.30886572 29.66069 500.5443
## Feb 1941  0.93054781 30.34087 499.6077
## Mar 1941 -0.22492668 32.01089 498.4770
## Apr 1941  1.80899626 28.50112 501.2086
## May 1941  0.50784966 31.43644 501.1260
## Jun 1941  1.45269393 29.73437 500.6066
## Jul 1941  1.99963310 29.35371 498.6896
## Aug 1941  1.46702960 29.57435 499.6374
## Sep 1941  1.37605199 29.68409 500.0454
## Oct 1941  2.70349095 29.96825 499.6607
## Nov 1941 -0.03546152 29.04966 501.9248
## Dec 1941  2.32812210 29.84856 499.7123
## Jan 1942  0.40571289 29.49766 498.6045
## Feb 1942  2.61128458 28.72180 500.4350
## Mar 1942 -0.11267383 29.41831 500.5698
## Apr 1942 -0.46018298 29.60005 499.7704
## May 1942  1.73215598 30.67877 500.7278
## Jun 1942 -0.61033943 29.86472 500.5900
## Jul 1942  1.33206736 30.32859 499.6907
```

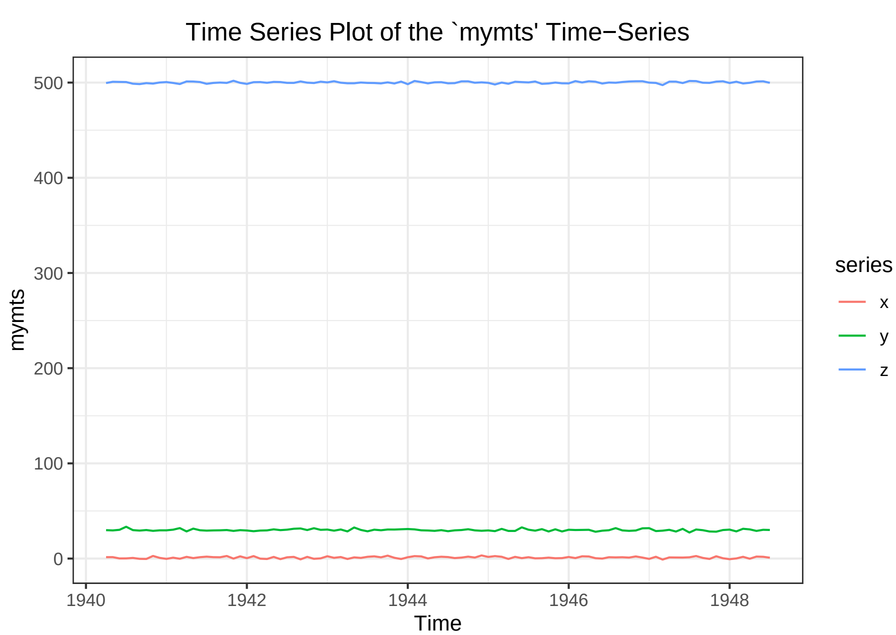
```
## Aug 1942  1.76085616 31.30714 499.6516
## Sep 1942 -0.85366955 31.60610 501.2060
## Oct 1942  1.79115157 29.95238 499.9182
## Nov 1942 -0.28174039 31.88079 499.5227
## Dec 1942  0.22012266 30.16986 500.9506
## Jan 1943  2.43834228 30.46903 500.2272
## Feb 1943  0.76517046 29.30462 501.4385
## Mar 1943  1.61219714 30.54121 499.8786
## Apr 1943 -0.45847063 28.51256 499.2938
## May 1943  1.24276907 32.63799 499.2861
## Jun 1943  0.68260789 30.06999 500.0912
## Jul 1943  1.85937333 28.61832 499.6628
## Aug 1943  2.34415507 30.33621 499.5449
## Sep 1943  1.31555720 29.74085 499.1711
## Oct 1943  3.04544486 30.51940 500.2317
## Nov 1943  0.81752947 30.47371 499.0529
## Dec 1943 -0.48393823 30.77232 501.0480
## Jan 1944  1.39716563 31.01997 498.3103
## Feb 1944  2.56965755 30.66090 501.7181
## Mar 1944  2.27023296 29.67146 500.5579
## Apr 1944  0.12855665 29.49982 499.1805
## May 1944  1.36088326 29.09545 500.3088
## Jun 1944  1.91588753 29.84240 500.4543
## Jul 1944  1.57451813 28.72539 499.2390
## Aug 1944  0.51459597 29.62820 499.4080
## Sep 1944  1.01574267 29.96911 501.3104
## Oct 1944  2.01421468 30.80206 501.3224
## Nov 1944  1.05814926 29.62449 499.8477
## Dec 1944  3.26829549 29.19358 500.2824
## Jan 1945  1.70865319 29.56288 499.7287
## Feb 1945  2.57247878 28.81038 498.0113
## Mar 1945  1.88720230 31.18393 500.0002
## Apr 1945 -0.41166914 29.01367 498.7472
## May 1945  1.80377513 29.01178 500.8901
## Jun 1945  0.44761960 32.77873 500.5178
## Jul 1945  1.42573830 30.32139 500.1941
## Aug 1945  0.22952536 29.30319 501.1462
## Sep 1945  0.37141318 30.91817 498.6691
## Oct 1945  0.98161035 28.43434 499.0585
## Nov 1945  0.34002587 30.74192 500.0088
## Dec 1945  0.48957134 28.47928 499.2432
## Jan 1946  1.64637043 30.25345 499.1858
## Feb 1946  0.53402838 29.99529 501.5672
## Mar 1946  2.37546031 30.09232 500.1563
## Apr 1946  2.19759957 30.22118 501.4116
## May 1946  0.38367022 28.16100 500.9055
```

```
## Jun 1946 -0.03894626 29.26891 499.0285
## Jul 1946  1.39954949 29.76422 500.0708
## Aug 1946  1.22058087 31.97935 499.8144
## Sep 1946  1.39729109 29.56818 500.6330
## Oct 1946  1.02702970 29.09356 501.1575
## Nov 1946  2.23407288 29.47902 501.3850
## Dec 1946  1.03060886 31.77104 501.4232
## Jan 1947 -0.36391409 31.91089 499.9674
## Feb 1947  1.90495376 28.87947 499.6365
## Mar 1947 -1.06942831 29.30213 497.3460
## Apr 1947  1.21278838 30.20043 501.0113
## May 1947  1.12402274 28.39637 500.9910
## Jun 1947  1.05353083 31.21420 499.5156
## Jul 1947  1.33005175 27.35457 501.7041
## Aug 1947  2.66493296 30.53251 501.5721
## Sep 1947  0.69573063 29.79494 499.8651
## Oct 1947 -0.41182405 28.44126 499.6831
## Nov 1947  2.42143748 28.23419 501.0438
## Dec 1947  0.34725532 29.96483 501.3890
## Jan 1948 -0.67167091 30.39152 499.5824
## Feb 1948  0.12753867 28.55380 500.9327
## Mar 1948  1.83922250 31.28148 499.0765
## Apr 1948 -0.16800389 30.63600 499.7599
## May 1948  2.08739965 29.04089 501.1305
## Jun 1948  1.85917397 30.20759 501.3492
## Jul 1948  0.89464946 30.01724 499.7520
```

```
# Standard exploratory tools
plot(mymts)
```




```r
theme_set(theme_bw())
autoplot(mymts) +
  ggtitle("Time Series Plot of the `mymts' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5)) #for centering the text
```



```r
head(mymts)
```

```
##                   x        y        z
## Apr 1940  1.4777390 29.83632 499.5348
## May 1940  1.4961828 29.57833 500.8470
## Jun 1940  0.1404157 30.18632 500.6993
## Jul 1940  0.1709400 33.41376 500.5783
## Aug 1940  0.6784269 29.88003 498.8215
## Sep 1940 -0.3037704 29.39793 498.4307
```

```r
class(mymts)
```

```
## [1] "mts"    "ts"     "matrix"
```

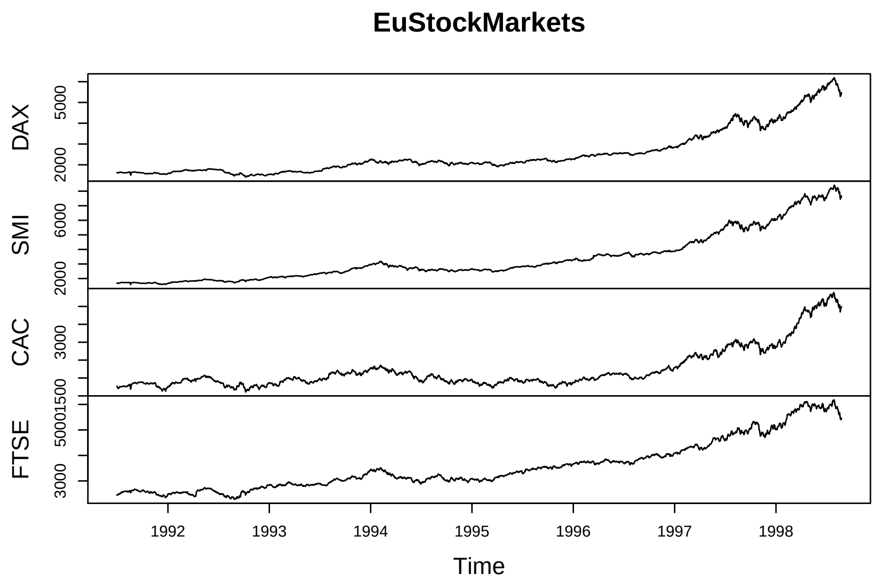
```r
# Our further exercise dataset
class(EuStockMarkets)
```

```
## [1] "mts"    "ts"     "matrix"
```

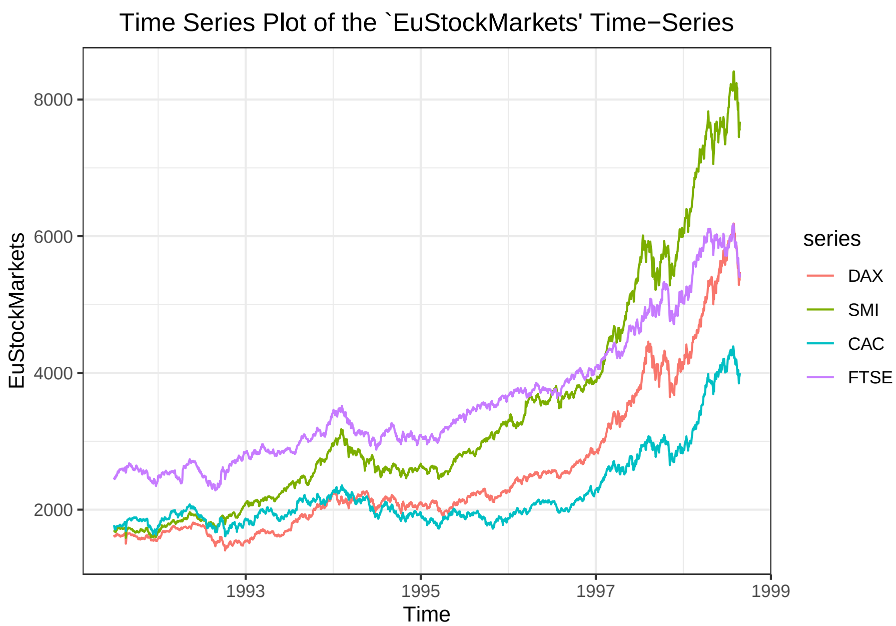
```r
head(EuStockMarkets)
```

```
## Time Series:
## Start = c(1991, 130)
## End = c(1991, 135)
## Frequency = 260
##              DAX    SMI    CAC   FTSE
## 1991.496 1628.75 1678.1 1772.8 2443.6
## 1991.500 1613.63 1688.5 1750.5 2460.2
## 1991.504 1606.51 1678.6 1718.0 2448.2
## 1991.508 1621.04 1684.1 1708.1 2470.4
## 1991.512 1618.16 1686.6 1723.1 2484.7
## 1991.515 1610.61 1671.6 1714.3 2466.8
```

```r
plot(EuStockMarkets)
```




```r
autoplot(EuStockMarkets) +
  ggtitle("Time Series Plot of the `EuStockMarkets' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# Main packages - problem: both have different functions VAR
## Testing for stationarity
### tseries - standard test adt.test
apply(EuStockMarkets, 2, adf.test)
```

```
## Warning in FUN(newX[, i], ...): p-value greater than printed p-value
```

```
## $DAX
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -0.82073, Lag order = 12, p-value = 0.9598
## alternative hypothesis: stationary
##
##
## $SMI
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -0.522, Lag order = 12, p-value = 0.9808
## alternative hypothesis: stationary
##
##
## $CAC
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -0.24897, Lag order = 12, p-value = 0.99
## alternative hypothesis: stationary
##
##
## $FTSE
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -1.9736, Lag order = 12, p-value = 0.5895
## alternative hypothesis: stationary
```

```
# Alternative: lib fUnitRoots, function
apply(EuStockMarkets, 2, adfTest,
```

```
    lags=0, #maximum number of lags used for error term correction
    type="c", #type of unit root regression
    title = "ADF Test for EuStockMarkets Data") #title of the project
```

```
## Warning in FUN(newX[, i], ...): p-value greater than printed p-value

## Warning in FUN(newX[, i], ...): p-value greater than printed p-value

## Warning in FUN(newX[, i], ...): p-value greater than printed p-value
```

```
## $DAX
##
## Title:
##  ADF Test for EuStockMarkets Data
##
## Test Results:
##   PARAMETER:
##     Lag Order: 0
##   STATISTIC:
##     Dickey-Fuller: 1.9429
##   P VALUE:
##     0.99
##
## Description:
##  Mon Aug 19 23:35:56 2019 by user: Tejendra
##
##
## $SMI
##
## Title:
##  ADF Test for EuStockMarkets Data
##
## Test Results:
##   PARAMETER:
##     Lag Order: 0
##   STATISTIC:
##     Dickey-Fuller: 2.2138
##   P VALUE:
##     0.99
##
## Description:
##  Mon Aug 19 23:35:56 2019 by user: Tejendra
##
##
## $CAC
```

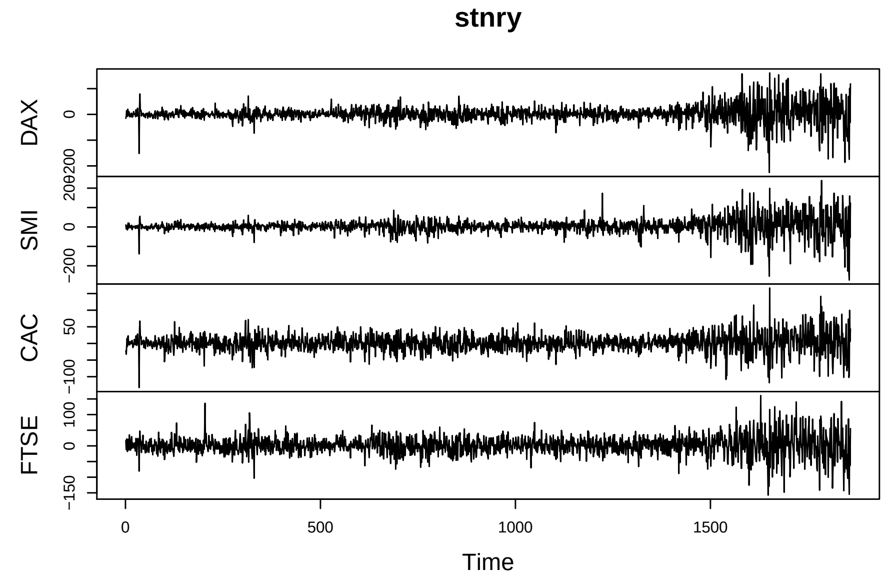
```
##
## Title:
##  ADF Test for EuStockMarkets Data
##
## Test Results:
##   PARAMETER:
##     Lag Order: 0
##   STATISTIC:
##     Dickey-Fuller: 1.2494
##   P VALUE:
##     0.99
##
## Description:
##  Mon Aug 19 23:35:56 2019 by user: Tejendra
##
##
## $FTSE
##
## Title:
##  ADF Test for EuStockMarkets Data
##
## Test Results:
##   PARAMETER:
##     Lag Order: 0
##   STATISTIC:
##     Dickey-Fuller: 0.2207
##   P VALUE:
##     0.9735
##
## Description:
##  Mon Aug 19 23:35:56 2019 by user: Tejendra
```

```
# Differencing the whole mts
stnry = diffM(EuStockMarkets) #difference operation on a vector of time series. Defaul

# Retest
apply(stnry, 2, adf.test)
```

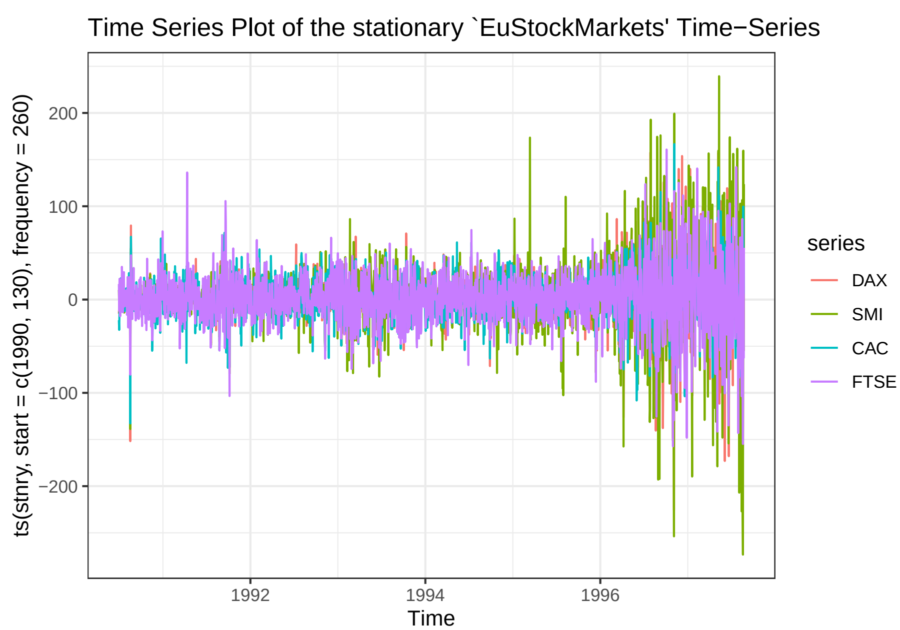
```
## Warning in FUN(newX[, i], ...): p-value smaller than printed p-value

## Warning in FUN(newX[, i], ...): p-value smaller than printed p-value

## Warning in FUN(newX[, i], ...): p-value smaller than printed p-value

## Warning in FUN(newX[, i], ...): p-value smaller than printed p-value
```

```
## $DAX
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -9.9997, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
##
##
## $SMI
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -10.769, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
##
##
## $CAC
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -11.447, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
##
##
## $FTSE
##
##  Augmented Dickey-Fuller Test
##
## data:  newX[, i]
## Dickey-Fuller = -10.838, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

```
## VAR modeling
plot.ts(stnry)
```




```r
autoplot(ts(stnry,
            start = c(1990,130),
            frequency = 260)) +
  ggtitle("Time Series Plot of the stationary `EuStockMarkets' Time-Series")
```



```r
# Lag order identification
#We will use two different functions, from two different packages to identify the lag order for t
VARselect(stnry,
          type = "none", #type of deterministic regressors to include. We use none becasue the ti
          lag.max = 10) #highest lag order
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      1      1      9
##
## $criteria
##                   1            2            3            4            5
## AIC(n) 2.527062e+01 2.527564e+01 2.526566e+01 2.525844e+01 2.525725e+01
## HQ(n)  2.528823e+01 2.531088e+01 2.531850e+01 2.532891e+01 2.534534e+01
## SC(n)  2.531840e+01 2.537122e+01 2.540902e+01 2.544959e+01 2.549619e+01
## FPE(n) 9.438206e+10 9.485771e+10 9.391500e+10 9.324010e+10 9.312964e+10
##                   6            7            8            9           10
## AIC(n) 2.525408e+01 2.525692e+01 2.525696e+01 2.525073e+01 2.525455e+01
## HQ(n)  2.535978e+01 2.538023e+01 2.539789e+01 2.540927e+01 2.543071e+01
## SC(n)  2.554080e+01 2.559143e+01 2.563926e+01 2.568081e+01 2.573242e+01
## FPE(n) 9.283467e+10 9.309877e+10 9.310329e+10 9.252533e+10 9.288047e+10
```

```r
# Creating a VAR model with vars
var.a <- vars::VAR(stnry,
                   lag.max = 10, #highest lag order for lag length selection according to the cho
                   ic = "AIC", #information criterion
                   type = "none") #type of deterministic regressors to include
summary(var.a)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: DAX, SMI, CAC, FTSE
## Deterministic variables: none
## Sample size: 1850
## Log Likelihood: -33712.408
## Roots of the characteristic polynomial:
## 0.817 0.817 0.8116 0.8116 0.7915 0.7915 0.7864 0.7864 0.7784 0.7784 0.7579 0.7579 0.7541 0.754
## Call:
## vars::VAR(y = stnry, type = "none", lag.max = 10, ic = "AIC")
##
##
## Estimation results for equation DAX:
## ===================================
```

```
## DAX = DAX.l1 + SMI.l1 + CAC.l1 + FTSE.l1 + DAX.l2 + SMI.l2 + CAC.l2 + FTSE.l2 + DAX
##
##           Estimate Std. Error t value Pr(>|t|)
## DAX.l1   0.0096570  0.0424492   0.227 0.820065
## SMI.l1  -0.1008170  0.0297641  -3.387 0.000721 ***
## CAC.l1   0.0752689  0.0465795   1.616 0.106285
## FTSE.l1  0.0730055  0.0366170   1.994 0.046328 *
## DAX.l2   0.0190453  0.0423265   0.450 0.652792
## SMI.l2  -0.0172409  0.0298939  -0.577 0.564188
## CAC.l2   0.0687124  0.0465965   1.475 0.140487
## FTSE.l2 -0.0804753  0.0369389  -2.179 0.029489 *
## DAX.l3  -0.0676359  0.0423179  -1.598 0.110155
## SMI.l3   0.0135412  0.0299928   0.451 0.651696
## CAC.l3   0.0484694  0.0466586   1.039 0.299032
## FTSE.l3  0.0409793  0.0369675   1.109 0.267783
## DAX.l4  -0.0501669  0.0422723  -1.187 0.235480
## SMI.l4   0.0162536  0.0300860   0.540 0.589099
## CAC.l4   0.1001510  0.0469324   2.134 0.032981 *
## FTSE.l4 -0.0451988  0.0369319  -1.224 0.221170
## DAX.l5   0.0109497  0.0424940   0.258 0.796687
## SMI.l5  -0.0978623  0.0303192  -3.228 0.001270 **
## CAC.l5   0.0731622  0.0469140   1.559 0.119054
## FTSE.l5 -0.0254787  0.0369942  -0.689 0.491086
## DAX.l6  -0.0121897  0.0424062  -0.287 0.773800
## SMI.l6   0.0246183  0.0303677   0.811 0.417660
## CAC.l6   0.0871855  0.0468724   1.860 0.063039 .
## FTSE.l6  0.0007736  0.0369967   0.021 0.983320
## DAX.l7   0.0786601  0.0425103   1.850 0.064421 .
## SMI.l7  -0.0050826  0.0302543  -0.168 0.866604
## CAC.l7  -0.0691098  0.0466880  -1.480 0.138981
## FTSE.l7 -0.0418380  0.0370855  -1.128 0.259406
## DAX.l8  -0.0336346  0.0425857  -0.790 0.429743
## SMI.l8   0.0963209  0.0304325   3.165 0.001576 **
## CAC.l8  -0.1180253  0.0466645  -2.529 0.011515 *
## FTSE.l8  0.0517022  0.0371047   1.393 0.163666
## DAX.l9  -0.0262047  0.0423049  -0.619 0.535714
## SMI.l9   0.0052002  0.0303603   0.171 0.864022
## CAC.l9   0.1359369  0.0467408   2.908 0.003678 **
## FTSE.l9 -0.0068091  0.0369929  -0.184 0.853983
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 32.09 on 1814 degrees of freedom
## Multiple R-Squared: 0.05129, Adjusted R-squared: 0.03246
## F-statistic: 2.724 on 36 and 1814 DF,  p-value: 2.04e-07
```

```
##
##
## Estimation results for equation SMI:
## ====================================
## SMI = DAX.l1 + SMI.l1 + CAC.l1 + FTSE.l1 + DAX.l2 + SMI.l2 + CAC.l2 + FTSE.l2 + DAX.l3 + SMI.l
##
##          Estimate Std. Error t value Pr(>|t|)
## DAX.l1   0.035132   0.052025   0.675 0.499578
## SMI.l1  -0.038299   0.036478  -1.050 0.293895
## CAC.l1   0.046647   0.057087   0.817 0.413969
## FTSE.l1  0.127516   0.044877   2.841 0.004541 **
## DAX.l2   0.006278   0.051875   0.121 0.903691
## SMI.l2   0.018350   0.036637   0.501 0.616532
## CAC.l2   0.104672   0.057108   1.833 0.066983 .
## FTSE.l2 -0.096675   0.045272  -2.135 0.032859 *
## DAX.l3  -0.148622   0.051864  -2.866 0.004210 **
## SMI.l3   0.004229   0.036759   0.115 0.908422
## CAC.l3   0.094768   0.057184   1.657 0.097644 .
## FTSE.l3  0.131679   0.045307   2.906 0.003701 **
## DAX.l4  -0.175243   0.051808  -3.383 0.000733 ***
## SMI.l4   0.029175   0.036873   0.791 0.428904
## CAC.l4   0.124249   0.057520   2.160 0.030895 *
## FTSE.l4  0.011514   0.045263   0.254 0.799239
## DAX.l5   0.007207   0.052080   0.138 0.889954
## SMI.l5  -0.089506   0.037159  -2.409 0.016106 *
## CAC.l5   0.070892   0.057497   1.233 0.217751
## FTSE.l5 -0.037913   0.045339  -0.836 0.403156
## DAX.l6  -0.072106   0.051972  -1.387 0.165490
## SMI.l6   0.011650   0.037218   0.313 0.754308
## CAC.l6   0.102452   0.057446   1.783 0.074681 .
## FTSE.l6 -0.001026   0.045343  -0.023 0.981944
## DAX.l7   0.147987   0.052100   2.840 0.004555 **
## SMI.l7  -0.012999   0.037079  -0.351 0.725941
## CAC.l7  -0.123208   0.057220  -2.153 0.031432 *
## FTSE.l7 -0.049168   0.045451  -1.082 0.279502
## DAX.l8   0.008599   0.052192   0.165 0.869153
## SMI.l8   0.089777   0.037298   2.407 0.016182 *
## CAC.l8  -0.099393   0.057191  -1.738 0.082397 .
## FTSE.l8 -0.019262   0.045475  -0.424 0.671921
## DAX.l9   0.072664   0.051848   1.401 0.161245
## SMI.l9  -0.091853   0.037209  -2.469 0.013657 *
## CAC.l9   0.081425   0.057285   1.421 0.155371
## FTSE.l9  0.068442   0.045338   1.510 0.131322
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##
## Residual standard error: 39.33 on 1814 degrees of freedom
## Multiple R-Squared: 0.05981, Adjusted R-squared: 0.04115
## F-statistic: 3.206 on 36 and 1814 DF,  p-value: 7.227e-10
##
##
## Estimation results for equation CAC:
## ====================================
## CAC = DAX.l1 + SMI.l1 + CAC.l1 + FTSE.l1 + DAX.l2 + SMI.l2 + CAC.l2 + FTSE.l2 + DAX
##
##          Estimate Std. Error t value Pr(>|t|)
## DAX.l1  -0.001041   0.034340  -0.030  0.97582
## SMI.l1  -0.071184   0.024078  -2.956  0.00315 **
## CAC.l1   0.043492   0.037681   1.154  0.24857
## FTSE.l1  0.082268   0.029622   2.777  0.00554 **
## DAX.l2   0.014488   0.034241   0.423  0.67225
## SMI.l2  -0.027912   0.024183  -1.154  0.24857
## CAC.l2   0.083842   0.037695   2.224  0.02626 *
## FTSE.l2 -0.063578   0.029882  -2.128  0.03350 *
## DAX.l3  -0.032437   0.034234  -0.948  0.34350
## SMI.l3   0.031449   0.024263   1.296  0.19508
## CAC.l3  -0.059480   0.037745  -1.576  0.11524
## FTSE.l3  0.023769   0.029905   0.795  0.42683
## DAX.l4  -0.112680   0.034197  -3.295  0.00100 **
## SMI.l4   0.045902   0.024338   1.886  0.05946 .
## CAC.l4   0.071056   0.037967   1.872  0.06143 .
## FTSE.l4 -0.020521   0.029877  -0.687  0.49225
## DAX.l5  -0.040047   0.034376  -1.165  0.24418
## SMI.l5  -0.040002   0.024527  -1.631  0.10308
## CAC.l5   0.044130   0.037952   1.163  0.24507
## FTSE.l5 -0.011466   0.029927  -0.383  0.70166
## DAX.l6  -0.010487   0.034305  -0.306  0.75987
## SMI.l6   0.017464   0.024566   0.711  0.47724
## CAC.l6   0.046108   0.037918   1.216  0.22415
## FTSE.l6 -0.002253   0.029929  -0.075  0.94000
## DAX.l7   0.093443   0.034389   2.717  0.00665 **
## SMI.l7  -0.011696   0.024475  -0.478  0.63280
## CAC.l7  -0.058576   0.037769  -1.551  0.12110
## FTSE.l7 -0.059667   0.030001  -1.989  0.04687 *
## DAX.l8   0.012292   0.034450   0.357  0.72128
## SMI.l8   0.026246   0.024619   1.066  0.28653
## CAC.l8  -0.102523   0.037750  -2.716  0.00667 **
## FTSE.l8  0.048842   0.030016   1.627  0.10387
## DAX.l9   0.019936   0.034223   0.583  0.56028
## SMI.l9  -0.025465   0.024560  -1.037  0.29995
## CAC.l9   0.048093   0.037812   1.272  0.20357
```

```
## FTSE.l9 -0.003901   0.029926  -0.130  0.89630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 25.96 on 1814 degrees of freedom
## Multiple R-Squared: 0.04515, Adjusted R-squared: 0.0262
## F-statistic: 2.382 on 36 and 1814 DF,  p-value: 8.732e-06
##
##
## Estimation results for equation FTSE:
## =====================================
## FTSE = DAX.l1 + SMI.l1 + CAC.l1 + FTSE.l1 + DAX.l2 + SMI.l2 + CAC.l2 + FTSE.l2 + DAX.l3 + SMI.
##
##          Estimate Std. Error t value Pr(>|t|)
## DAX.l1   0.025600   0.039796   0.643  0.52012
## SMI.l1  -0.085874   0.027904  -3.078  0.00212 **
## CAC.l1  -0.003879   0.043668  -0.089  0.92923
## FTSE.l1  0.165616   0.034328   4.824 1.52e-06 ***
## DAX.l2   0.023464   0.039681   0.591  0.55438
## SMI.l2  -0.023240   0.028025  -0.829  0.40708
## CAC.l2   0.028324   0.043684   0.648  0.51683
## FTSE.l2 -0.031301   0.034630  -0.904  0.36619
## DAX.l3  -0.052914   0.039673  -1.334  0.18245
## SMI.l3   0.012312   0.028118   0.438  0.66154
## CAC.l3   0.057729   0.043742   1.320  0.18709
## FTSE.l3  0.007780   0.034657   0.224  0.82240
## DAX.l4  -0.054187   0.039630  -1.367  0.17170
## SMI.l4   0.043084   0.028206   1.527  0.12681
## CAC.l4   0.078160   0.043999   1.776  0.07583 .
## FTSE.l4 -0.083589   0.034624  -2.414  0.01587 *
## DAX.l5   0.001615   0.039838   0.041  0.96767
## SMI.l5  -0.042176   0.028424  -1.484  0.13803
## CAC.l5   0.102931   0.043982   2.340  0.01938 *
## FTSE.l5 -0.069017   0.034682  -1.990  0.04674 *
## DAX.l6  -0.027039   0.039756  -0.680  0.49651
## SMI.l6   0.058310   0.028470   2.048  0.04069 *
## CAC.l6   0.094202   0.043943   2.144  0.03219 *
## FTSE.l6 -0.088315   0.034684  -2.546  0.01097 *
## DAX.l7   0.054056   0.039853   1.356  0.17514
## SMI.l7   0.052084   0.028363   1.836  0.06648 .
## CAC.l7  -0.065521   0.043770  -1.497  0.13458
## FTSE.l7 -0.055592   0.034768  -1.599  0.11000
## DAX.l8  -0.004926   0.039924  -0.123  0.90181
## SMI.l8   0.057267   0.028530   2.007  0.04488 *
## CAC.l8  -0.080907   0.043748  -1.849  0.06456 .
```

```
## FTSE.l8  0.017291   0.034786   0.497  0.61919
## DAX.l9   0.003630   0.039661   0.092  0.92709
## SMI.l9  -0.017471   0.028463  -0.614  0.53942
## CAC.l9   0.069078   0.043819   1.576  0.11510
## FTSE.l9  0.012475   0.034681   0.360  0.71911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 30.08 on 1814 degrees of freedom
## Multiple R-Squared: 0.05941, Adjusted R-squared: 0.04075
## F-statistic: 3.183 on 36 and 1814 DF,  p-value: 9.501e-10
##
##
##
## Covariance matrix of residuals:
##         DAX    SMI   CAC  FTSE
## DAX  1025.7  940.4 616.8 647.6
## SMI   940.4 1537.5 650.9 722.0
## CAC   616.8  650.9 672.0 522.2
## FTSE  647.6  722.0 522.2 903.2
##
## Correlation matrix of residuals:
##         DAX    SMI    CAC   FTSE
## DAX  1.0000 0.7488 0.7430 0.6729
## SMI  0.7488 1.0000 0.6403 0.6127
## CAC  0.7430 0.6403 1.0000 0.6703
## FTSE 0.6729 0.6127 0.6703 1.0000
```

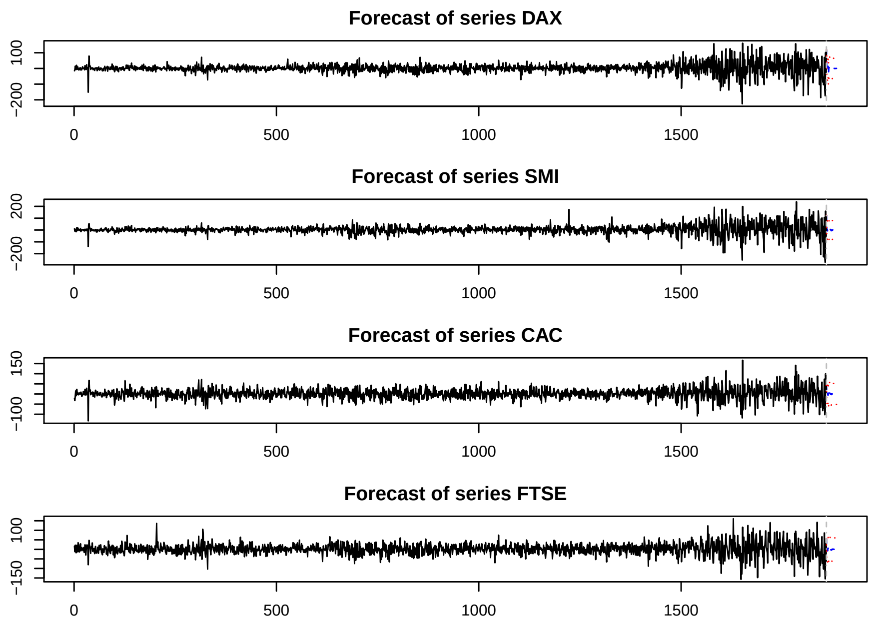
```r
# Residual diagnostics
#serial.test function takes the VAR model as the input.
serial.test(var.a)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.a
## Chi-squared = 183.36, df = 112, p-value = 2.444e-05
```

```r
#selecting the variables
# Granger test for causality
#for causality function to give reliable results we need all the variables of the mult
causality(var.a, #VAR model
          cause = c("DAX")) #cause variable. If not specified then first column of x i
```

```
## $Granger
##
##  Granger causality H0: DAX do not Granger-cause SMI CAC FTSE
##
## data:  VAR object var.a
## F-Test = 1.7314, df1 = 27, df2 = 7256, p-value = 0.01074
##
##
## $Instant
##
##  H0: No instantaneous causality between: DAX and SMI CAC FTSE
##
## data:  VAR object var.a
## Chi-squared = 759.19, df = 3, p-value < 2.2e-16
```

```
## Forecasting VAR models
fcast = predict(var.a, n.ahead = 25) # we forecast over a short horizon because beyond short hor
par(mar = c(2.5,2.5,2.5,2.5))
plot(fcast)
```




```
# Forecasting the DAX index
DAX = fcast$fcst[1]; DAX # type list
```

```
## $DAX
```

```
##              fcst     lower    upper       CI
##  [1,]   5.60084304 -57.29384 68.49552 62.89468
##  [2,]  -5.91965377 -69.10567 57.26636 63.18601
##  [3,] -13.00325572 -76.29843 50.29191 63.29517
##  [4,]  13.72698427 -49.63068 77.08465 63.35767
##  [5,] -36.35436670 -99.79645 27.08772 63.44208
##  [6,]  -0.64933493 -64.37759 63.07892 63.72826
##  [7,]   5.96623225 -57.93847 69.87093 63.90470
##  [8,]   8.82501176 -55.17502 72.82505 64.00004
##  [9,]   2.15548901 -62.14270 66.45368 64.29819
## [10,]   2.12089142 -62.42840 66.67018 64.54929
## [11,]  -0.80619025 -65.36242 63.75003 64.55622
## [12,]  -4.24776524 -68.81260 60.31707 64.56484
## [13,]   0.31282719 -64.25708 64.88274 64.56991
## [14,]  -2.10767272 -66.68072 62.46537 64.57304
## [15,]   1.30500015 -63.28275 65.89275 64.58775
## [16,]   0.23742105 -64.35338 64.82822 64.59080
## [17,]   1.78179652 -62.81456 66.37815 64.59635
## [18,]  -0.36003725 -64.96020 64.24012 64.60016
## [19,]   0.72242303 -63.87835 65.32319 64.60077
## [20,]  -0.69067282 -65.29176 63.91041 64.60109
## [21,]  -0.43392018 -65.03547 64.16762 64.60155
## [22,]  -0.39358579 -64.99520 64.20802 64.60161
## [23,]  -0.08492216 -64.68679 64.51694 64.60186
## [24,]   0.19373467 -64.40828 64.79575 64.60202
## [25,]   0.22812144 -64.37395 64.83020 64.60207
```

```
# Extracting the forecast column
x = DAX$DAX[,1]; x
```

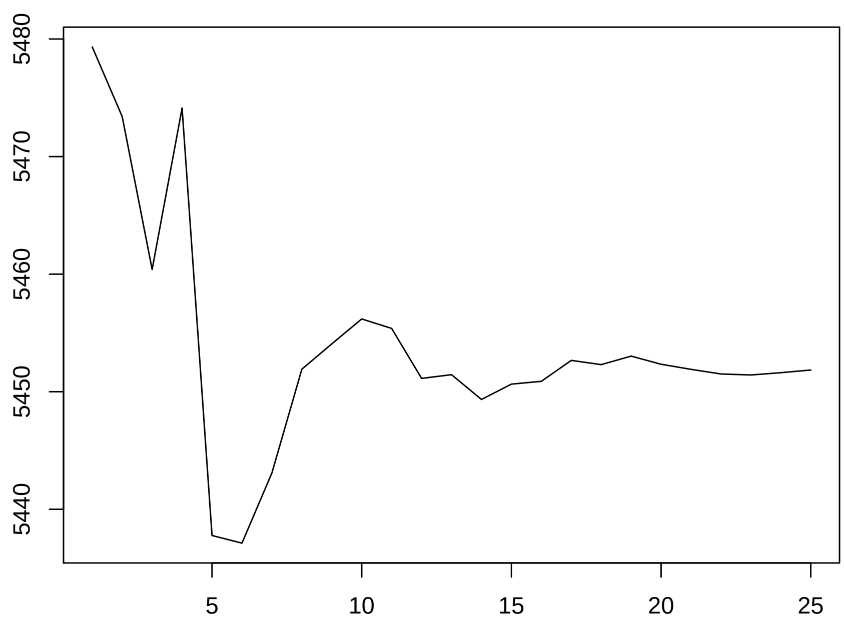
```
##  [1]   5.60084304  -5.91965377 -13.00325572  13.72698427 -36.35436670
##  [6]  -0.64933493   5.96623225   8.82501176   2.15548901   2.12089142
## [11]  -0.80619025  -4.24776524   0.31282719  -2.10767272   1.30500015
## [16]   0.23742105   1.78179652  -0.36003725   0.72242303  -0.69067282
## [21]  -0.43392018  -0.39358579  -0.08492216   0.19373467   0.22812144
```

```
tail(EuStockMarkets)
```

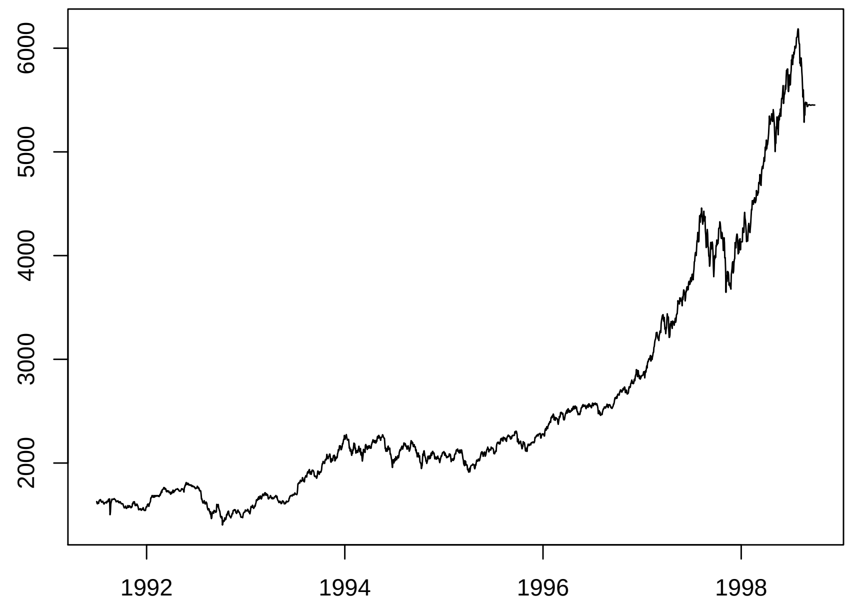
```
## Time Series:
## Start = c(1998, 164)
## End = c(1998, 169)
## Frequency = 260
##              DAX    SMI    CAC   FTSE
## 1998.627 5598.32 7952.9 4041.9 5680.4
```

```
## 1998.631 5460.43 7721.3 3939.5 5587.6
## 1998.635 5285.78 7447.9 3846.0 5432.8
## 1998.638 5386.94 7607.5 3945.7 5462.2
## 1998.642 5355.03 7552.6 3951.7 5399.5
## 1998.646 5473.72 7676.3 3995.0 5455.0
```

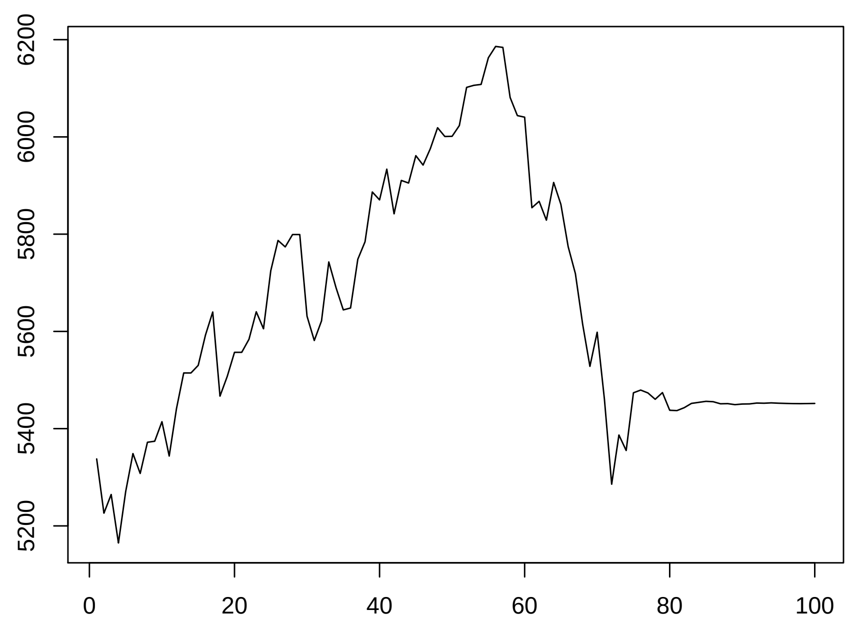
```
# Inverting the differencing
#To get the data to the original scale we invert the time series
#since the values are just difference from the previous value, to get the values on the original
#the plot of the predicted values will also show that over longer horizon the predicted values ar
x = cumsum(x) + 5473.72
par(mar = c(2.5,2.5,1,2.5)) #bottom, left, top, and right
plot.ts(x)
```




```
# Adding data and forecast to one time series
DAXinv =ts(c(EuStockMarkets[,1], x),
           start = c(1991,130),
           frequency = 260)
plot(DAXinv)
```



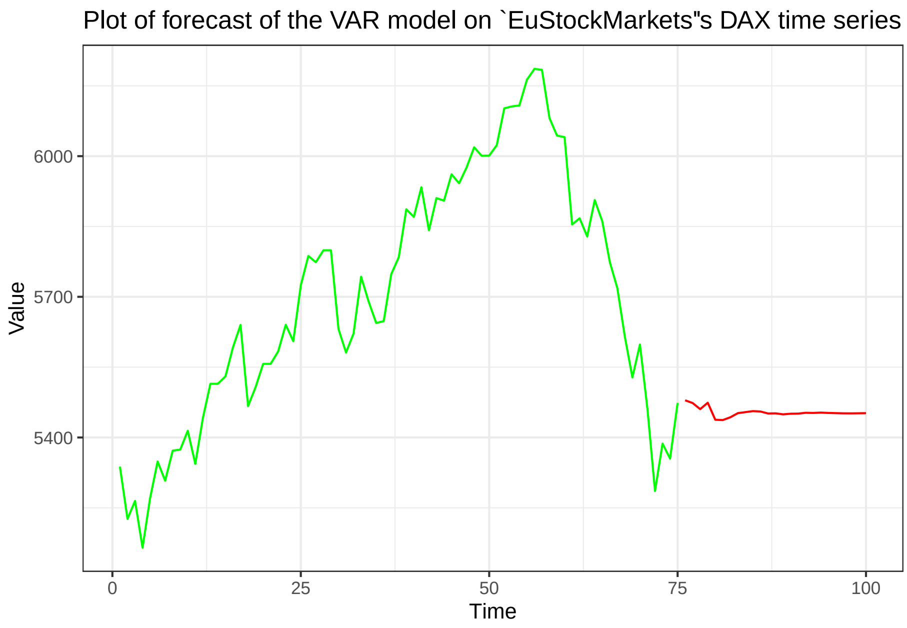

```
plot.ts(DAXinv[1786:1885])
```



```
DAXinv_datframe <- as.data.frame(DAXinv[1786:1885])
colnames(DAXinv_datframe) <- c("x")
head(DAXinv_datframe)
```

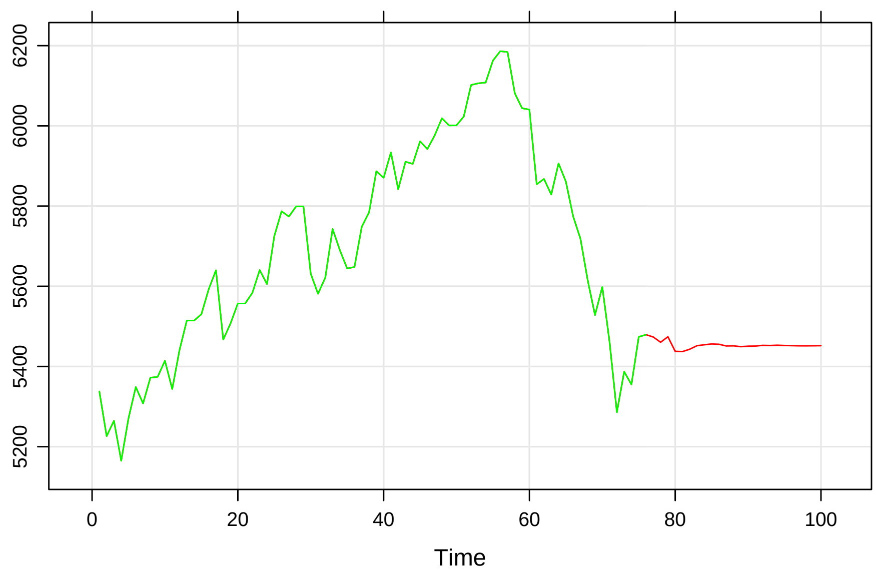
```
##         x
## 1 5337.75
## 2 5226.20
## 3 5264.62
## 4 5164.89
## 5 5270.61
## 6 5348.75
```

```
ggplot() +
  geom_line(data = as.data.frame(DAXinv_datframe[1:75,]), aes(y = get("DAXinv_datframe[1:75, ]"),
  geom_line(data = as.data.frame(DAXinv_datframe[76:100,]), aes(y = get("DAXinv_datframe[76:100,
  ggtitle("Plot of forecast of the VAR model on `EuStockMarkets''s DAX time series") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Time") + ylab("Value")
```

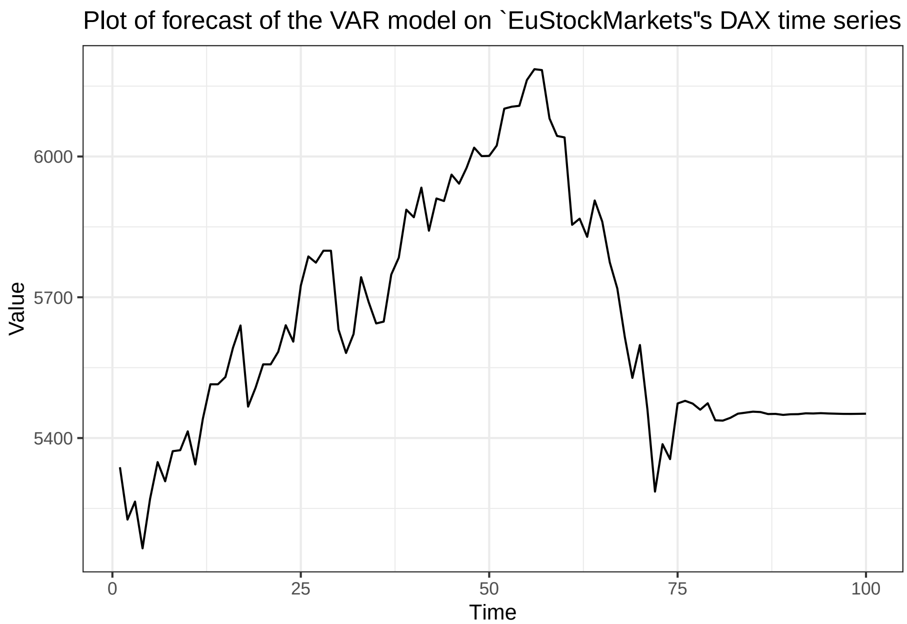



```
## Creating an advanced plot with visual separation
# Converting to object zoo
x = zoo(DAXinv[1786:1885])
```

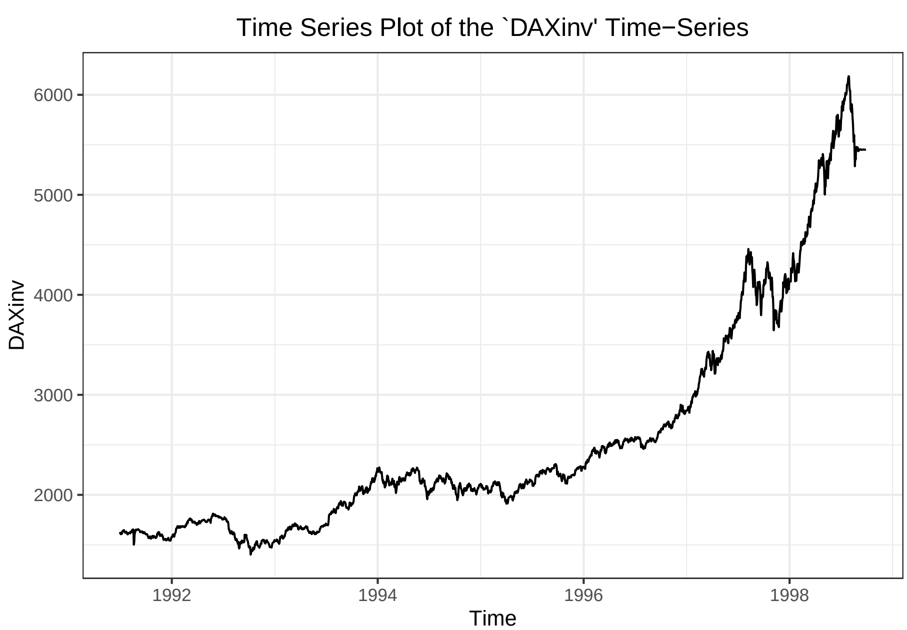
```
# Advanced xyplot from lattice
xyplot(x, grid=TRUE, panel = function(x, y, ...){
  panel.xyplot(x, y, col="red", ...)
  grid.clip(x = unit(76, "native"), just=c("right"))
  panel.xyplot(x, y, col="green", ...) })
```




```
#we repeat the plots from above using the ggplot2 package
# Inverting the differencing
x_dat_frame <- as.data.frame(x)
ggplot(x_dat_frame, aes(y = x, x = seq(1, length(x_dat_frame$x)))) +
  geom_line() +
  ggtitle("Plot of forecast of the VAR model on `EuStockMarkets''s DAX time series") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Time") + ylab("Value")
```




```r
# Adding data and forecast to one time series
autoplot(DAXinv) +
  ggtitle("Time Series Plot of the `DAXinv' Time-Series") +
  theme(plot.title = element_text(hjust = 0.5))
```



Chapter 8

Neural Networks in Time Series Analysis

```
# preamble setting the directories and loading packages
rm(list = ls())
setwd("C:/Users/Tejendra/Desktop/FoldersOnDesktop/UdemyCourse/Section7")
require(tidyverse)
require(tidymodels)
require(data.table)
require(tidyposterior)
require(tsibble)  #tsibble for time series based on tidy principles
require(fable)  #for forecasting based on tidy principles
require(ggfortify)  #for plotting timeseries
require(forecast)  #for forecast function
require(tseries)
require(chron)
require(lubridate)
require(directlabels)
require(zoo)
require(lmtest)
require(TTR)  #for smoothing the time series
require(MTS)
require(vars)
require(fUnitRoots)
require(lattice)
require(grid)
```

NNAR-Neural Network Autoregression Model- has two components, p & k. p denotes the number of lagged values that are used as inputs. k denotes the number of hidden nodes that are present. Output is denoted by $NNAR(p, k)$. If the dataset is seasonal then also the notation is pretty similar, i.e., $NNAR(p, P, k)$ where P denotes the number of seasonal lags. p is choosen based on the information criterion, like AIC. Neural nets has inherent random component. Therefore, it is suggested that the neural net model is run several times, 20 is the minimum requirement. Final result is then presented as mean or median. Also neural nets are known to not work well with the trend data. We should therefore, de-trend or differnce the data before running neural net model.

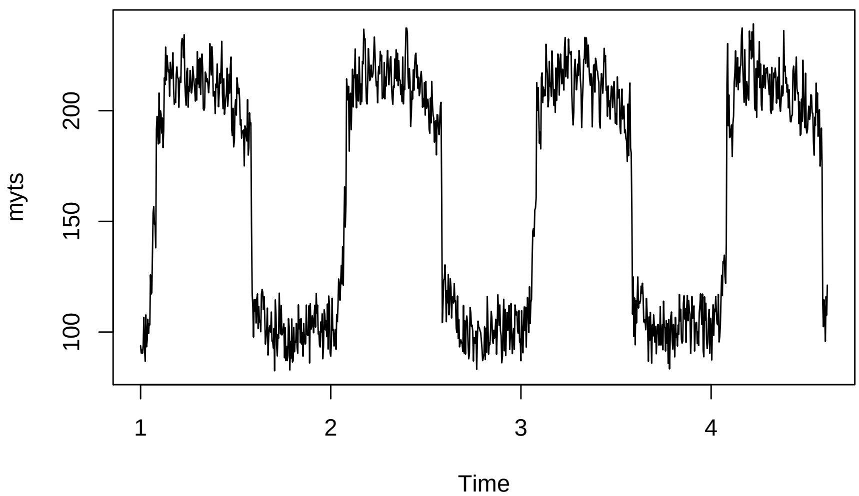
Looking at the data we see that the relationship between appliances and watt is unidirectional as the number of appliances impact the electricity consumption and not vice versa. Therefore, appliances can be used as an external regressor in the model. Moreover, since electricity consumption fluctuates a lot daily, we will use daily frequency. Most models are not good at handling large frequency data. One way out of this problem is to use the aggregation. But this might lead to over smoothening in the data. We also check for all the columns to have same length and all of them being numeric. There are many packages that allows one to compute neural net models. However, *nnetar*() from *forecast* is most user friendly. One handy thing about *nnetar*() is automatic selection of parameters. For more advanced implementation of the neural nets one can look at *mlp*() function form *nnfor* package.

```r
## Import the APTelectricity.csv file as APTelectric
APTelectric <- read_csv("APTelectricity.csv")
```

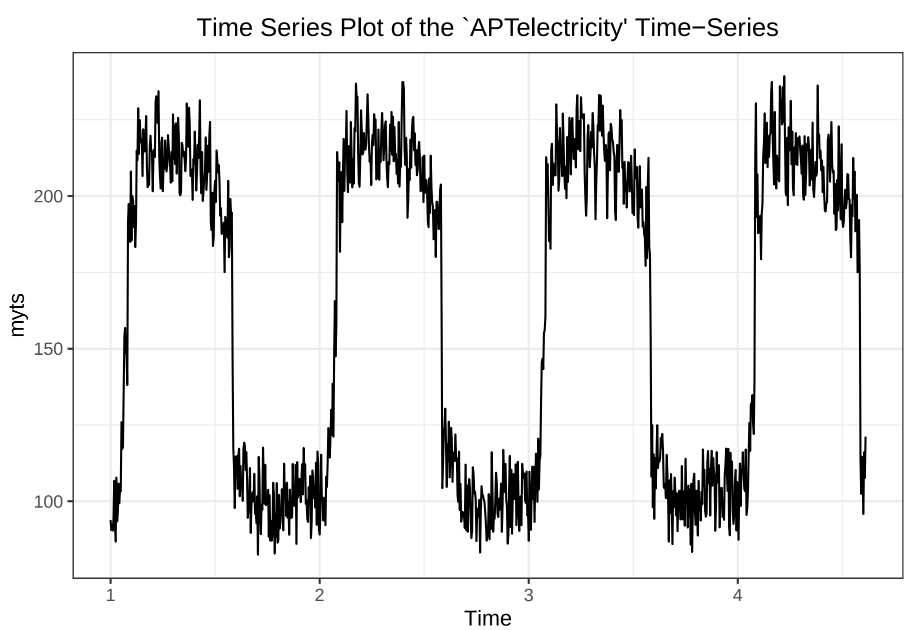
```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   watt = col_double(),
##   appliances = col_double()
## )
```

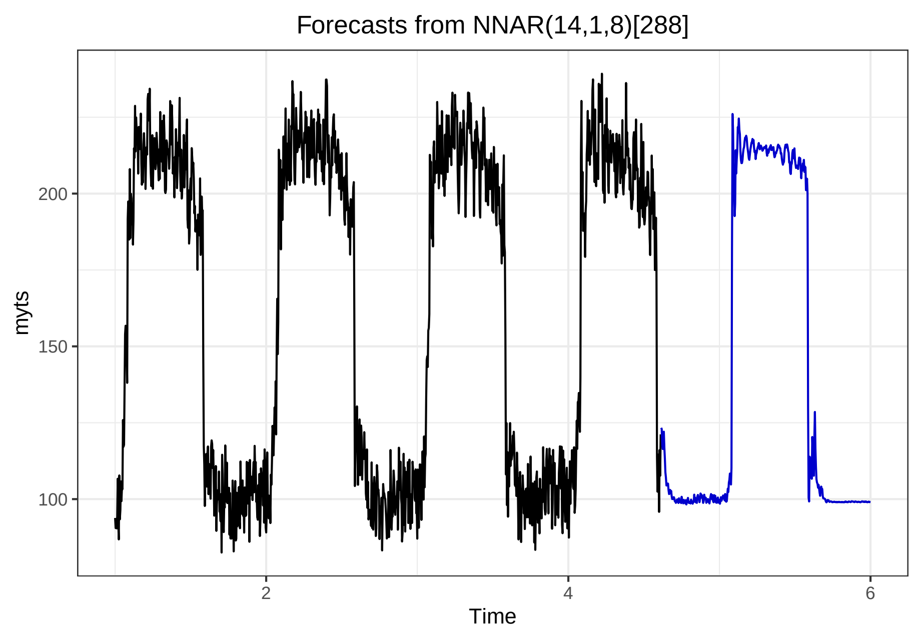
```r
myts = ts(APTelectric$watt, frequency = 288)
plot(myts)  #we see evidence of seasonality
```




```r
theme_set(theme_bw())
autoplot(myts) + ggtitle("Time Series Plot of the `APTelectricity' Time-Series") +
    theme(plot.title = element_text(hjust = 0.5))
```



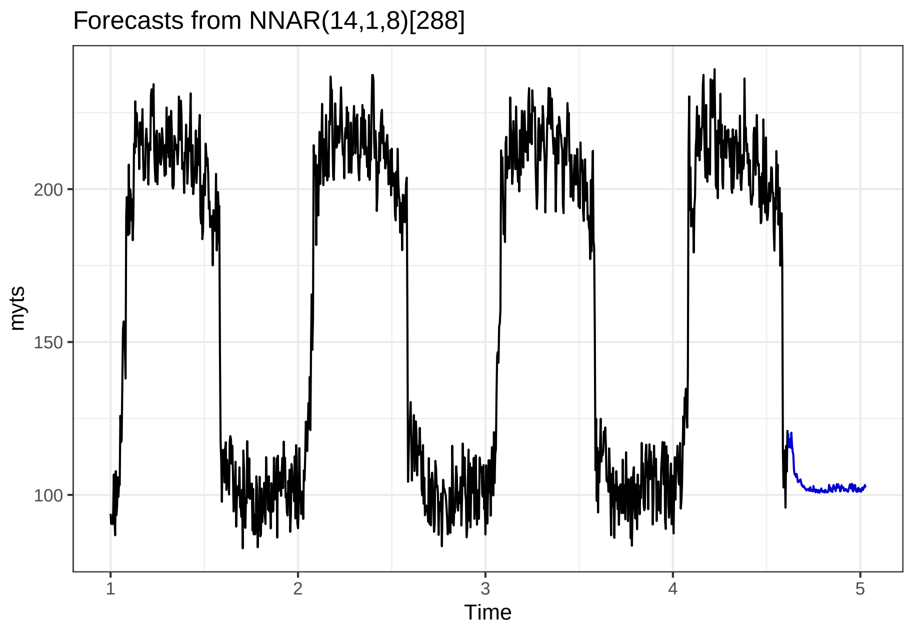
```
set.seed(34)
# nnetar() requires a numeric vector or time series object as
# input ?nnetar() can be seen for more info on the function
# nnetar() by default fits multiple neural net models and
# gives averaged results xreg option allows for only numeric
# vectors in nnetar() function
fit = nnetar(myts)
nnetforecast <- forecast(fit, h = 400, PI = F)  #Prediction intervals do not come by d
autoplot(nnetforecast) + theme(plot.title = element_text(hjust = 0.5))
```




```
## Using an external regressor in a neural net
fit2 = nnetar(myts, xreg = APTelectric$appliances)
```

For forecast of neural net models with external regressor, we need to have future values of the external regressor to be fed in the forecast function. More than one external regressors can be used in the forecast of the neural net models.

```
# Defining the vector which we want to forecast
y = rep(2, times = 12 * 10)
nnetforecast <- forecast(fit2, xreg = y, PI = F)
autoplot(nnetforecast)
```



Forecasts from NNAR(14,1,8)[288]
myts
200
150
100
1
2
3
4
5
Time