Python Code

```
import requests
from bs4 import BeautifulSoup
from datetime import datetime
import pandas as pd

years = ['1908','1912','1920','1924','1928','1932','1936','1948','1952',
         '1956','1960','1964','1968','1972','1976',
         '1980','1984','1988','1992','1996','2000',
         '2004','2008','2012','2016','2020']

res = []
for year in years:
    URL = "http://www.olympicgamesmarathon.com/olympiad{}.php".format(year)
    page = requests.get(URL)

    soup = BeautifulSoup(page.content, "html.parser")
    results = soup.findAll("table")

    #Get the raw data
    text = []
    for table in results:
        rank = table.findAll("td")

        for i in range(len(rank)):
            clean_text = rank[i].text.strip()
            if clean_text == 'Did not finish':
                break
            if clean_text != 'Results':
                text.append(clean_text)

    while '' in text:
        text.remove('')


   #Get rank and time
   raw_data = []
   for i in range(0, len(text), 5):
       try:
               raw_data.append((text[i][:-1], text[i+4]))
       except:
           pass


   #Get top 100 times
   times = []
   for rank, time in raw_data:
       times.append(time)

   times.sort()
   times = times[:100]

   tmp = []
   for item in times:
       tmp.append([year, item])

   res.append(tmp)

full_data = []
for year in res:
    for item in year:
        full_data.append(item)

df = pd.DataFrame(full_data, columns = ['Year', 'Time'])

writer = pd.ExcelWriter('test.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='welcome', index=False, encoding = 'utf-8')
writer.save()
```

R Notebook

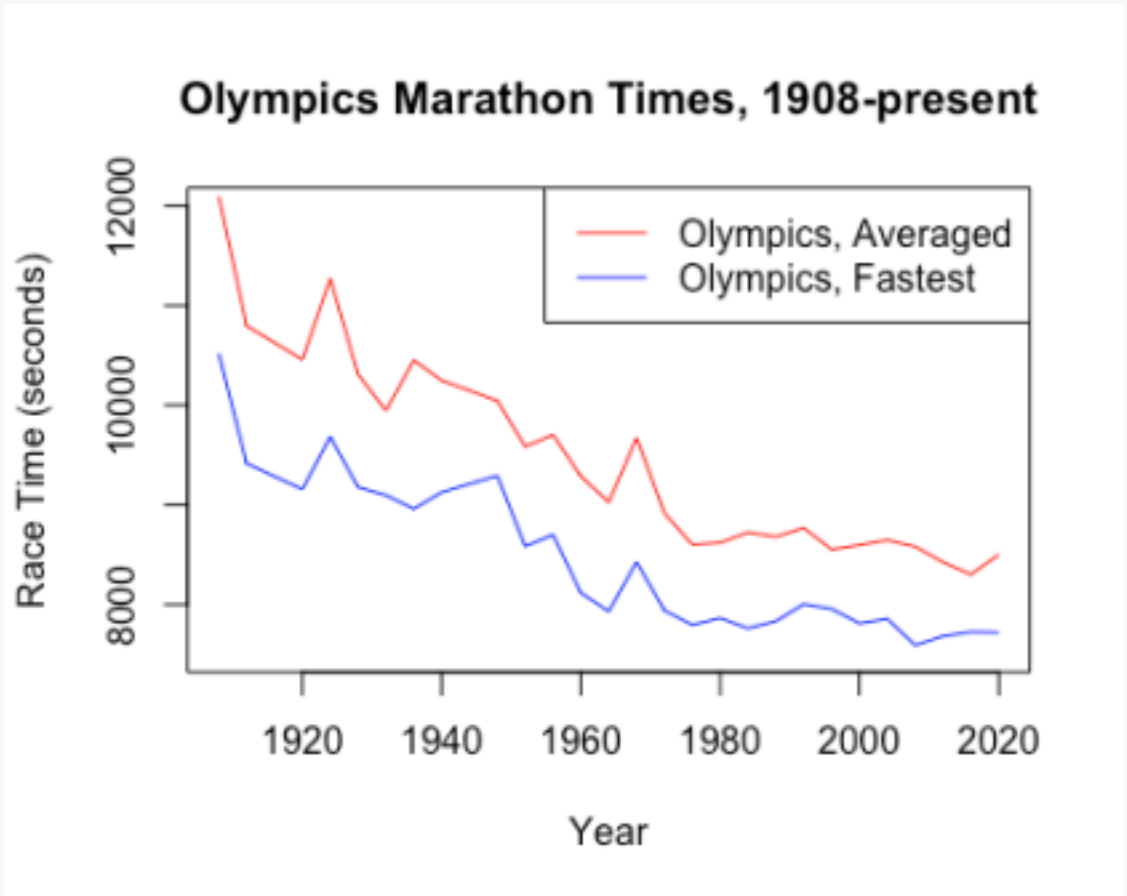
```
library(readr)
Olympics <- read_csv("Olympics time series test.csv")
head(Olympics)
```

```
## # A tibble: 6 x 3
## Year Average Fastest
## <dbl> <time> <time>
## 1 1908 03:21:34 02:55:18
## 2 1912 02:59:55 02:36:55
## 3 1916 02:57:06 02:34:45
## 4 1920 02:54:17 02:32:36
## 5 1924 03:07:49 02:41:23
## 6 1928 02:51:49 02:32:57
```

```
Olympics.ts.avg = ts(Olympics[,2], start=1908,
frequency=0.25) Olympics.ts.fast = ts(Olympics[,3],
start=1908, frequency=0.25)
```

```
plot(Olympics.ts.avg, main="Olympics Marathon Times, 1908-present",
xlab="Year", xlim=c(1908,2020), ylab="Race Time (seconds)", ylim=c(7500,
12000), type="l" ,col="red")
lines(Olympics.ts.fast, col="blue")
```

```
legend("topright", inset(-0.2,0), legend=c("Olympics, Averaged", "Olympics,
Fastest") , col=c("red", "blue"), lty=1, lwd=1)
```



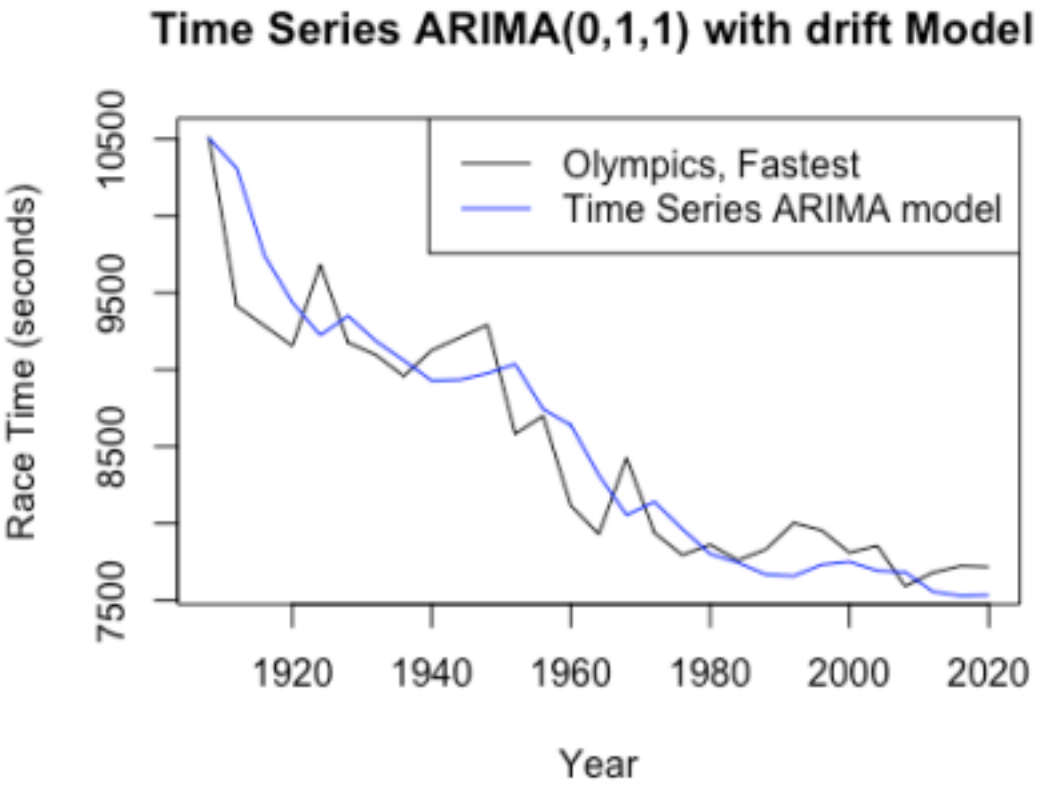

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
## method from
## as.zoo.data.frame zoo
```

```
arimafit <- auto.arima(Olympics.ts.fast)
summary(arimafit)
```

```
## Series: Olympics.ts.fast
## ARIMA(0,1,1) with drift
##
## Coefficients:
## ma1 drift
## -0.5422 -84.3461
## s.e. 0.2378 28.6445
##
## sigma^2 estimated as 104759: log likelihood=-200.7
## AIC=407.4 AICc=408.4 BIC=411.39
##
## Training set error measures:
## ME RMSE MAE MPE MAPE MASE ## Training set -24.55457 306.4674 242.3018
-0.2510558 2.821512 0.0285406
## ACF1
## Training set 0.2012937
```

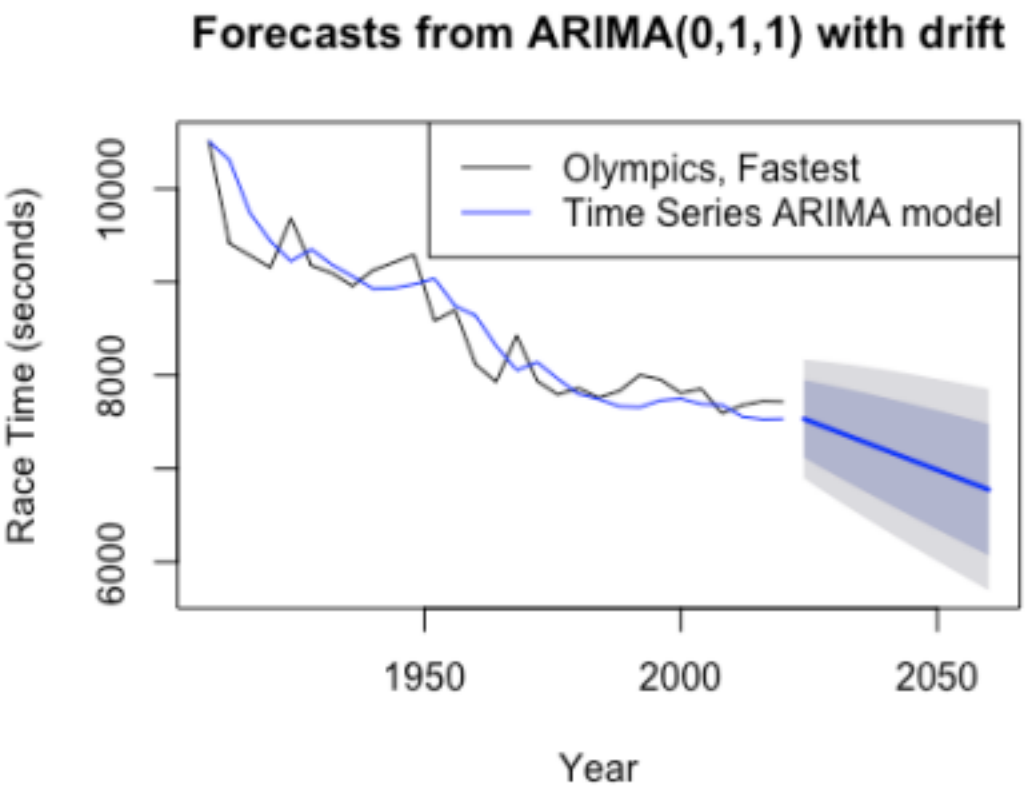
```
plot(Olympics.ts.fast, main= "Time Series ARIMA(0,1,1) with drift Model ",
xlab="Year",ylab="Race Time (seconds)")
lines(fitted(arimafit), col="blue")
legend("topright", inset(-0.2,0), legend=c("Olympics, Fastest", "Time
Series  ARIMA model") , col=c("black", "blue"), lty=1, lwd=1)
```




```
plot(forecast(arimafit), xlab="Year", ylab="Race Time
(seconds)") lines(fitted(arimafit), col="blue")
forecast(arimafit)
```

```
##  Point  Forecast  Lo  80  Hi  80  Lo  95  Hi  95  ##  2024
7533.989  7119.194  7948.783  6899.616  8168.362  ##  2028
7449.643  6993.448  7905.837  6751.954  8147.332  ##  2032
7365.296  6871.158  7859.434  6609.578  8121.015  ##  2036
7280.950  6751.581  7810.319  6471.351  8090.550  ##  2040
7196.604  6634.207  7759.001  6336.492  8056.716  ##  2044
7112.258  6518.668  7705.849  6204.440  8020.076  ##  2048
7027.912  6404.688  7651.136  6074.773  7981.051  ##  2052
6943.566  6292.054  7595.078  5947.165  7939.967  ##  2056
6859.220  6180.599  7537.841  5821.358  7897.081  ##  2060
6774.874 6070.185 7479.562 5697.146 7852.602
```

```
legend("topright", inset(-0.2,0), legend=c("Olympics, Fastest", "Time
Series  ARIMA model") , col=c("black", "blue"), lty=1, lwd=1)
```




```
# training set
Olympics.train=subset(Olympics[c(1:25),])
```

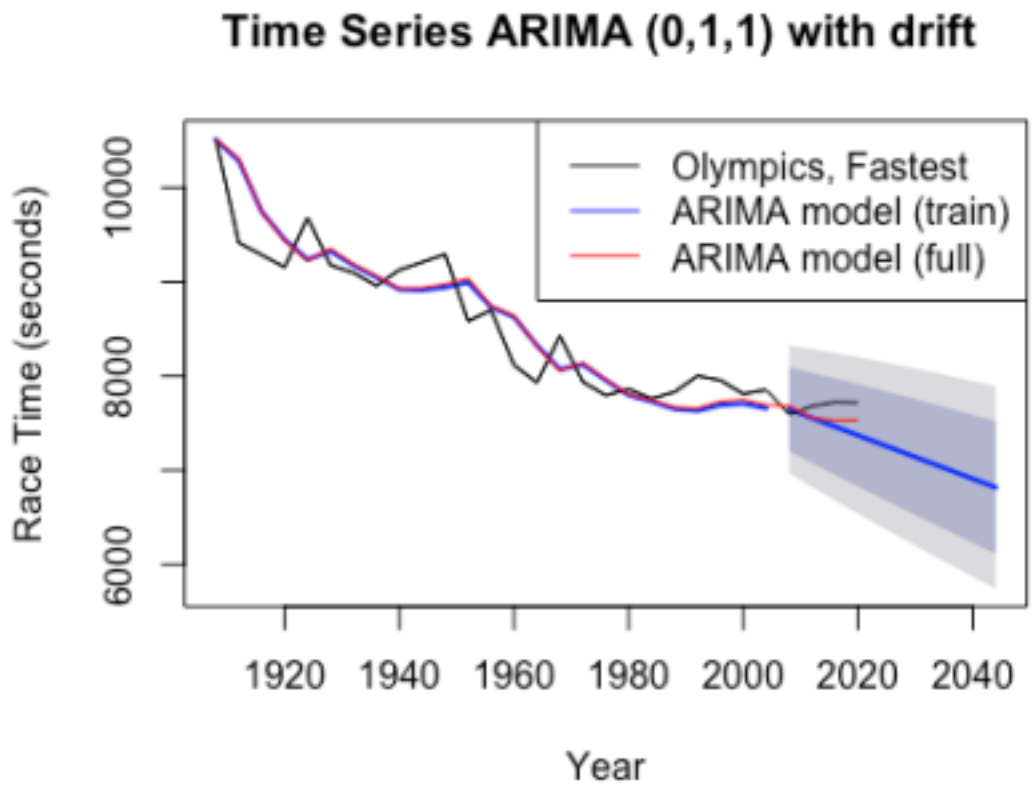
```
Olympics.ts.train = ts(Olympics.train[,3], start=1908, frequency=0.25)
```

```
library(forecast)
arimafit.train = auto.arima(Olympics.ts.train)
summary(arimafit.train)
```

```
## Series: Olympics.ts.train
## ARIMA(0,1,1) with drift
##
## Coefficients:
## ma1 drift
## -0.5887 -92.1383
## s.e. 0.3093 29.9457
##
## sigma^2 estimated as 118626: log likelihood=-173.43
## AIC=352.85 AICc=354.05 BIC=356.39
##
## Training set error measures:
## ME RMSE MAE MPE MAPE MASE  ACF1
## Training set -30.83625 323.0964 260.2571 -0.335048 3.004911 0.03019406
0.223875
```

```
plot(forecast(arimafit.train), main= "Time Series ARIMA (0,1,1) with drift
",  xlab="Year",ylab="Race Time (seconds)")
```

```
lines(fitted(arimafit.train), col="blue", lwd=2)
lines(Olympics.ts.fast)
lines(fitted(arimafit),col="red")

legend("topright", inset(-0.2,0), legend=c("Olympics, Fastest", "ARIMA
model  (train)", "ARIMA model (full)") , col=c("black", "blue", "red"),
lty=1,  lwd=1)
```




```
forecast(arimafit.train)
```

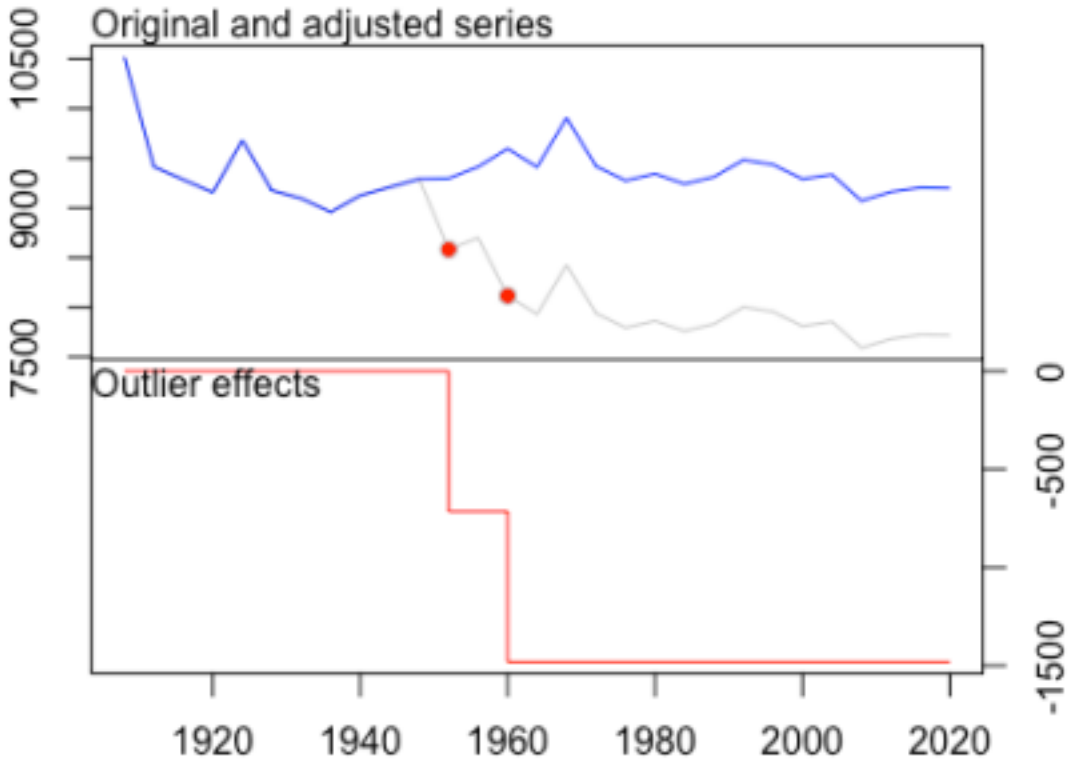
```
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2008 7647.506 7206.111 8088.900 6972.451 8322.560
## 2012 7555.367 7078.094 8032.641 6825.440 8285.295
## 2016 7463.229 6952.591 7973.868 6682.275 8244.183
## 2020 7371.091 6829.138 7913.044 6542.245 8199.936
## 2024 7278.953 6707.398 7850.507 6404.836 8153.069
## 2028 7186.814 6587.118 7786.510 6269.658 8103.970 ##
2032 7094.676 6468.100 7721.251 6136.411 8052.940 ## 2036
7002.538  6350.190  7654.886  6004.857  8000.218  ##  2040
6910.399  6233.259  7587.540  5874.802  7945.996  ##  2044
6818.261 6117.204 7519.318 5746.087 7890.435
```

```
library(tsoutliers)
Olympics.tso = tso(Olympics.ts.fast)
```

```
## Warning in locate.outliers.oloop(y = y, fit = fit, types = types, cval =
cval, :
## the first 2 residuals were set to zero
```

```
## Warning in locate.outliers.oloop(y = y, fit = fit, types = types, cval =
cval, :
## the first 2 residuals were set to zero
```

```
plot(Olympics.tso)
```




```
head(Olympics.tso)
```

```
## $outliers
## type ind time coefhat tstat
## 1 LS 12 1952 -714.5000 -3.210108
## 2 LS 14 1960 -766.3125 -3.528762
##
## $y
## Time Series:
## Start = 1908
## End = 2020
## Frequency = 0.25
## Fastest
## [1,] 10518
## [2,] 9415
## [3,] 9285
## [4,] 9156
## [5,] 9683
## [6,] 9177
## [7,] 9096
## [8,] 8959
## [9,] 9126
```

```
## [10,] 9209
## [11,] 9292
## [12,] 8583
## [13,] 8700
## [14,] 8116
## [15,] 7931
## [16,] 8426
## [17,] 7939
## [18,] 7795
## [19,] 7863
## [20,] 7761
## [21,] 7832
## [22,] 8003
## [23,] 7956
## [24,] 7811
## [25,] 7855
## [26,] 7592
## [27,] 7681
## [28,] 7724
## [29,] 7718
##
## $yadj
## Time Series:
## Start = 1908
## End = 2020
## Frequency = 0.25
## y
## [1,] 10518.000
## [2,] 9415.000
## [3,] 9285.000
## [4,] 9156.000
## [5,] 9683.000
## [6,] 9177.000
## [7,] 9096.000
## [8,] 8959.000
## [9,] 9126.000
## [10,] 9209.000
## [11,] 9292.000
## [12,] 9297.500
## [13,] 9414.500
## [14,] 9596.813
## [15,] 9411.813
## [16,] 9906.813
## [17,] 9419.813
## [18,] 9275.813
## [19,] 9343.813
## [20,] 9241.813
## [21,] 9312.813
## [22,] 9483.813
## [23,] 9436.813
## [24,] 9291.813
## [25,] 9335.813
## [26,] 9072.813
```

```
## [27,] 9161.813
## [28,] 9204.813
## [29,] 9198.813
##
## $cval
## [1] 3
##
## $fit
## Series: Olympics.ts.fast
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
## intercept LS12 LS14
## 9356.0000 -714.5000 -766.3125
## s.e. 87.3023 222.5782 217.1618
##
## sigma^2 estimated as 93512: log likelihood=-205.53
## AIC=419.06 AICc=420.73 BIC=424.53
##
## $effects
## Time Series:
## Start = 1908
## End = 2020
## Frequency = 0.25
## Fastest
## [1,] 0.000
## [2,] 0.000
## [3,] 0.000
## [4,] 0.000
## [5,] 0.000
## [6,] 0.000
## [7,] 0.000
## [8,] 0.000
## [9,] 0.000
## [10,] 0.000
## [11,] 0.000
## [12,] -714.500
## [13,] -714.500
## [14,] -1480.813
## [15,] -1480.813
## [16,] -1480.813
## [17,] -1480.813
## [18,] -1480.813
## [19,] -1480.813
## [20,] -1480.813
## [21,] -1480.813
## [22,] -1480.813
## [23,] -1480.813
## [24,] -1480.813
## [25,] -1480.813
## [26,] -1480.813
## [27,] -1480.813
## [28,] -1480.813
```

```
## [29,] -1480.813
```

```
City.ts <- read_csv("City marathon ts.csv")
```

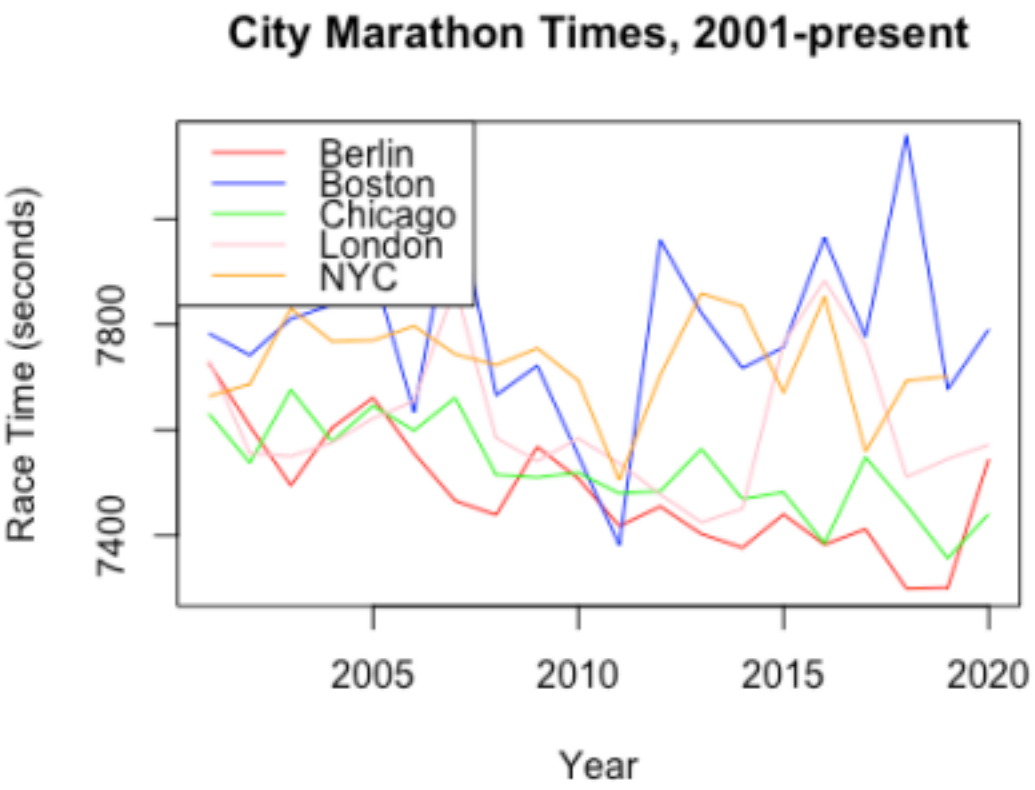
```
## Parsed with column
specification: ## cols(
## City = col_character(),
## Year = col_double(),
## `Fastest Time` = col_time(format = "")
## )
```

```
head(City.ts)
```

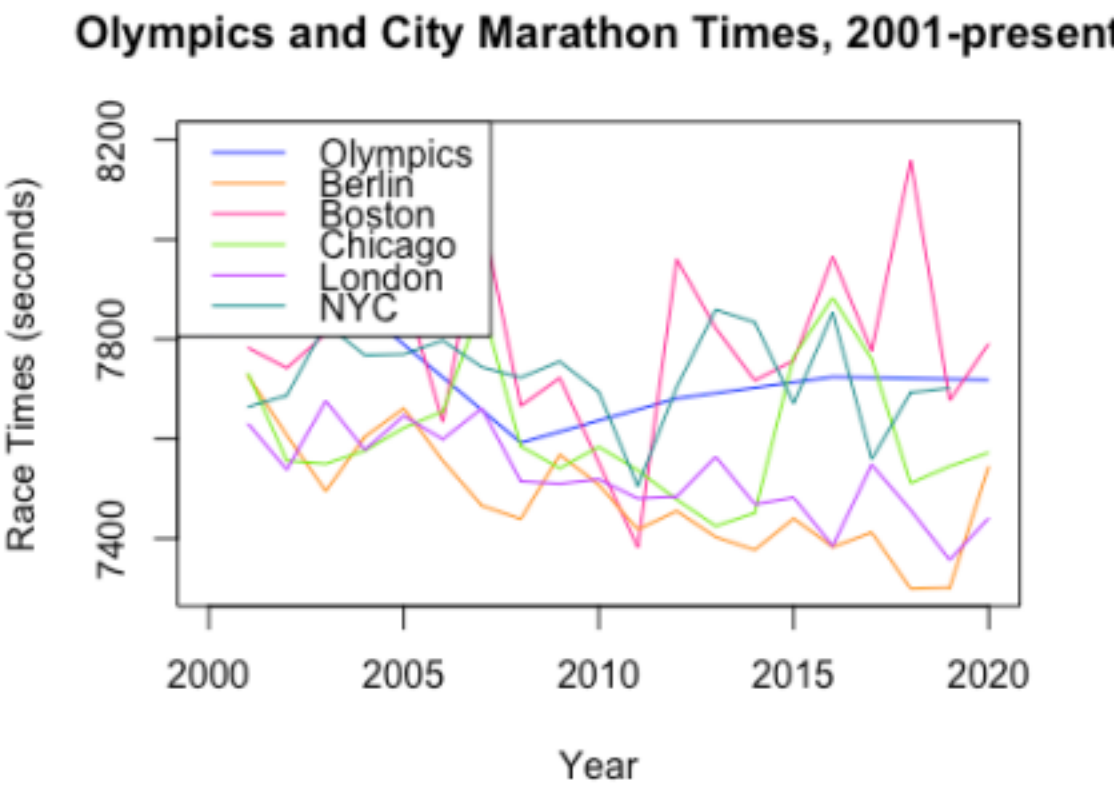
```
## # A tibble: 6 x 3
## City Year `Fastest Time`
## <chr> <dbl> <time>
## 1 Berlin 2001 02:08:47
## 2 Berlin 2002 02:06:47
## 3 Berlin 2003 02:04:55
## 4 Berlin 2004 02:06:44
## 5 Berlin 2005 02:07:41
## 6 Berlin 2006 02:05:56
```

```
Berlin = subset(City.ts, City=="Berlin")
Boston = subset(City.ts, City=="Boston")
London = subset(City.ts, City=="London")
Chicago = subset(City.ts,
City=="Chicago")
NYC = subset(City.ts, City=="NYC")
Berlin.ts=ts(Berlin[,3], start=2001, frequency=1)
Boston.ts=ts(Boston[,3], start=2001, frequency=1)
London.ts=ts(London[,3], start=2001, frequency=1)
Chicago.ts=ts(Chicago[,3], start=2001, frequency=1)
NYC.ts=ts(NYC[,3], start=2001, frequency=1)
```

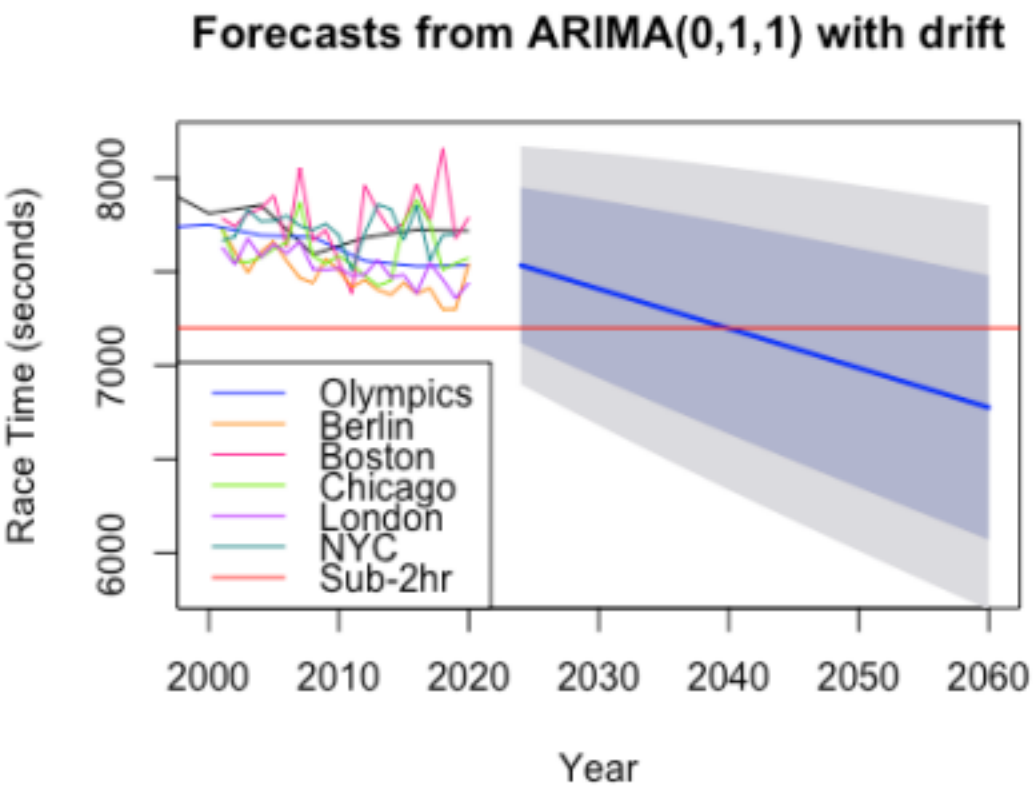
```
plot(Berlin.ts, main="City Marathon Times, 2001-present", xlab="Year",
ylab="Race Time (seconds)", ylim=c(7300, 8150), type="l" ,col="red")
lines(Boston.ts, col="blue")
lines(London.ts, col="green")
lines(Chicago.ts, col="pink")
lines(NYC.ts, col="orange ")
par(xpd=NA, mar=par()$mar+c(-4,0,-1,1))
legend("topleft", inset(-0.2,0), legend=c("Berlin", "Boston", "Chicago",
"London", "NYC") , col=c("red", "blue", "green", "pink", "orange"), lty=1,
lwd=1)
```



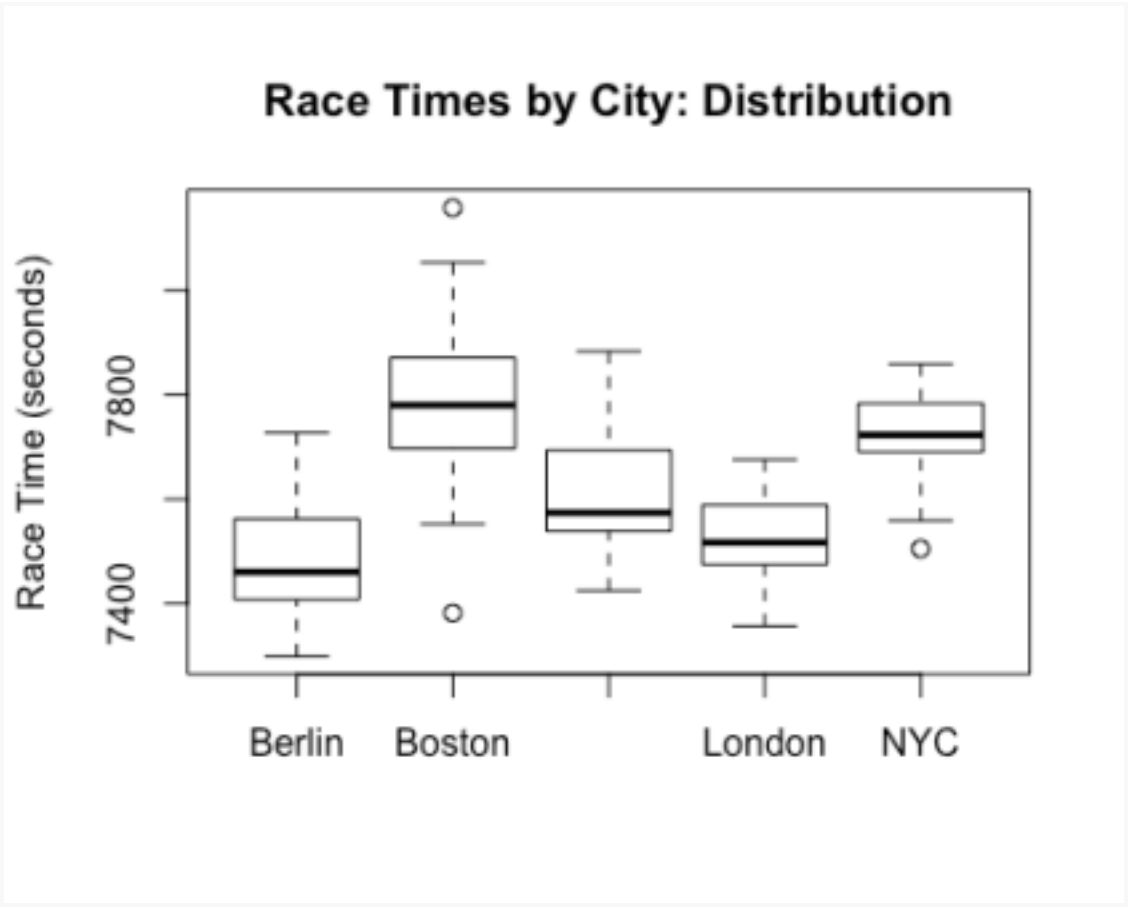

```
plot(Olympics.ts.fast, main="Olympics and City Marathon Times,
2001-present",  xlab="Year", xlim=c(2000,2020), ylab="Race Times (seconds)",
ylim=c(7300,  8200), type="l" ,col="blue")
lines(Boston.ts, col="deeppink")
lines(Berlin.ts, col="darkorange")
lines(Chicago.ts, col="chartreuse2")
lines(London.ts, col="darkorchid1")
lines(NYC.ts, col="darkcyan")

par(xpd=NA, mar=par()$mar+c(-4,0,-1,1))
legend("topleft", inset(-0.2,0), legend=c("Olympics", "Berlin", "Boston",
"Chicago", "London", "NYC") , col=c("blue", "darkorange", "deeppink",
"chartreuse2", "darkorchid1", "darkcyan"), lty=1, lwd=1)
```




```
plot(forecast(arimafit), xlim=c(2000,2060), ylim=c(5800, 8200),
xlab="Year",  ylab="Race Time (seconds)")

lines(fitted(arimafit), col="blue")
lines(Boston.ts, col="deeppink")
lines(Berlin.ts, col="darkorange")
lines(Chicago.ts, col="chartreuse2")
lines(London.ts, col="darkorchid1")
lines(NYC.ts, col="darkcyan")

abline(h=7200, col="red", lwd=1)

par(xpd=NA, mar=par()$mar+c(-4,0,-1,1))
legend("bottomleft", inset(-0.2,0), legend=c("Olympics", "Berlin", "Boston",
"Chicago", "London", "NYC", "Sub-2hr") , col=c("blue", "darkorange",
"deeppink", "chartreuse2", "darkorchid1", "darkcyan", "red"), lty=1,
lwd=1)
```




```
boxplot(Berlin$`Fastest Time`, Boston$`Fastest Time`, Chicago$`Fastest
Time`,  London$`Fastest Time`, NYC$`Fastest Time`, names=c("Berlin",
"Boston",  "Chicago", "London", "NYC") , main="Race Times by City:
Distribution",  ylab="Race Time (seconds)" )
```



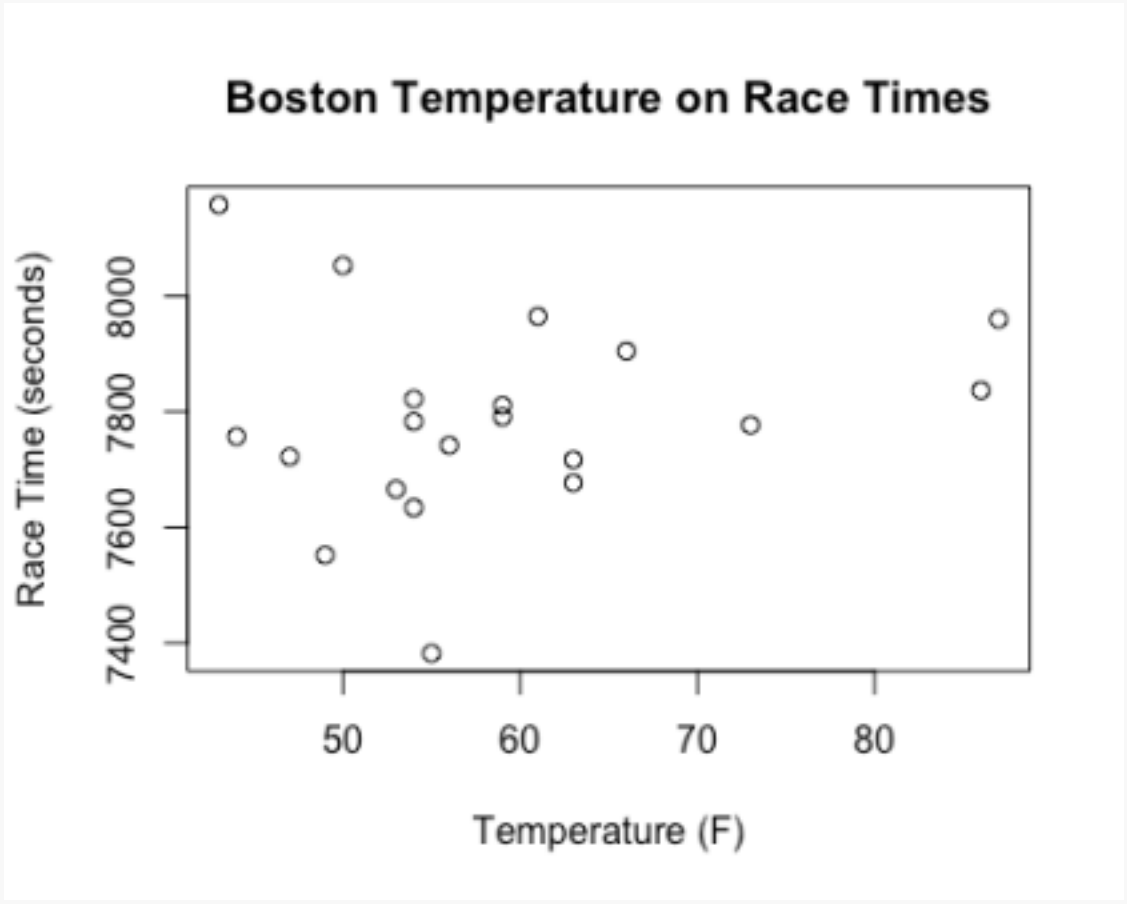

```
library(readr)
Boston_temp <- read_csv("Boston temp.csv")
```

```
## Parsed with column specification:
## cols(
## Temperature = col_double(),
## Time = col_time(format = ""),
## Seconds = col_double()
## )
```

```
head(Boston_temp)
```

```
## # A tibble: 6 x 3
## Temperature Time Seconds
## <dbl> <time> <dbl>
## 1 55 02:03:02 7382
## 2 49 02:05:52 7552
## 3 54 02:07:14 7634
## 4 53 02:07:46 7666
## 5 63 02:07:57 7677
## 6 63 02:08:37 7717
```

```
plot(Boston_temp$Temperature, Boston_temp$Time, main="Boston Temperature on
Race Times", xlab="Temperature (F)", ylab="Race Time (seconds)")
```




```
abline
```

```
## function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
## coef = NULL, untf = FALSE, ...)
## {
## int_abline <- function(a, b, h, v, untf, col = par("col"),  ##
lty = par("lty"), lwd = par("lwd"), ...)
 .External.graphics(C_abline,
## a, b, h, v, untf, col, lty, lwd, ...)
## if (!is.null(reg)) {
## if (!is.null(a))
## warning("'a' is overridden by 'reg'")
## a <- reg
## }
## if (is.object(a) || is.list(a)) {
## p <- length(coefa <- as.vector(coef(a)))
## if (p > 2)
## warning(gettextf("only using the first two of %d regression
coefficients",
## p), domain = NA)
## islm <- inherits(a, "lm")
## noInt <- if (islm)
## !as.logical(attr(stats::terms(a), "intercept"))
## else p == 1
## if (noInt) {
## a <- 0
## b <- coefa[1L]
## }
```

```
## else {
## a <- coefa[1L]
## b <- if (p >= 2)
## coefa[2L]
## else 0
## }
## }
## if (!is.null(coef)) {
## if (!is.null(a))
## warning("'a' and 'b' are overridden by 'coef'") ## a <-
coef[1L]
## b <- coef[2L]
## }
## int_abline(a = a, b = b, h = h, v = v, untf = untf, ...)
## invisible()
## }
## <bytecode: 0x7fc1ca7e08f0>
## <environment: namespace:graphics>
```

```
modBosttemp=lm(Seconds~Temperature,
data=Boston_temp) summary(modBosttemp)
```

```
##
## Call:
## lm(formula = Seconds ~ Temperature, data =
Boston_temp) ##
## Residuals:
## Min 1Q Median 3Q Max
## -396.81 -84.32 0.45 60.39 400.47
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)  ##
(Intercept) 7681.261 202.403 37.950 <2e-16 *** ##
Temperature 1.774 3.376 0.525 0.606  ## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 176.6 on 18 degrees of freedom ##
Multiple R-squared: 0.0151, Adjusted R-squared: -0.03962  ##
F-statistic: 0.276 on 1 and 18 DF, p-value: 0.6058
```

```
London_Gender <- read_csv("London Gender.csv")
```

```
## Warning: Missing column names filled in: 'X4' [4]
## Parsed with column specification:
## cols(
## Year = col_double(),
## Time = col_time(format = ""),
## Gender = col_character(),
## X4 = col_logical()
## )
```

```
head(London_Gender)
```

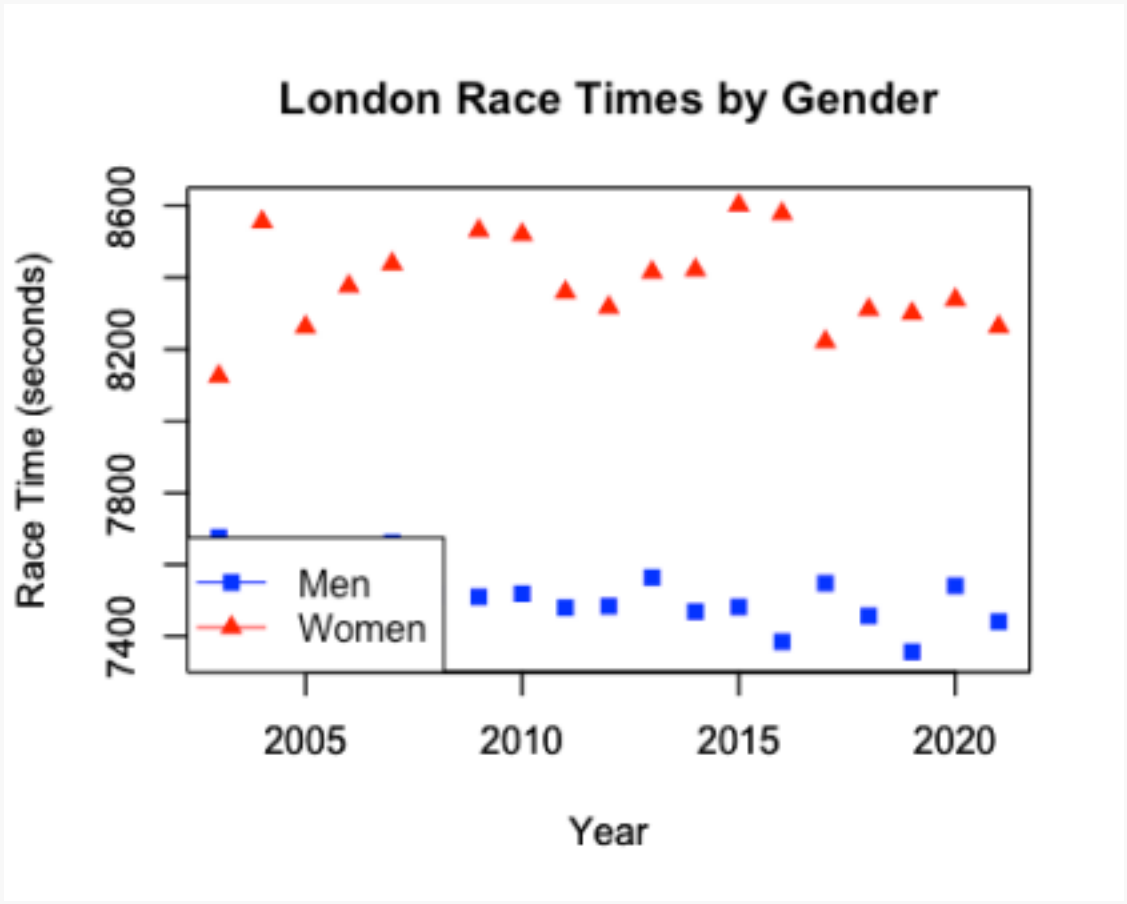
```
## # A tibble: 6 x 4
## Year Time Gender X4
```

```
## <dbl> <time> <chr> <lgl>
## 1 2003 02:15:25 F NA
## 2 2004 02:22:35 F NA
## 3 2005 02:17:42 F NA
## 4 2006 02:19:36 F NA
## 5 2007 02:20:38 F NA
## 6 2009 02:22:11 F NA
```

```
LondonM = subset(London_Gender, Gender=="M")
LondonF = subset(London_Gender, Gender=="F")
```

```
plot(LondonM$Year, LondonM$Time, pch=15, col="blue", ylim=c(7350,8600),
xlab="Year", ylab="Race Time (seconds)", main="London Race Times by
Gender") par(new=TRUE)
plot(LondonF$Year, LondonF$Time, ylim=c(7350,8600), xlab="Year", ylab="Race
Time (seconds)", pch=17, col="red")
```

```
legend("bottomleft", inset(-0.2,0), legend=c("Men", "Women") ,
col=c("blue",  "red"), pch=c(15, 17), lty=1, lwd=1)
```




```
library(readr)
Chicago_Age <- read_csv("Chicago Age.csv")
```

```
## Parsed with column specification:
## cols(
## Year = col_double(),
## Time = col_time(format = ""),
## Age = col_character()
## )
```

```r
head(Chicago_Age)
```

```
## # A tibble: 6 x 3
## Year Time Age
## <dbl> <time> <chr>
## 1 2014 02:09:08 16-19
## 2 2000 02:16:01 20-24
## 3 2001 02:08:52 20-24
## 4 2002 02:13:57 20-24
## 5 2003 02:20:05 20-24
## 6 2004 02:16:07 20-24
```

```r
View(Chicago_Age)
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout =
TRUE):
## running command ''/usr/bin/otool' -L
'/Library/Frameworks/R.framework/Resources/
## modules/R_de.so'' had status 1
```

```r
unique(Chicago_Age[c("Age")])
```

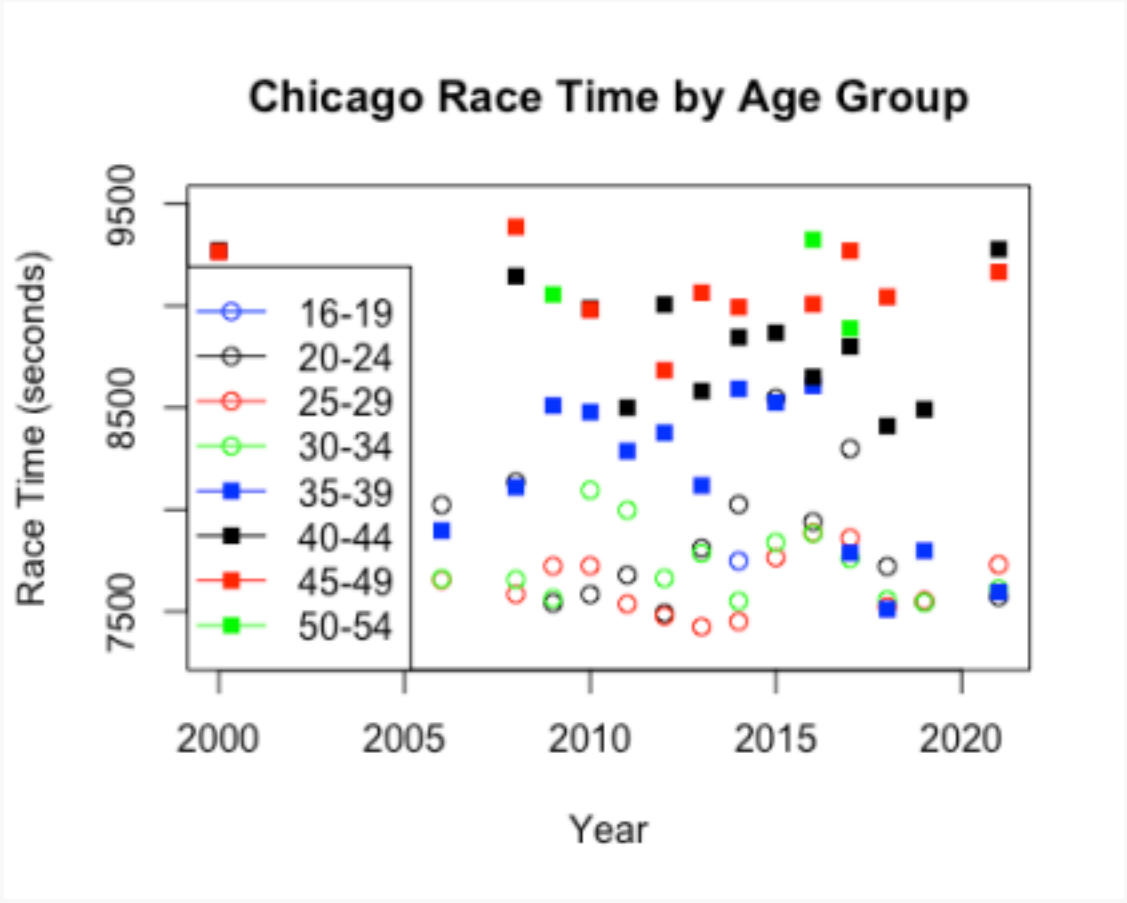
```
## # A tibble: 9 x 1
## Age
## <chr>
## 1 16-19
## 2 20-24
## 3 25-29
## 4 30-34
## 5 35-39
## 6 40-44
## 7 45-49
## 8 50-54
## 9 <NA>
```

```r
Chicago16.19 = subset(Chicago_Age, Age=="16-19")
Chicago20.24 = subset(Chicago_Age, Age=="20-24")
Chicago25.29 = subset(Chicago_Age, Age=="25-29")
Chicago30.34 = subset(Chicago_Age, Age=="30-34")
Chicago35.39 = subset(Chicago_Age, Age=="35-39")
Chicago40.44 = subset(Chicago_Age, Age=="40-44")
Chicago45.49 = subset(Chicago_Age, Age=="45-49")
Chicago50.54 = subset(Chicago_Age, Age=="50-54")
```

```r
plot(Time~Year, data=Chicago20.24, ylim=c(7300, 9500), ylab="Race Time
(seconds)", main="Chicago Race Time by Age Group")
points(Time~Year, data=Chicago25.29, col="red")
points(Time~Year, data=Chicago16.19, col="blue")
points(Time~Year, data=Chicago30.34, col="green")
points(Time~Year, data=Chicago35.39, col="blue", pch=15)
points(Time~Year, data=Chicago40.44, col="black", pch=15)
points(Time~Year, data=Chicago45.49, col="red", pch=15)
points(Time~Year, data=Chicago50.54, col="green", pch=15)
```

```r
legend("bottomleft", inset(-0.2,0), legend=c("16-19", "20-24", "25-29", "30-
```

```
34", "35-39", "40-44", "45-49", "50-54") , col=c("blue", "black", "red",
"green", "blue", "black", "red", "green"), pch=c(1, 1, 1, 1, 15, 15, 15,
15),  lty=1, lwd=1)
```



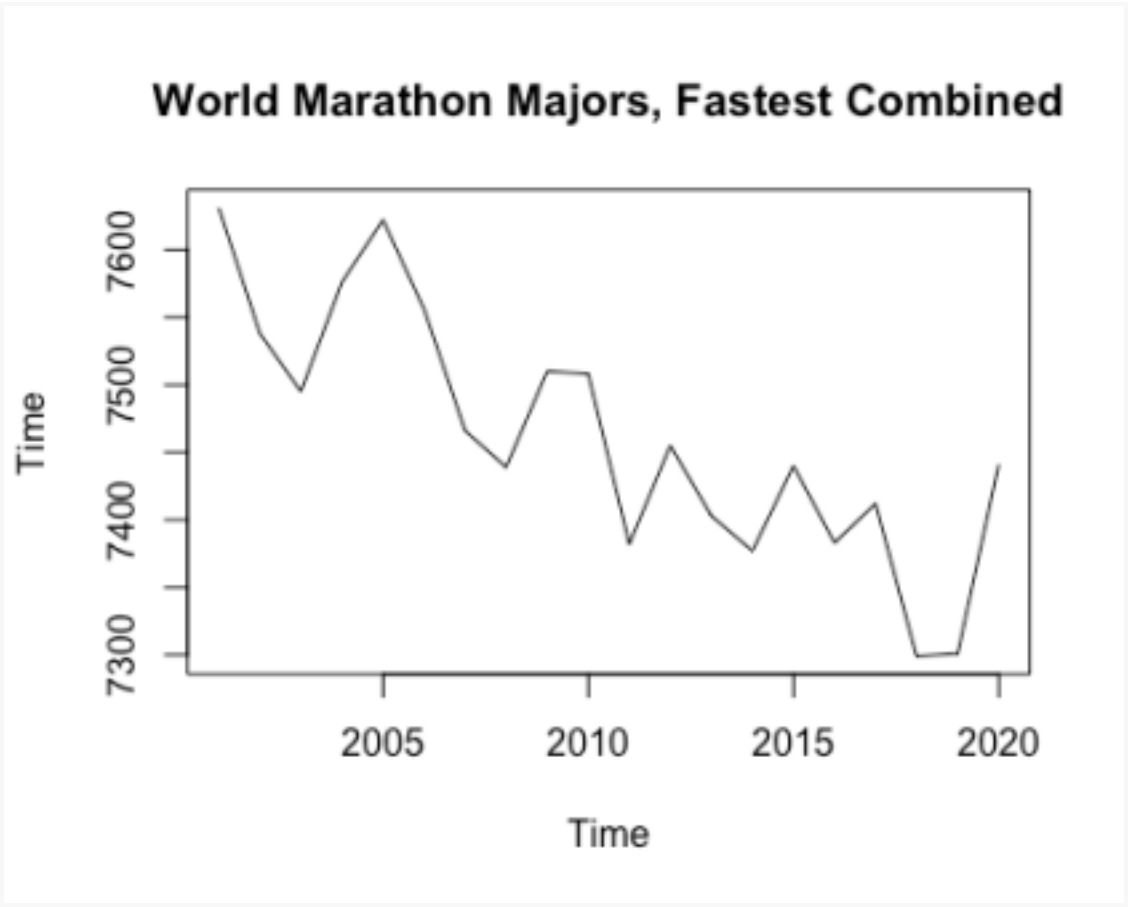

```
WMM.data <- read_csv("World Marathon Majors.csv")
```

```
## Parsed with column specification:
## cols(
## Year = col_double(),
## Time = col_time(format = "")
## )
```

```
head(WMM.data)
```

```
## # A tibble: 6 x 2
## Year Time
## <dbl> <time>
## 1 2001 02:07:11
## 2 2002 02:05:38
## 3 2003 02:04:55
## 4 2004 02:06:16
## 5 2005 02:07:02
## 6 2006 02:05:56
```

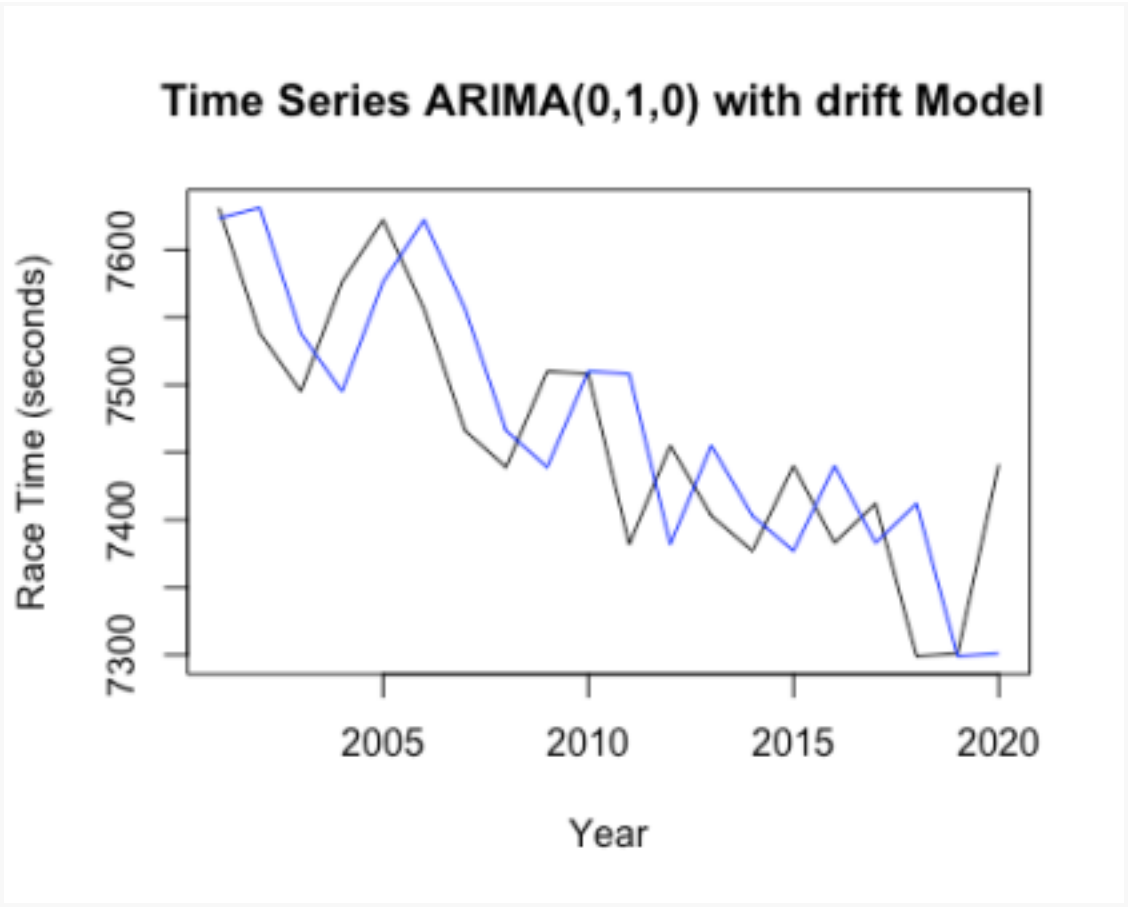
```
WMM.ts = ts(WMM.data[,2], start=2001, frequency=1)
plot(WMM.ts, main="World Marathon Majors, Fastest Combined")
```




```
library(forecast)
arimafit.wmm <- auto.arima(WMM.ts)
summary(arimafit.wmm)
```

```
## Series: WMM.ts
## ARIMA(0,1,0)
##
## sigma^2 estimated as 5394: log likelihood=-108.59
## AIC=219.18 AICc=219.41 BIC=220.12
##
## Training set error measures:
## ME RMSE MAE MPE MAPE MASE ## Training set -9.11845 71.5822 60.38155
-0.1256865 0.8101054 0.9560412 ## ACF1
## Training set -0.1763578
```

```
plot(WMM.ts, main= "Time Series ARIMA(0,1,0) with drift Model ",
xlab="Year",ylab="Race Time (seconds)")
lines(fitted(arimafit.wmm), col="blue")
```


Time Series ARIMA(0,1,0) with drift Model
Race Time (seconds)
7300
7400
7500
7600
2005
2010
2015
2020
Year