Final Data Management Project

Christina Chang

12/8/2017

1. Data Preparation

Load Packages

```r
# read data
library(foreign)
library(readr)

# clean & manipulate data
library(tidyverse)
library(dplyr)
library(reshape2)

# plots
library(ggplot2)

# time series
library(tseries)
library(dynlm)
library(urca)

# forecast
library(forecast)
library(scales)

# knit
library(knitr)
library(rvest)
```

Import Data

By visual inspection of the file, we don't want R to read the first two rows. NA argument to specify that any blanks ("") or asterisks ("*") would be considered missing data. Note: the asterisks are specified to be redacted information, based on the UNHCR website.

```
setwd("/Users/Berlin/Desktop/HertieDataScience/final project")

d1 <- read_csv("unhcr_popstats_people of concern.csv", skip = 2, na = c("","*"))
d2 <- read_csv("unhcr_popstats_refugee status.csv", skip = 2, na = c("", "*"))

View(d1)
View(d2)
```

Tidy Data

The UNHCR data is an unbalanced panel dataset. After tidying the data, we have created a balanced panel dataset with all entities (countries) observed in all years (from 2000 to 2016).

Steps to clean and manipulate data:

```
str(d1)
# Year correctly structured as integer
# Country destination and origin correctly structured as character
# All other variables should be numeric

d1[4:11] <- lapply(d1[4:11], as.numeric)

summary(d1) # years from 1951 to 2016

str(d2)
# Year correctly structured as integer
# Country destination and origin correctly structured as character
# Do not need RSD procedure type information
# All other variables should be numeric

d2[4] <- NULL

d2[4:13] <- lapply(d2[4:13], as.numeric)

summary(d2) # years from 2000 to 2016

df <- merge(d1, d2, by = c("Year", "Country / territory of asylum/residence", "Origin"))

View(df)
str(df)
summary(df) # after merge, data before the year 2000 drops out.

# identify missing data
apply(df,2, function(x) sum(is.na(x)))
```

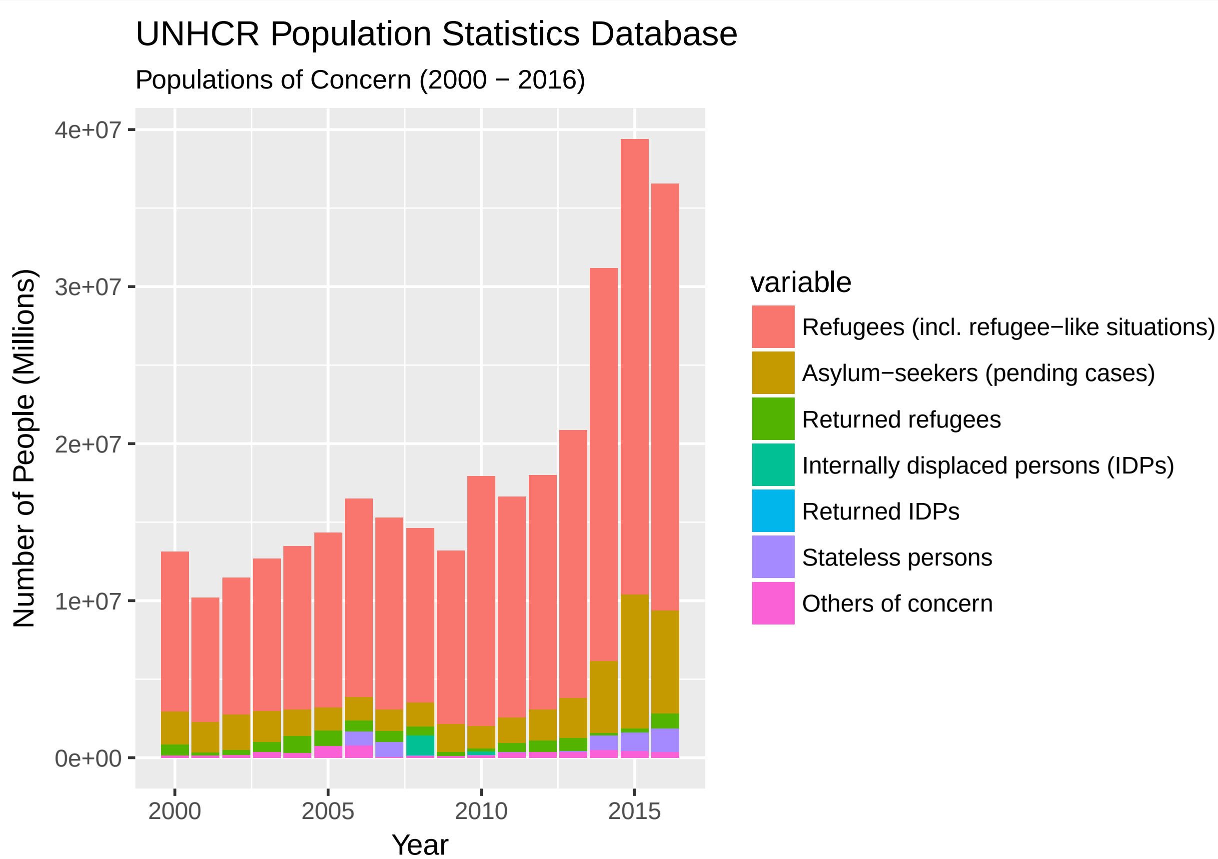
2. Exploratory Data Analysis

People of Concern:

```r
# subset for only PoC category counts by year

PoC_count <- df[c(1,4:10)]

PoC_count <- melt(PoC_count, id=c("Year"))

str(PoC_count)
```

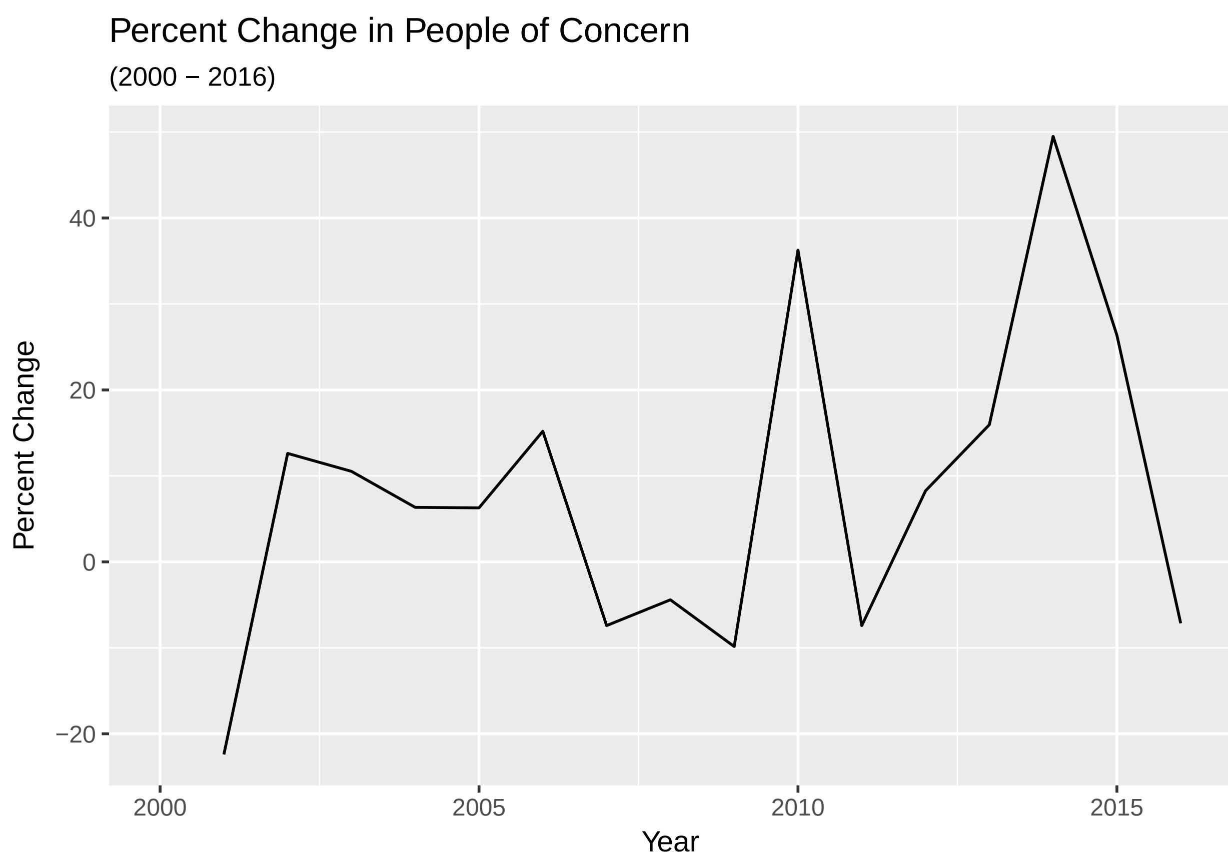
```
## 'data.frame':    813456 obs. of  3 variables:
##  $ Year    : int  2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ variable: Factor w/ 7 levels "Refugees (incl. refugee-like situations)",..: 1 1 1 1 1 1 1 1 1 1 .
##  $ value   : num  NA NA 9 507 2 5 NA 1 5 20 ...
```

```r
plot1 <- ggplot(PoC_count,aes(Year,value, na.rm = TRUE)) +
  geom_bar(aes(fill=variable),stat="identity") +
  labs(title="UNHCR Population Statistics Database",
       subtitle="Populations of Concern (2000 - 2016)",
       x="Year",
       y="Number of People (Millions)")

plot1
```



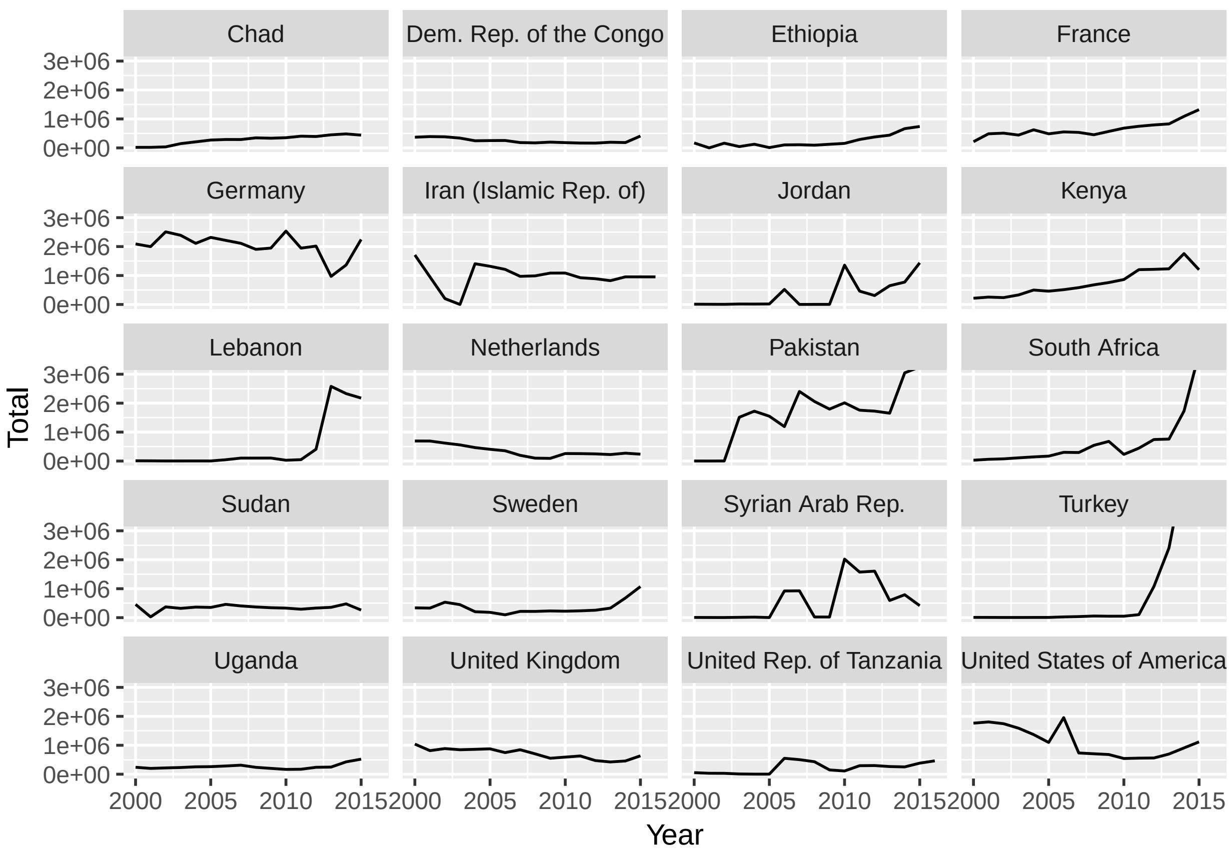
Percent Change in Total Population by "People of Concern"

```
Year_Pop <- aggregate(df$`Total Population`, by=list(Year = df$Year), FUN=sum, na.rm = TRUE)

Year_Pop$rate <- NA

Year_Pop$rate[which(Year_Pop$Year>2000)] = 100*(diff(Year_Pop$x)/Year_Pop[-nrow(Year_Pop),]$x)

View(Year_Pop)

plot2 <- ggplot(Year_Pop, aes(x= Year, y= rate)) + geom_line() +
  labs(title="Percent Change in People of Concern",
       subtitle="(2000 - 2016)",
       x="Year",
       y="Percent Change")

plot2
```



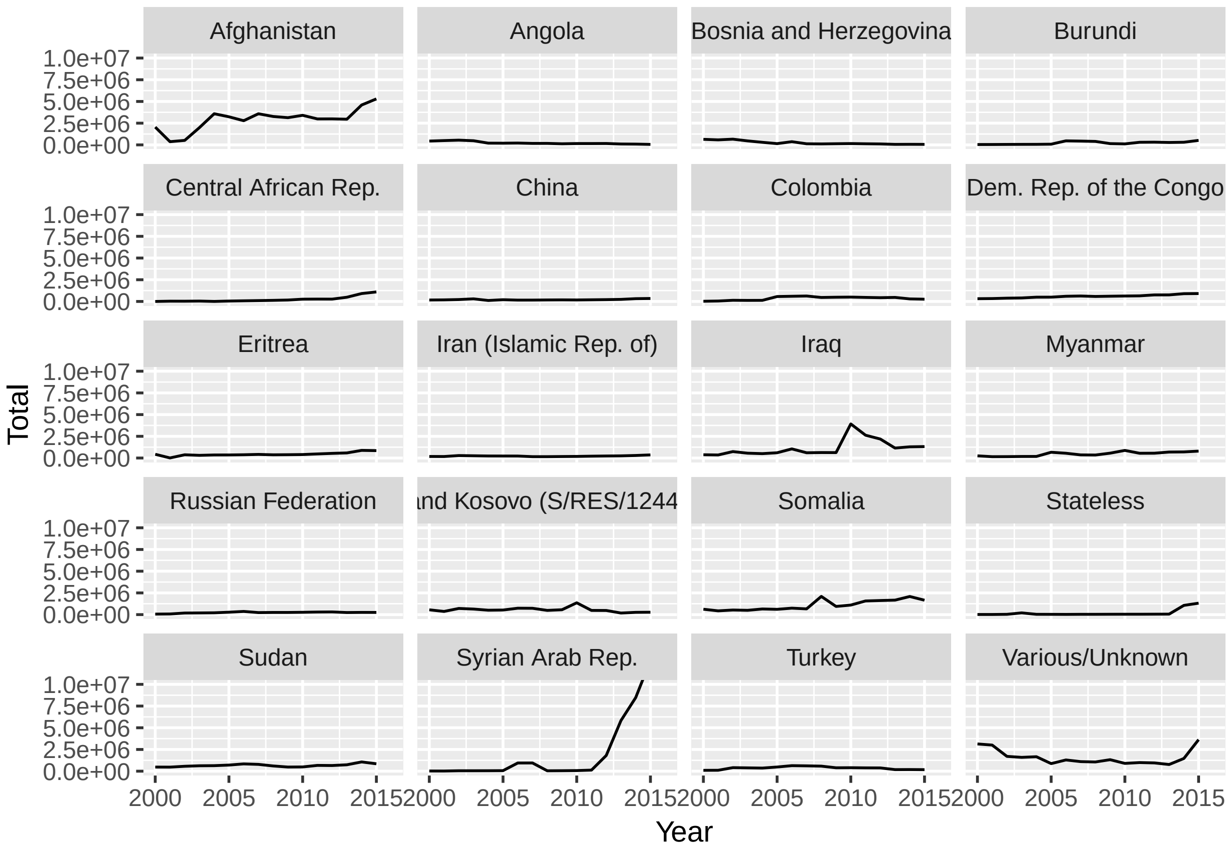
Top Countries of Destination

```
destination_country_total <- df %>%
  group_by(`Country / territory of asylum/residence`, Year) %>%
  summarise(Total = sum(`Total Population`))

View(destination_country_total)

top_destcountries <- destination_country_total %>%
  group_by(`Country / territory of asylum/residence`) %>%
  summarise(Total = sum(Total, na.rm = TRUE)) %>%
  top_n(20)

View(top_destcountries)

top_destcountries2 <- as.character(top_destcountries$`Country / territory of asylum/residence`)

plot3 <- destination_country_total %>%
  filter(`Country / territory of asylum/residence` %in% top_destcountries2) %>%
  ggplot(mapping = aes(x = Year, y = Total)) +
  geom_line() + coord_cartesian(ylim = c(0, 3e6)) +
  facet_wrap(~`Country / territory of asylum/residence`, ncol=4)

plot3
```



Top Countries of Origin

```
origin_country_total <- df %>%
  group_by(Origin, Year) %>%
  summarise(Total = sum(`Total Population`))

top_origcountries <- origin_country_total %>%
  group_by(Origin) %>%
  summarise(Total = sum(Total, na.rm = TRUE)) %>%
  top_n(20)

top_origcountries2 <- as.character(top_origcountries$Origin)

plot4 <- origin_country_total %>%
  filter(Origin %in% top_origcountries2) %>%
  ggplot(mapping = aes(x = Year, y = Total)) +
  geom_line() + coord_cartesian(ylim = c(0, 1e7)) +
  facet_wrap( ~ Origin, ncol=4)

plot4
```



3. Time Series Analysis

We run a time-series analysis to see if the total number of "Persons of Concern" (POC) in world effects the total number of POCs in Germany over time. y = Total PoC in Germany; x = Total PoC in the world; t = Years (2000 to 2016)

Prepare data for Time Series analysis

```
# create new dataframe
Germany_Poc <- df %>% group_by(`Country / territory of asylum/residence`, Year) %>%
  filter('Germany'  %in% `Country / territory of asylum/residence`) %>%
  summarise(German_Total = sum(`Total Population`, na.rm = TRUE))

View(Germany_Poc)

df_ts <- merge(Germany_Poc, Year_Pop, by = "Year")

View(df_ts)

# declare variables to be time series using ts()
df_ts$Year <- ts(df_ts$Year)
df_ts$German_Total<- ts(df_ts$German_Total)
df_ts$x <- ts(df_ts$x)

# run preliminary OLS model
summary(m1 <- dynlm(German_Total ~ x, data = df_ts))
```

```
##
## Time series regression with "ts" data:
## Start = 1, End = 17
##
## Call:
## dynlm(formula = German_Total ~ x, data = df_ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1211808  -170365    16838   237008  1383368
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.862e+06  3.388e+05   5.494 6.17e-05 ***
## x           1.533e-02  1.662e-02   0.922    0.371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 579400 on 15 degrees of freedom
## Multiple R-squared:  0.05367,    Adjusted R-squared:  -0.009417
## F-statistic: 0.8507 on 1 and 15 DF,  p-value: 0.3709
```

Results of OLS model (m1): Positive and substantially small coefficient, but not statistically significant.

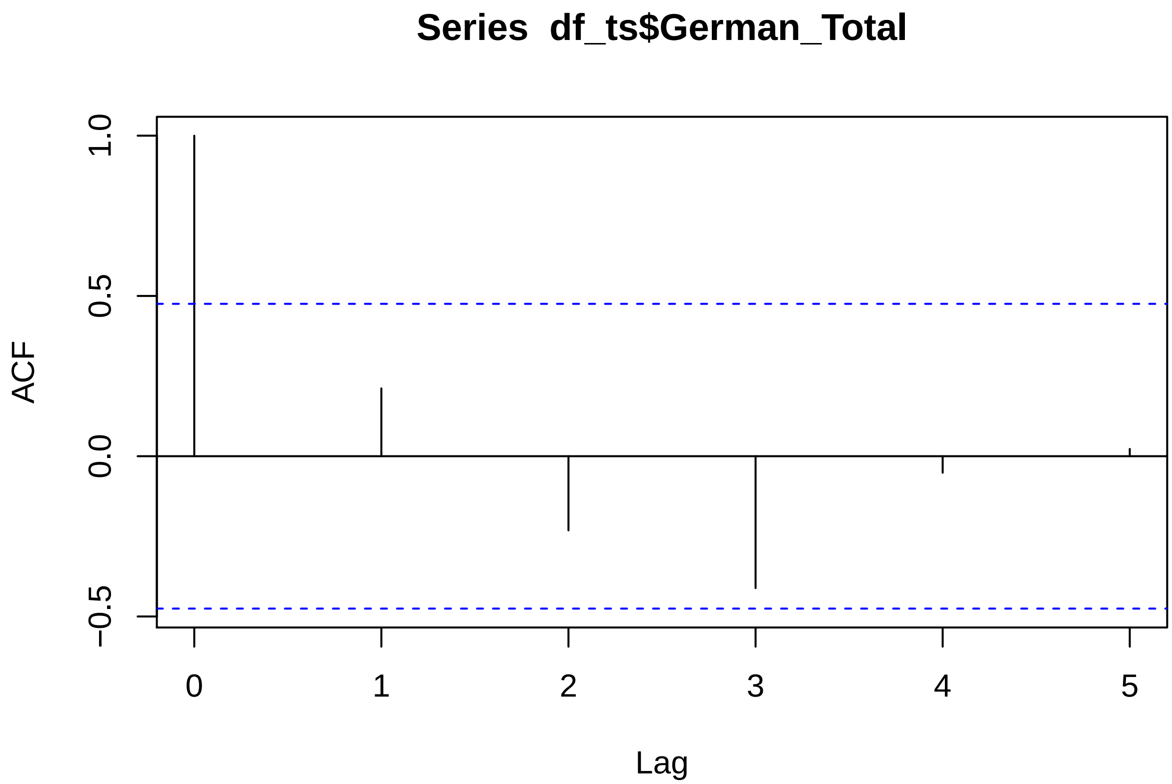
Can only use OLS regression with time series data if the following two conditions are met:

(a) Weak Dependence / Weak Persistence

```
summary(dynlm(German_Total ~ L(German_Total, 1), data = df_ts))
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 17
##
## Call:
## dynlm(formula = German_Total ~ L(German_Total, 1), data = df_ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1167925  -222002  -102511   197473  1559245
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.184e+06  7.852e+05   1.508    0.154
## L(German_Total, 1) 4.728e-01  3.777e-01   1.252    0.231
##
## Residual standard error: 584500 on 14 degrees of freedom
## Multiple R-squared:  0.1006, Adjusted R-squared:  0.03641
## F-statistic: 1.567 on 1 and 14 DF,  p-value: 0.2312
```

```
# The rho is less than 1, so stability condition is met

acf(df_ts$German_Total, na.action = na.pass, lag.max = 5)
```




```
# Correlation coefficient is statistically insignificant after 1 lag, so it is not persistent
```

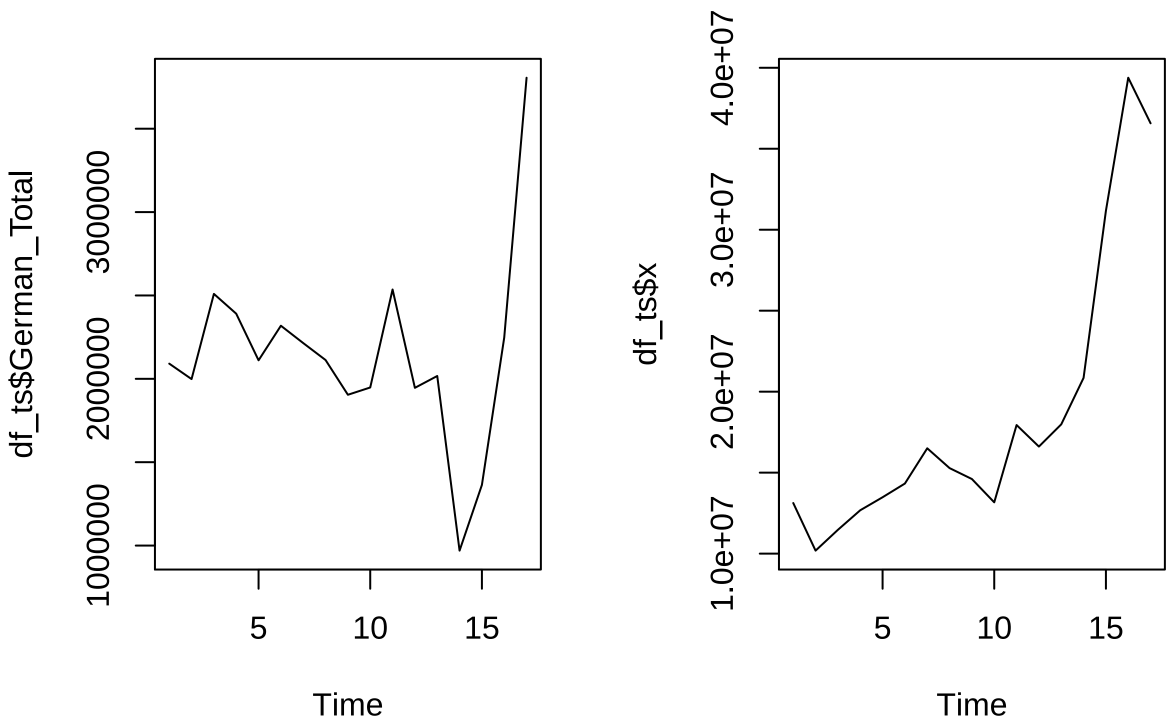
Conclusion: The data is weakly dependent allowing for a dynamically complete model.

(b) Stationarity

```r
# Unit Root - Dickey Fuller Test
adf.test(df_ts$German_Total)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df_ts$German_Total
## Dickey-Fuller = -3.9925, Lag order = 2, p-value = 0.02352
## alternative hypothesis: stationary
```

```r
# p-value is less than .05, so it has no unit root

# Trends
par(mfrow = c(1, 2))
plot(df_ts$German_Total) #Total POCs in Germany
plot(df_ts$x) #Total POCs in the world
```

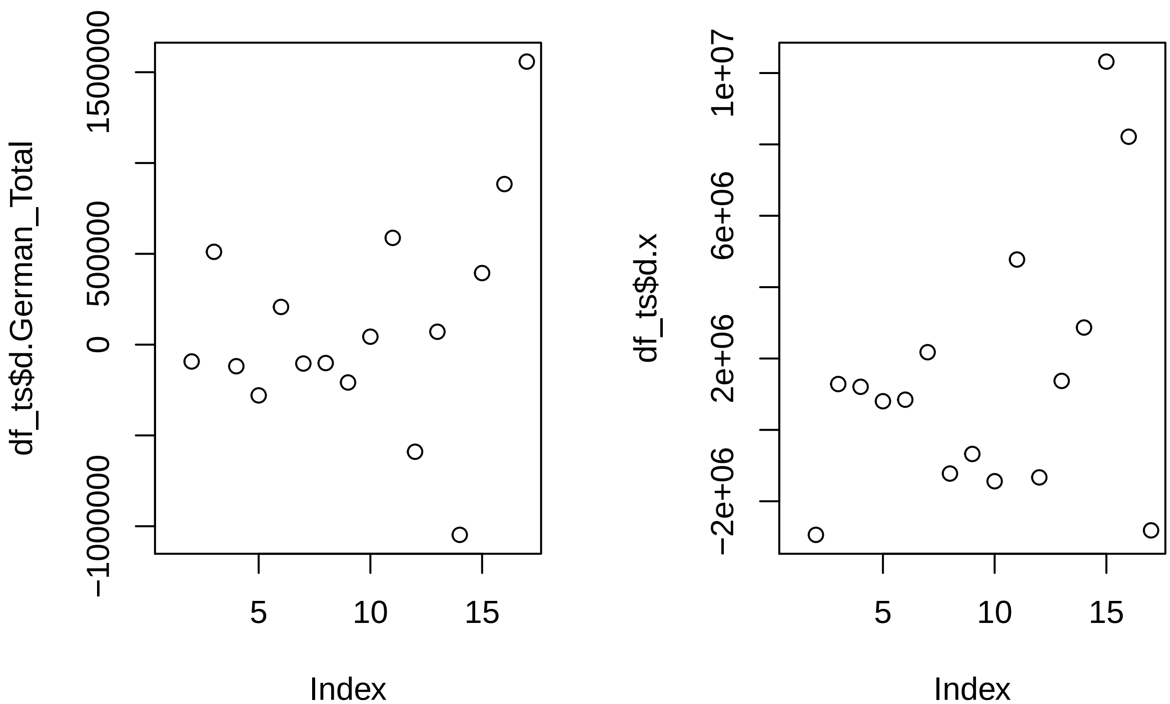


Conclusions: Total POCs in Germany looks like a stochastic (inconsistent) trend. Total POCs in the world looks like a deterministic trend. That is, it is non-stationary.

Need to account for this trend in total POCs in the world before running an OLS regression.

Method 1: First Differencing

```r
df_ts$d.German_Total <- c(NA, diff(df_ts$German_Total))
df_ts$d.x <- c(NA, diff(df_ts$x))

par(mfrow = c(1, 2))
plot(df_ts$d.German_Total)
plot(df_ts$d.x)
```




```r
summary(m2 <- dynlm(d.German_Total ~ d.x, data = df_ts))
```

```
##
## Time series regression with "numeric" data:
## Start = 1, End = 16
##
## Call:
## dynlm(formula = d.German_Total ~ d.x, data = df_ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1194561  -237301   -53873   184974  1573241
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.535e+04  1.660e+05   0.394    0.700
## d.x         2.854e-02  4.299e-02   0.664    0.518
##
## Residual standard error: 614200 on 14 degrees of freedom
##   (1 observation deleted due to missingness)
```

```
## Multiple R-squared:  0.03051,    Adjusted R-squared:  -0.03874
## F-statistic: 0.4406 on 1 and 14 DF,  p-value: 0.5176
```

Results of OLS model (m2): Positive and substantially small coefficient, but not statistically significant.

The problem with first differencing is that we lose statistical power as we lose observations.

Method 2: Detrending

```
fit1 <- lm(German_Total ~ Year, df_ts)
df_ts$resid.German_Total <- residuals(fit1)

fit2 <- lm(x ~ Year, df_ts)
df_ts$resid.x <- residuals(fit2)

summary(m3 <- dynlm(resid.German_Total ~ resid.x, data = df_ts))
```

```
##
## Time series regression with "numeric" data:
## Start = 1, End = 17
##
## Call:
## dynlm(formula = resid.German_Total ~ resid.x, data = df_ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -975522 -208415   51245  174882 1348800
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.412e-11  1.330e+05   0.000    1.000
## resid.x     4.494e-02  2.743e-02   1.638    0.122
##
## Residual standard error: 548500 on 15 degrees of freedom
## Multiple R-squared:  0.1518, Adjusted R-squared:  0.09523
## F-statistic: 2.684 on 1 and 15 DF,  p-value: 0.1222
```

Results of OLS model (m3): Positive and substantially small coefficient, but not statistically significant

So even after detrending, there is no statistically significant coefficient to show a causal effect between the total POC in the world and the total POCs in Germany over time.

4. Forecasting

As we have monthly data on asylum-seekers in Germany, we can use it to predict future numbers of asylum-seekers using a forecasting model.

Import Data

By visual inspection of the file, we don't want R to read the first two rows.

```
df3 <- read_csv("unhcr_popstats_export_asylum_seekers_monthly_2017_12_04_203715.csv", skip = 2)

View(df3)
str(df3)
summary(df3)
```

Tidy Data

In the forecasting model, NA values returns errors. As such, we specify that any NA values are assigned a value of 0.

```
df3[5] <- lapply(df3[5], as.numeric)

apply(df3,2, function(x) sum(is.na(x)))

df3$Value[is.na(df3$Value)] <- 0
```

Declare variables as time series

```
Germany_Total.Monthly <- df3 %>%
  group_by(`Country / territory of asylum/residence`, Year, Month) %>%
  summarise(Total = sum(Value))

Germany_monthly <- ts(Germany_Total.Monthly$Total,
                      start = c(1999, 1), frequency = 12)
```

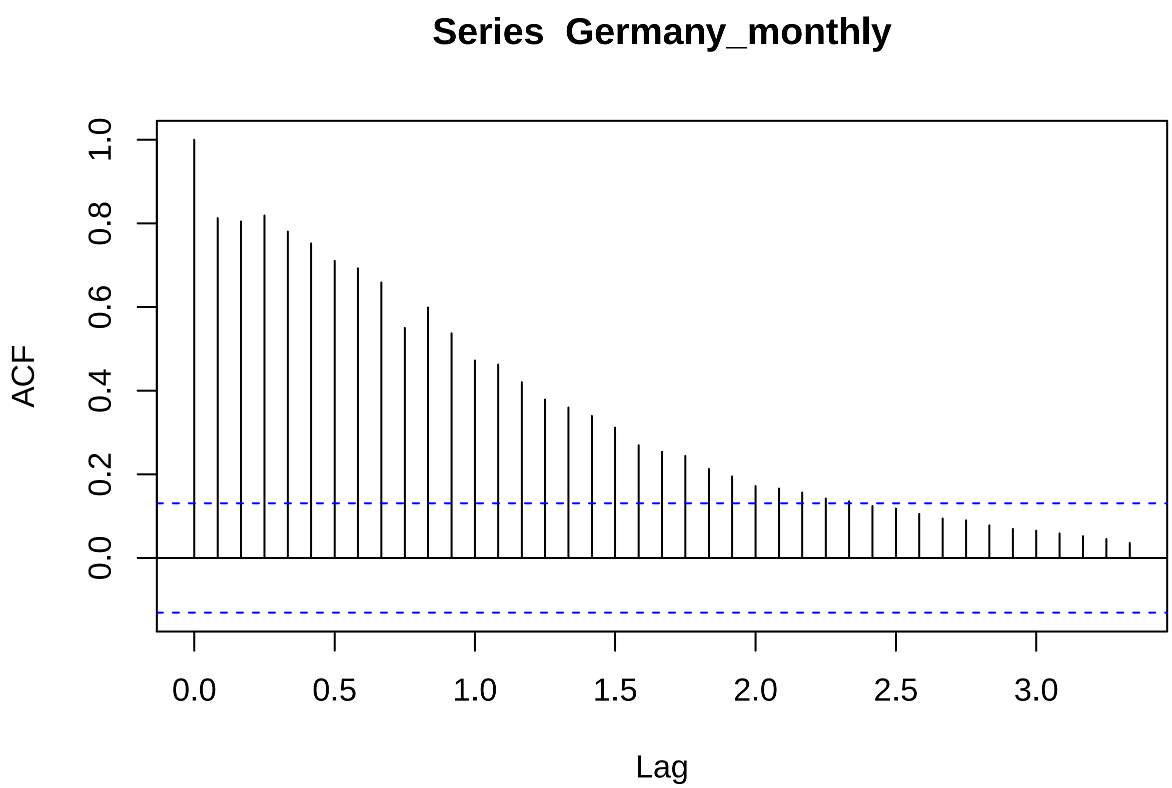
Test for weak dependence (weak persistence):

```
summary(dynlm(Germany_monthly ~ L(Germany_monthly, 1)))
```

```
##
## Time series regression with "ts" data:
## Start = 1999(2), End = 2017(9)
##
## Call:
## dynlm(formula = Germany_monthly ~ L(Germany_monthly, 1))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -57837  -1677  -1197     32  55483
##
```

```
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1919.7596   722.1649   2.658  0.00842 **
## L(Germany_monthly, 1)     0.8129     0.0391  20.792  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9061 on 222 degrees of freedom
## Multiple R-squared:  0.6607, Adjusted R-squared:  0.6592
## F-statistic: 432.3 on 1 and 222 DF,  p-value: < 2.2e-16
```

```r
# The rho is less than 1, so stability condition is met

acf(Germany_monthly, na.action = na.pass, lag.max = 40)
```



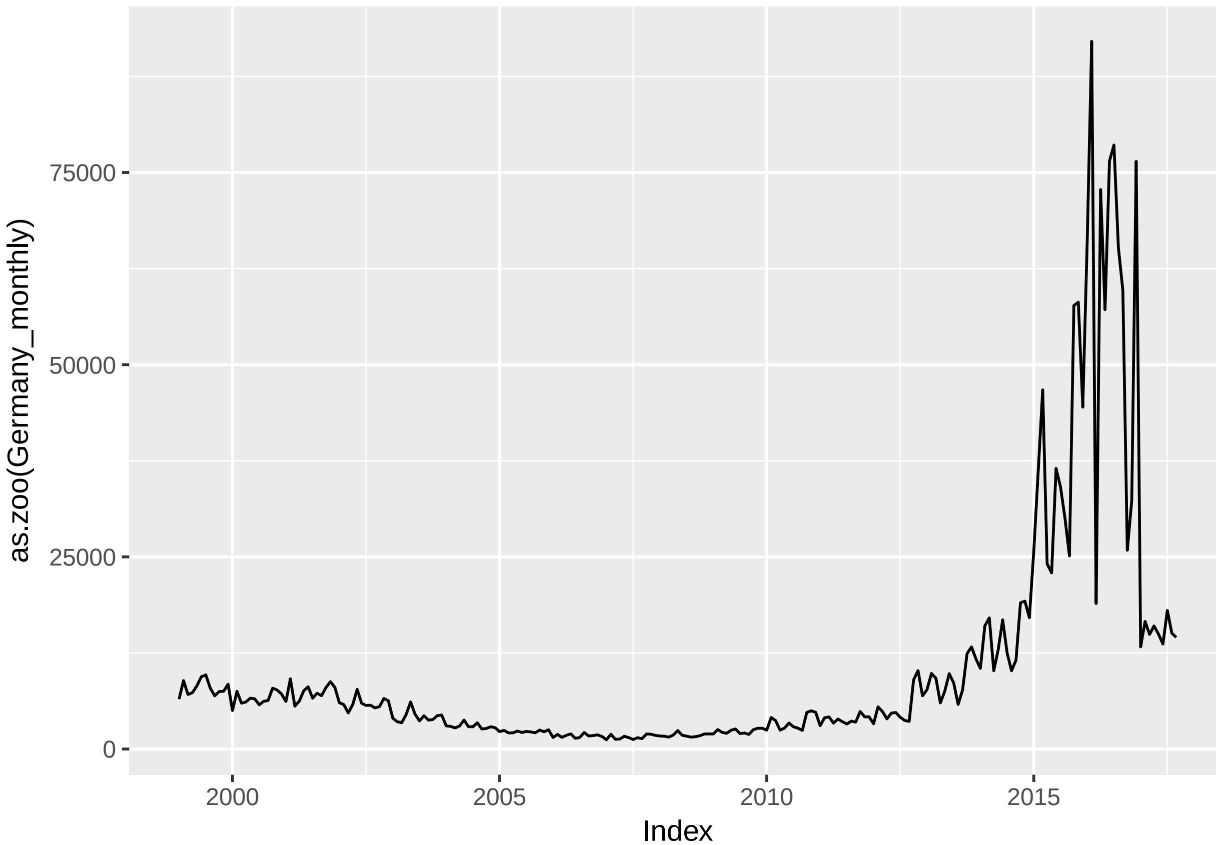

```r
# Correlation coefficient is statistically insignificant after 2.5 lag, so it is not persistent
```

Test for stationarity

```r
# Unit Root - Dickey Fuller Test
adf.test(Germany_monthly)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Germany_monthly
## Dickey-Fuller = -2.4282, Lag order = 6, p-value = 0.3961
## alternative hypothesis: stationary
```

```r
# p value is less than .05, there is no unit root.

# Trends
autoplot(as.zoo(Germany_monthly), geom = "line")
```

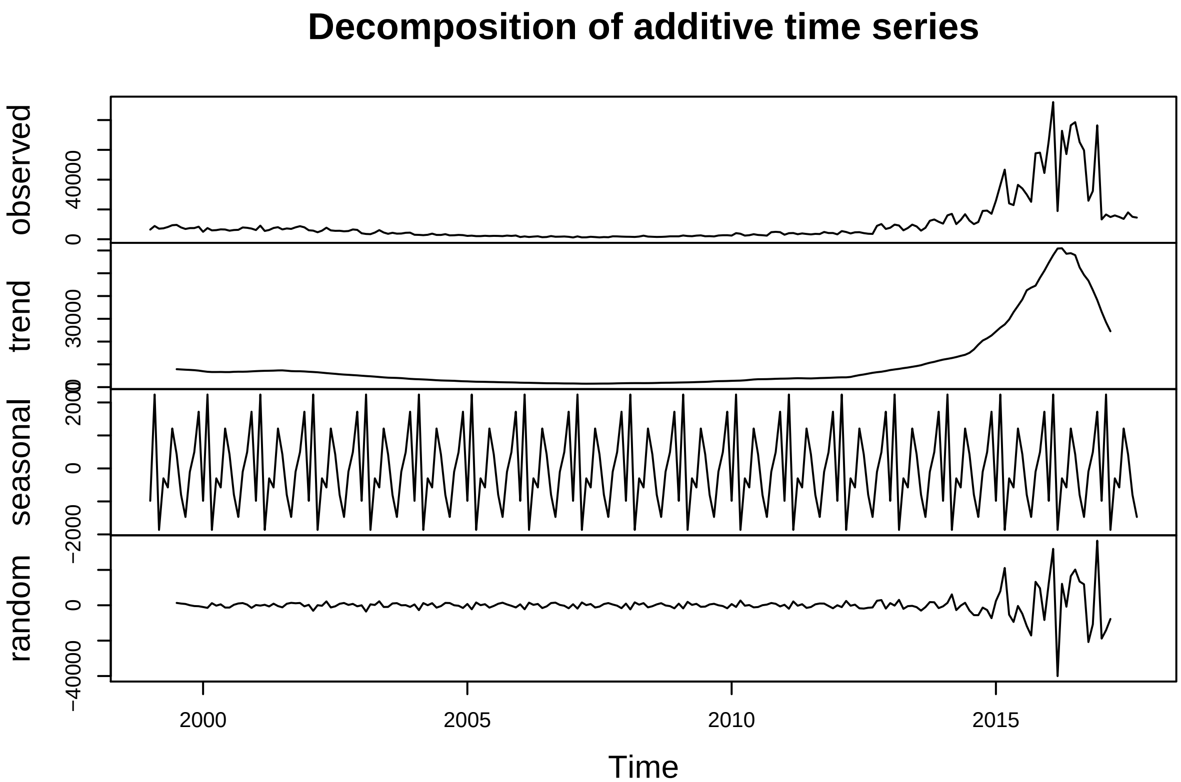



```r
# For forecasting, just observe the trend.
# Clear spike since 2015 that seems to have dropped off in 2016.
```

Decompose

Decompose the additives of time series. This returns estimates of the seasonal component, trend component and irregular components ("random" components).

```r
plot(decompose(Germany_monthly))
```



Seasonal Changes

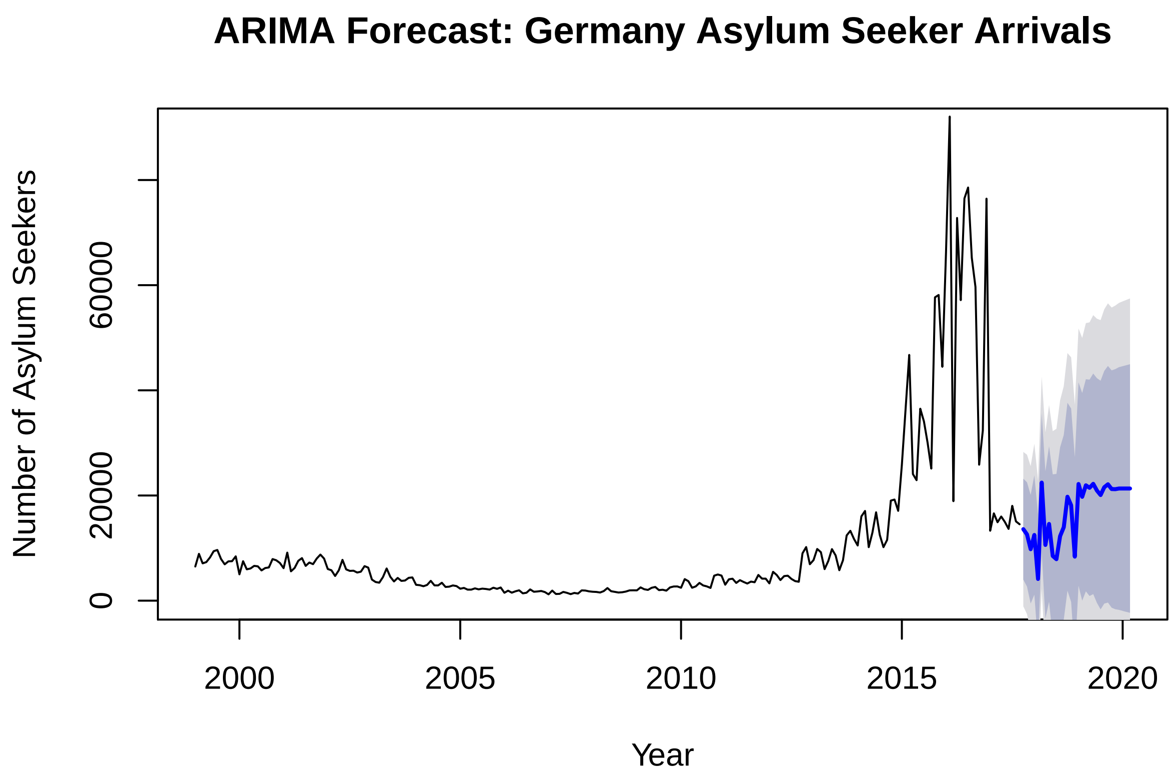
Look more closely at the seasonal changes in the number of asylum seekers.

```
stl(Germany_monthly, s.window="periodic")
# Germany has had a positive net flow of asylum seekers in February, June, July, November and December.
```

We attempt two methods for forecasting future monthly flows of asylum seekers for 2018 and 2019 in Germany.

Method 1: ARIMA Forecasting

```
plot(forecast(auto.arima(Germany_monthly), 30),
     main = "ARIMA Forecast: Germany Asylum Seeker Arrivals",
     ylab = "Number of Asylum Seekers",
     xlab = "Year", ylim=c(0, 90000))
```

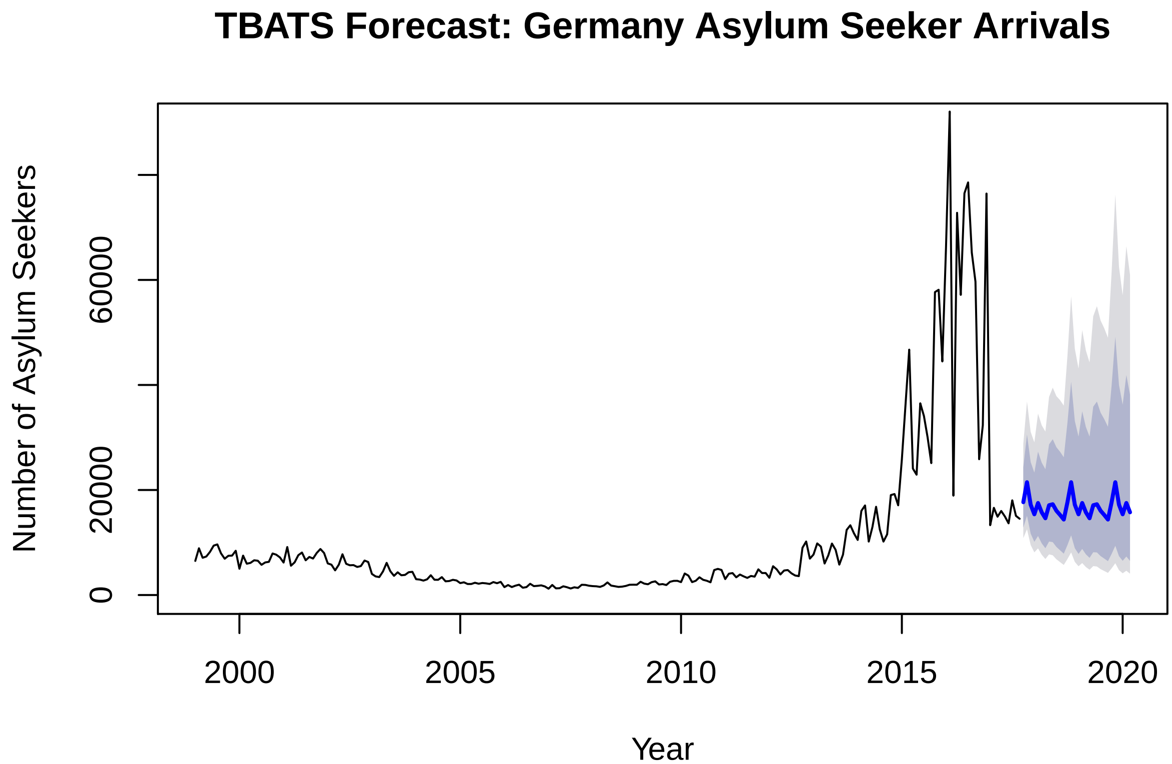



```r
# The wide confidence intervals show the uncertainty in forecasting with the dark grey representing 95 p

# ARIMA forecast values:
forecast(auto.arima(Germany_monthly), 24)
```

```
##          Point Forecast         Lo 80    Hi 80      Lo 95    Hi 95
## Oct 2017      13600.045    4001.42590 23198.66  -1079.776 28279.87
## Nov 2017      12602.114    2690.66757 22513.56  -2556.135 27760.36
## Dec 2017       9783.970    -539.22427 20107.16  -6003.993 25571.93
## Jan 2018      12492.452    1138.14968 23846.75  -4872.455 29857.36
## Feb 2018       4125.247   -8174.02286 16424.52 -14684.863 22935.36
## Mar 2018      22453.966    9277.32413 35630.61   2302.031 42605.90
## Apr 2018      10594.123   -3405.01062 24593.26 -10815.704 32003.95
## May 2018      14587.126    -188.78676 29363.04  -8010.682 37184.93
## Jun 2018       8506.528   -7007.31837 24020.38 -15219.853 32232.91
## Jul 2018       7885.977   -8332.26208 24104.22 -16917.679 32689.63
## Aug 2018      12268.575   -4624.71179 29161.86 -13567.478 38104.63
## Sep 2018      13974.220   -3568.15671 31516.60 -12854.530 40802.97
## Oct 2018      19766.918    1900.25116 37633.58  -7557.791 47091.63
## Nov 2018      18144.497    -251.95305 36540.95  -9990.446 46279.44
## Dec 2018       8374.367  -10523.66592 27272.40 -20527.680 37276.41
## Jan 2019      22188.573    2857.62113 41519.53  -7375.567 51752.71
## Feb 2019      19746.164      -8.22251 39500.55 -10465.563 49957.89
## Mar 2019      21940.352    1771.41815 42109.29  -8905.370 52786.07
## Apr 2019      21460.855     885.72496 42035.98 -10006.091 52927.80
## May 2019      22219.733    1246.27226 43193.19  -9856.407 54295.87
## Jun 2019      20951.172    -413.19458 42315.54 -11722.807 53625.15
## Jul 2019      20087.011   -1661.23559 41835.26 -13174.062 53348.08
## Aug 2019      21572.478    -552.98949 43697.95 -12265.504 55410.46
## Sep 2019      22135.917    -360.44712 44632.28 -12269.303 56541.14
```

Method 2: TBATS Forecasting

```r
plot(forecast(tbats(Germany_monthly), 30),
     main = "TBATS Forecast: Germany Asylum Seeker Arrivals",
     ylab = "Number of Asylum Seekers",
     xlab = "Year", ylim=c(0, 90000))
```




```r
#TBATS forecast values:
forecast(tbats(Germany_monthly), 24)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Oct 2017       17677.72 12843.324 24331.86 10844.966 28815.39
## Nov 2017       21482.21 15101.356 30559.19 12531.104 36827.17
## Dec 2017       17205.55 11693.651 25315.53  9531.590 31057.88
## Jan 2018       15371.73 10134.801 23314.71  8129.215 29066.77
## Feb 2018       17522.07 11242.393 27309.38  8888.643 34541.02
## Mar 2018       15753.26  9844.956 25207.34  7676.085 32329.65
## Apr 2018       14612.84  8911.857 23960.77  6859.269 31130.86
## May 2018       17094.15 10186.737 28685.32  7745.093 37728.38
## Jun 2018       17283.33 10070.144 29663.29  7565.720 39482.50
## Jul 2018       16052.74  9158.985 28135.25  6805.177 37866.81
## Aug 2018       15243.82  8521.070 27270.52  6262.908 37103.21
## Sep 2018       14373.66  7877.165 26227.99  5729.283 36060.75
## Oct 2018       17677.72  9519.380 32827.97  6859.701 45556.21
## Nov 2018       21482.21 11365.829 40602.86  8114.143 56874.17
## Dec 2018       17205.55  8929.264 33152.89  6309.918 46915.19
## Jan 2019       15371.73  7832.675 30167.21  5481.564 43106.30
## Feb 2019       17522.07  8776.609 34981.94  6086.666 50441.87
## Mar 2019       15753.26  7756.225 31995.61  5330.319 46557.28
## Apr 2019       14612.84  7077.428 30171.27  4821.688 44286.35
```

```
## May 2019       17094.15  8148.019 35862.68  5504.322 53087.34
## Jun 2019       17283.33  8108.219 36840.83  5431.517 54996.34
## Jul 2019       16052.74  7417.964 34738.70  4929.540 52274.73
## Aug 2019       15243.82  6939.160 33487.34  4574.840 50793.91
## Sep 2019       14373.66  6447.303 32044.75  4217.526 48986.59
```