Global Bilateral Migration

Remy LeWinter & Benjamin Romine

Setup

An installation of Neo4j is required to create this report.

Click the link below and follow the instructions to install Neo4j on your machine. https://neo4j.com/docs/operations-manual/current/installation/ You should now be able to run Neo4j from the command line.

Now, download Neo4j Desktop from this address: https://neo4j.com/download/ It will consist of only one file. Execute the file and Neo4j Desktop will open.

Create an account, log in, and create a project. Select the project and click "Add Graph". Name the graph.

Click "Manage" and go to the Settings tab. Scroll to the line of code "dbms.directories.import=import", it should be the first uncommented line. Comment it out by typing a "#" in front of it. Scroll to the line of code "dbms.security.auth_enabled=true", it should be the next uncommented line. Change it to "false" and click "Apply".

Now click "Start" (the play button) on the graph and select "Continue Anyway". Go to the Details tab and write down the HTTP port being used.

In this document, go to the first code chunk in the section labeled "Populating the Database" and ensure that the port number is correct (i.e. the number after "http://localhost:").

Once the database is populated via the code in this document, you can open Neo4j Browser from the Desktop to directly visualise the data model and perform Cypher queries.

Project Motivation

This project is designed as an exercise in using the property graph data model for highly connected data and analysing flow over a network. We use the Neo4j graph database to store and query bilateral migration and population data for over 200 countries.

About the Data

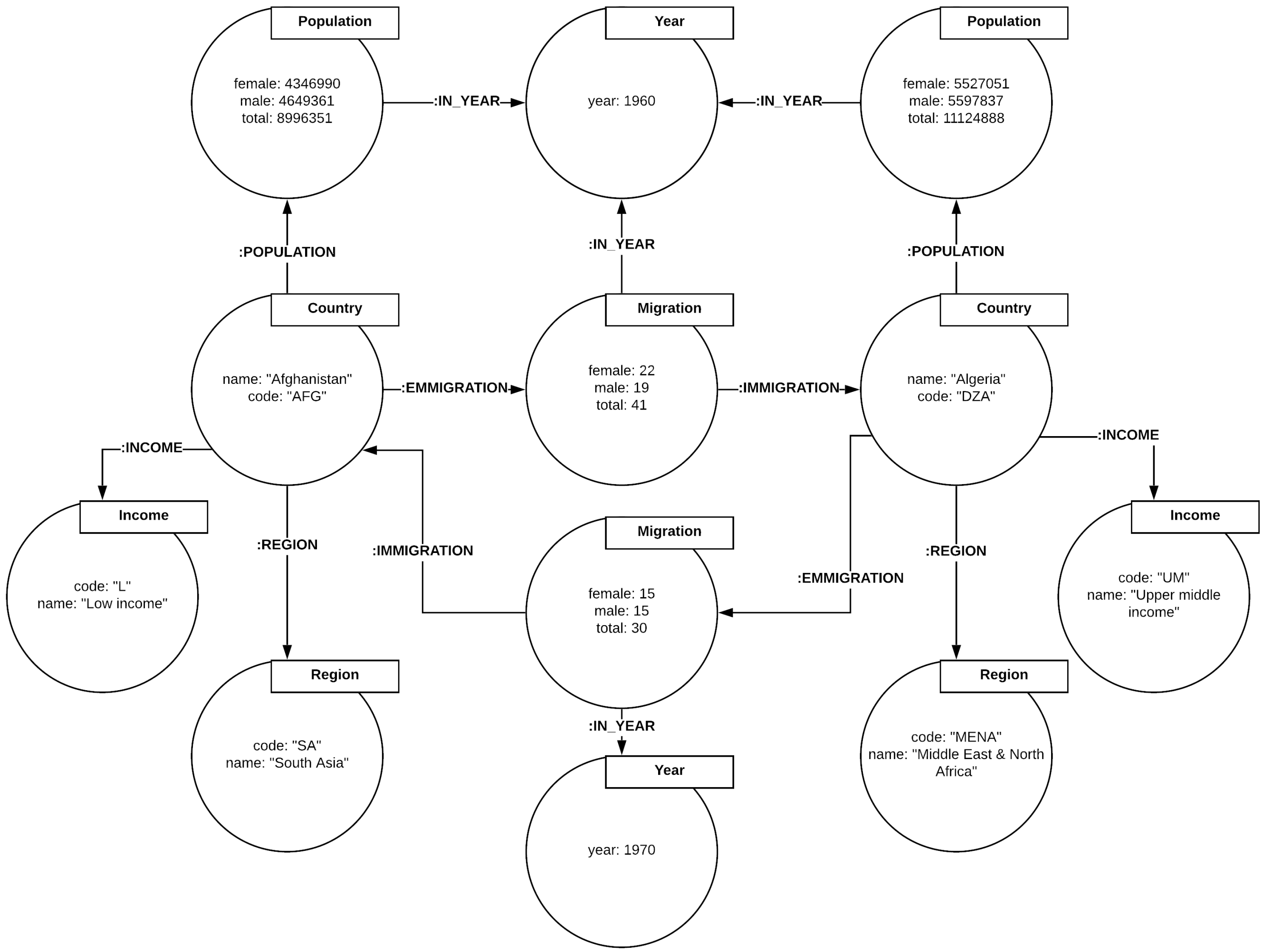
Source:

This project uses all available data from the World Bank Global Bilateral Migration Database, contaning the origin, destination, and gender of migrants for 232 countries in 1960, 1970, 1980, 1990, and 2000 with raw counts of annual migrant flow. This is joined with World Bank's World Development Indicator country-level population data for those years. Each country is also associated with a region and a categorical income level in metadata supplied by World Bank.

Data Model:

The property graph data model requires different thinking than the relational model. In the literal sense, relationships are more central to the graph model than the relational. Neo4j's native graph storage directly encodes relationships between entities, as opposed to the relational model's use of keys for joining tables.

Neo4j's native graph processing uses pointer chasing to match patterns of nodes and edges in relation some set of anchored nodes, which is much faster than searching on data contained in node and edge properties. Accordingly, facts are represented as nodes to optimise query performance. The chart below shows an example of our data model:



Data, metadata, and descriptions from World Bank are found in the ./data folder. Processed data used in the database are in the ./data/final folder.

Processing Data and Populating the Database

After following the setup instructions above, connect to the database:

```
con <- neo4j_api$new(
  url = "http://localhost:7474",
  user = "neo4j",
  password = "neo4j"
)
```

Clear the database if there's anything in it:

```
call_neo4j("MATCH (n) DETACH DELETE n", con)
```

```
## No data returned.
```

```
## list()
## attr(,"class")
## [1] "neo"  "neo"  "list"
```

Pre-Processing

Read the data and rename columns:

```
BilateralMigration <- select(read.csv("./data/Bilateral_Migration.csv", na.strings=c("","NA")),-4)
Population <- read.csv("./data/Population.csv", na.strings=c("","NA"))
IncomeRegion <- select(read.csv("./data/Country_Metadata.csv", na.strings=c("","NA")), 1, 3, 4)

colnames(BilateralMigration) <- c("Origin_Name","Origin_Code","Migration_by_Gender","Dest_Name","Dest_C
colnames(Population) <- c("Series_Name","Series_Code","Country_Name","Country_Code","y1960","y1970","y1
colnames(IncomeRegion) <- c("Country_Code","Income_Name","Region_Name")
```

Processing the Migration Data

If you run the commented line of code for each of the years, you will see that rows with missing values have no data. Therefore we select only rows with complete cases:

```
#BilateralMigration[!complete.cases(BilateralMigration),] %>% summarise(non_na = sum(!is.na(y2000)))

# select complete cases
BilateralMigration <- BilateralMigration[complete.cases(BilateralMigration),]

# gather the years and spread the genders
BilateralMigration <- gather(BilateralMigration, key="Year", value="Mig", y1960, y1970, y1980, y1990, y
BilateralMigration$Year <- BilateralMigration$Year %>% str_sub(2, 5)
BilateralMigration <- spread(BilateralMigration, Migration_by_Gender, Mig)
```

Processing the Population Data

Of the rows with missing values in Population, 6 have data for year 1990, and 7 have data for year 2000. Therefore we will not filter for only complete cases. Run the commented line below for each year to verify:

```
#Population[!complete.cases(Population),] %>% summarise(non_na = sum(!is.na(y2000)))
```

Replace “Series_Name” and “Series_Code” in Population with a single field:

```
Population <- Population %>%
  mutate(Population_by_Gender = Series_Code %>%
           str_replace("SP\\.POP\\.TOTL\\.MA\\.IN", "Male") %>%
           str_replace("SP\\.POP\\.TOTL\\.FE\\.IN", "Female") %>%
           str_replace("SP\\.POP\\.TOTL", "Total")) %>%
  select(Country_Name, Country_Code, Population_by_Gender, y1960, y1970, y1980, y1990, y2000)
```

Gather the years and spread the genders:

```r
# gather
Population <- gather(Population, key="Year", value="Pop", y1960, y1970, y1980, y1990, y2000)

# remove prefixes
Population$Year <- Population$Year %>% str_sub(2, 5)

# we can take complete cases now that years are gathered
Population <- Population[complete.cases(Population),]

# spread
Population <- spread(Population,Population_by_Gender,Pop)
```

Filter Population to only contain data on countries found in BilateralMigration. It is ok if there is migration but not population data for some countries. Since all countries in BilateralMigration are both origins and destinations (run commented line to verify), we only need use one:

```r
#setdiff(BilateralMigration$Dest_Code, BilateralMigration$Origin_Code)

Population <- Population %>% filter(Country_Code %in% BilateralMigration$Origin_Code)
```

Processing Income and Region Data

IncomeRegion contains rows with missing data. All of these are for aggregates and demographic groups, so we'll take only complete cases:

```r
IncomeRegion <- IncomeRegion[complete.cases(IncomeRegion),]
```

Create codes for income and region data:

```r
IncomeRegion$Income_Code <- IncomeRegion$Income_Name %>%
  str_replace("High income", "H") %>%
  str_replace("Low income", "L") %>%
  str_replace("Lower middle income", "LM") %>%
  str_replace("Upper middle income", "UM")
IncomeRegion$Region_Code <- IncomeRegion$Region_Name %>%
  str_replace("East Asia & Pacific", "EAP") %>%
  str_replace("Europe & Central Asia", "ECA") %>%
  str_replace("Latin America & Caribbean", "LAC") %>%
  str_replace("Middle East & North Africa", "MENA") %>%
  str_replace("North America", "NA") %>%
  str_replace("South Asia", "SA") %>%
  str_replace("Sub-Saharan Africa", "SAA")
```

If you run the two commented lines below, you'll see that the same 8 rows in IncomeRegion do not appear in Population or BilateralMigration. Remove those rows:

```r
#setdiff(IncomeRegion$Country_Code, BilateralMigration$Origin_Code)
#setdiff(IncomeRegion$Country_Code, Population$Country_Code)

IncomeRegion <- IncomeRegion %>% filter(Country_Code %in% Population$Country_Code)
```

Save Modified Data to CSV

```
write.csv(BilateralMigration, file="./data/final/Bilateral_Migration_Final.csv", row.names=F)
write.csv(BilateralMigration %>% distinct(Origin_Name, Origin_Code), file="./data/final/Country_Final.cs
write.csv(Population, file="./data/final/Population_Final.csv", row.names=F)
write.csv(IncomeRegion, file="./data/final/Income_Region_Final.csv", row.names=F)
```

Populating Neo4j

Constraints:

```
# Some constraints are restricted to the Enterprise version and may not always work
"CREATE CONSTRAINT ON (c:Country) ASSERT exists(c.name)
CREATE CONSTRAINT ON (c:Country) ASSERT exists(c.code)
CREATE CONSTRAINT ON (c:Country) ASSERT c.name IS UNIQUE
CREATE CONSTRAINT ON (c:Country) ASSERT c.code IS UNIQUE

CREATE CONSTRAINT ON (m:Migration) ASSERT exists(m.total)
CREATE CONSTRAINT ON (m:Migration) ASSERT exists(m.male)
CREATE CONSTRAINT ON (m:Migration) ASSERT exists(m.female)

CREATE CONSTRAINT ON (y:Year) ASSERT exists(y.year)
CREATE CONSTRAINT ON (y:Year) ASSERT y.year IS UNIQUE

CREATE CONSTRAINT ON (i:Income) ASSERT exists(i.code)
CREATE CONSTRAINT ON (i:Income) ASSERT exists(i.name)
CREATE CONSTRAINT ON (i:Income) ASSERT i.code IS UNIQUE
CREATE CONSTRAINT ON (i:Income) ASSERT i.name IS UNIQUE

CREATE CONSTRAINT ON (r:Region) ASSERT exists(r.code)
CREATE CONSTRAINT ON (r:Region) ASSERT exists(r.name)
CREATE CONSTRAINT ON (r:Region) ASSERT r.code IS UNIQUE
CREATE CONSTRAINT ON (r:Region) ASSERT r.name IS UNIQUE;" %>%
  call_neo4j(con)
```

Years:

```
"CREATE (:Year {year: 1960}), (:Year {year: 1970}), (:Year {year: 1980}), (:Year {year: 1990}), (:Year 
  call_neo4j(con)
```

```
## No data returned.
```

```
## list()
## attr(,"class")
## [1] "neo"  "neo"  "list"
```

Countries:

```
on_load_country <- 'CREATE (:Country {code: csvLine.Origin_Code, name: csvLine.Origin_Name})'

country_path <- str_c("file://", getwd(), "/data/final/Country_Final.csv")

load_csv(url=country_path, con=con, on_load=on_load_country, as="csvLine", periodic_commit=500)
```

```
## No data returned.
```

```
## # A tibble: 12 x 2
##    type                 value
##    <chr>                <dbl>
##  1 contains_updates         1
##  2 nodes_created          226
##  3 nodes_deleted            0
##  4 properties_set         452
##  5 relationships_created    0
##  6 relationship_deleted     0
##  7 labels_added           226
##  8 labels_removed           0
##  9 indexes_added            0
## 10 indexes_removed          0
## 11 constraints_added        0
## 12 constraints_removed      0
```

Migrations:

```
on_load_mig <-
'MATCH (o:Country {code: csvLine.Origin_Code})
MATCH (d:Country {code: csvLine.Dest_Code})
MATCH (y:Year {year: toInteger(csvLine.Year)})
CREATE (m:Migration {female: toInteger(csvLine.Female), male: toInteger(csvLine.Male), total: toInteger
MERGE (o)-[:EMMIGRATION]->(m)-[:IMMIGRATION]->(d)
MERGE (m)-[:IN_YEAR]->(y);'

mig_path <- str_c("file://", getwd(), "/data/final/Bilateral_Migration_Final.csv")

load_csv(url=mig_path, con=con, on_load=on_load_mig, as="csvLine", periodic_commit=500)
```

```
## No data returned.
```

```
## # A tibble: 12 x 2
##    type                  value
##    <chr>                 <dbl>
##  1 contains_updates          1
##  2 nodes_created        255380
##  3 nodes_deleted             0
##  4 properties_set       766140
##  5 relationships_created 766140
##  6 relationship_deleted      0
##  7 labels_added         255380
##  8 labels_removed            0
##  9 indexes_added             0
## 10 indexes_removed           0
## 11 constraints_added         0
## 12 constraints_removed       0
```

Populations:

```
on_load_pop <-
'MATCH (c:Country {code: csvLine.Country_Code})
MATCH (y:Year {year: toInteger(csvLine.Year)})
MERGE (p:Population {total: toInteger(csvLine.Total)})
FOREACH(ignoreMe IN CASE WHEN trim(csvLine.Female) <> "" THEN [1] ELSE [] END | SET p.female = toIntege
FOREACH(ignoreMe IN CASE WHEN trim(csvLine.Male) <> "" THEN [1] ELSE [] END | SET p.male = toInteger(cs
MERGE (c)-[:POPULATION]->(p)-[:IN_YEAR]->(y);'
```

```r
pop_path <- str_c("file://", getwd(),"/data/final/Population_Final.csv")

load_csv(url=pop_path, con=con, on_load=on_load_pop, as="csvLine", periodic_commit=500)
```

```
## No data returned.
```

```
## # A tibble: 12 x 2
##    type                  value
##    <chr>                 <dbl>
##  1 contains_updates          1
##  2 nodes_created          1036
##  3 nodes_deleted             0
##  4 properties_set         2920
##  5 relationships_created  2074
##  6 relationship_deleted      0
##  7 labels_added           1036
##  8 labels_removed            0
##  9 indexes_added             0
## 10 indexes_removed           0
## 11 constraints_added         0
## 12 constraints_removed       0
```

Incomes and Regions:

```r
on_load_ir <-
'MATCH (c:Country {code: csvLine.Country_Code})
MERGE (i:Income {code: csvLine.Income_Code, name: csvLine.Income_Name})
MERGE (r:Region {code: csvLine.Region_Code, name: csvLine.Region_Name})
MERGE (c)-[:INCOME]->(i)
MERGE (c)-[:REGION]->(r);'

ir_path <- str_c("file://", getwd(), "/data/final/Income_Region_Final.csv")

load_csv(url=ir_path, con=con, on_load=on_load_ir, as="csvLine", periodic_commit=500)
```

```
## No data returned.
```

```
## # A tibble: 12 x 2
##    type                  value
##    <chr>                 <dbl>
##  1 contains_updates          1
##  2 nodes_created            11
##  3 nodes_deleted             0
##  4 properties_set           22
##  5 relationships_created   416
##  6 relationship_deleted      0
##  7 labels_added             11
##  8 labels_removed            0
##  9 indexes_added             0
## 10 indexes_removed           0
## 11 constraints_added         0
## 12 constraints_removed       0
```

Now that the database is populated, let's free up some RAM:

```
rm(BilateralMigration, IncomeRegion, Population)
```

Summary Statistics

Number of Countries in each Income Level

```
countries_per_income_level <- data.frame(call_neo4j("match (i:Income)<-[ic:INCOME]-(:Country) return i.n
colnames(countries_per_income_level) <- c("Income Level", "Number of Countries")
print(countries_per_income_level)
```

```
##          Income Level Number of Countries
## 1          Low income                  33
## 2 Upper middle income                  54
## 3         High income                  75
## 4 Lower middle income                  46
```

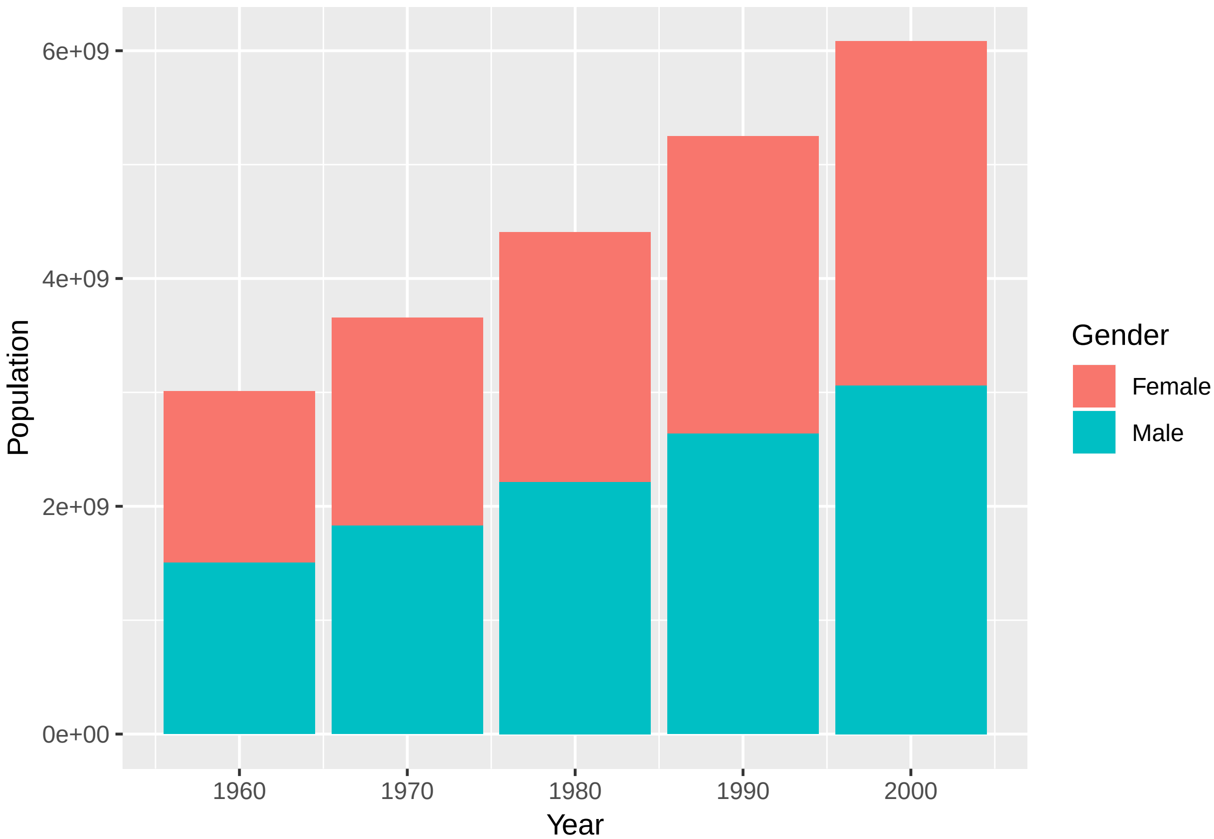
Number of Countries in each Region

```
countries_per_region <- data.frame(call_neo4j("match (r:Region)<-[rc:REGION]-(:Country) return r.name, c
colnames(countries_per_region) <- c("Region", "Number of Countries")
print(countries_per_region)
```

```
##                        Region Number of Countries
## 1                  South Asia                   8
## 2       Europe & Central Asia                  53
## 3 Middle East & North Africa                   21
## 4         East Asia & Pacific                  37
## 5          Sub-Saharan Africa                  47
## 6  Latin America & Caribbean                   39
## 7               North America                   3
```

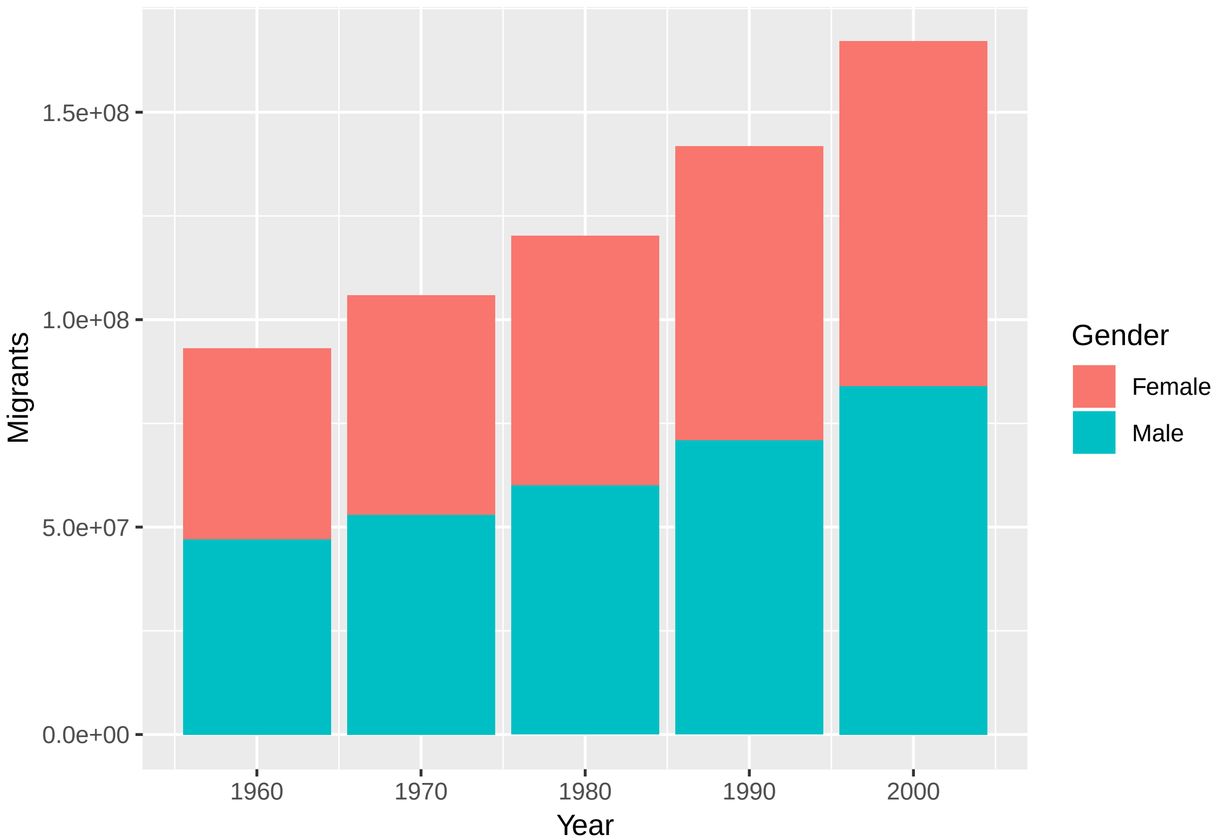
Yearly Global Population

```
global_pop <- call_neo4j("match (p:Population)-->(y:Year) return y.year as Year, sum(p.female) as Femal
colnames(global_pop) <- c("Year", "Female", "Male")
global_pop <- gather(global_pop, `Female`, `Male`, key="Gender", value="Population")

ggplot(global_pop, aes(x=`Year`, y=`Population`, fill=`Gender`)) +
  geom_bar(position="stack", stat="identity")
```



Yearly Global Migration

```
global_mig <- call_neo4j("match (m:Migration)-->(y:Year) return y.year as Year, sum(m.female) as Female
colnames(global_mig) <- c("Year", "Female", "Male")
global_mig <- gather(global_mig, `Female`, `Male`, key="Gender", value="Migrants")

ggplot(global_mig, aes(x=`Year`, y=`Migrants`, fill=`Gender`)) +
  geom_bar(position="stack", stat="identity")
```

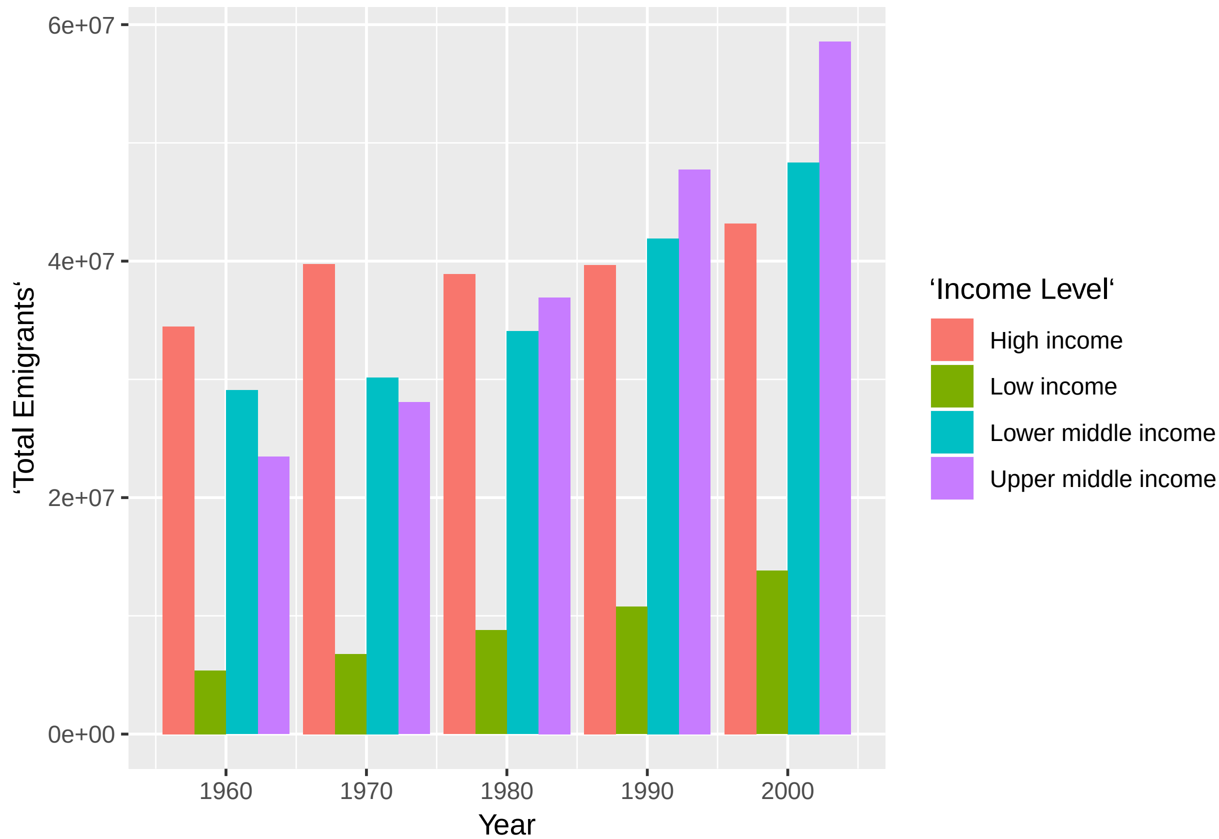


Total Migration by Income Level and Gender

```
emigrants_income <- call_neo4j('MATCH (i:Income)<-[:INCOME]-(c:Country)-[:EMMIGRATION]->(m:Migration)-[
  data.frame()
colnames(emigrants_income) <- c("Income Level", "Year", "Total Emigrants", "Total Male Emigrants", "Tota
immigrants_income <- call_neo4j('MATCH (i:Income)<-[:INCOME]-(c:Country)<-[:IMMIGRATION]-(m:Migration)-
  data.frame()
colnames(immigrants_income) <- c("Income Level", "Year", "Total Immigrants", "Total Male Immigrants", "
```

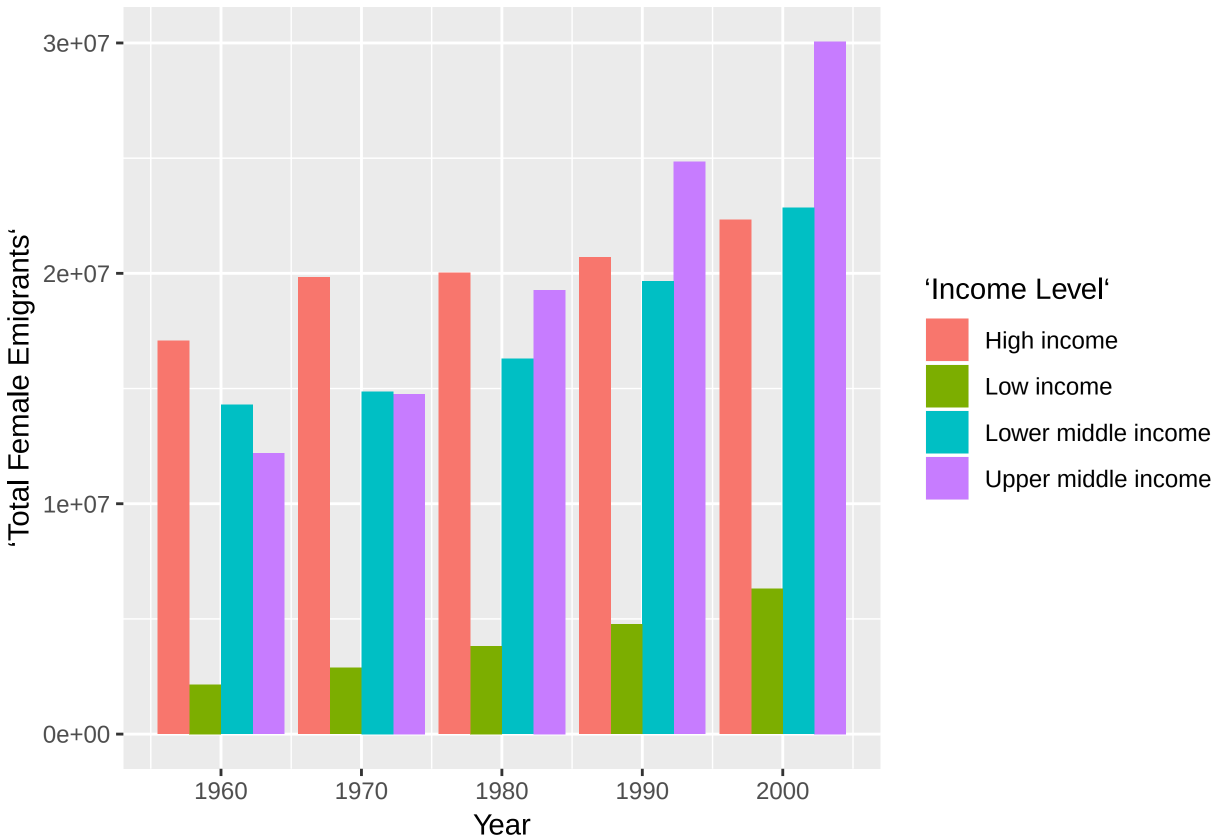
Over time, emigration has increased for all income levels, but the increase is most drastic for middle income countries:

```
ggplot(emigrants_income, aes(x = `Year`, y = `Total Emigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```

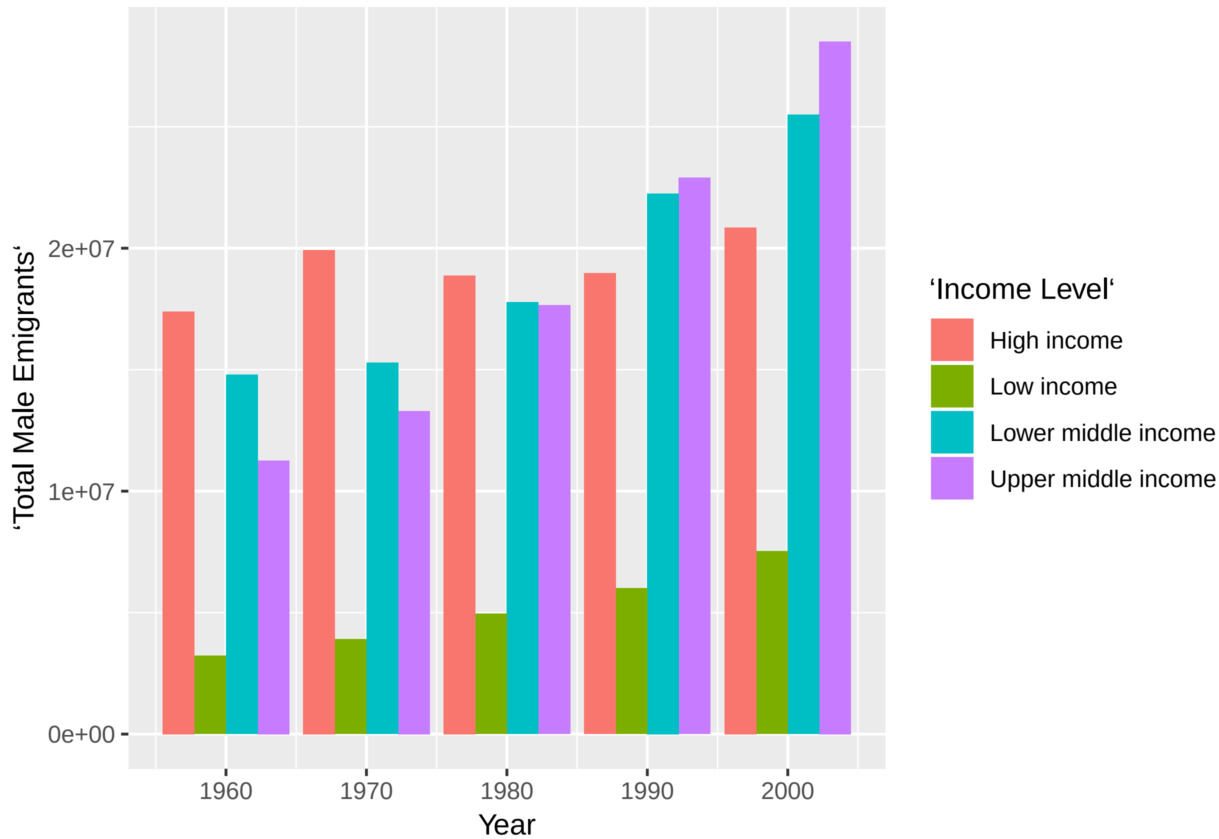


Emigration was roughly proportional between women and men, though the proportion of men emigrating from upper-middle income countries took longer to surpass emigration from lower-middle income countries than it did for women, for whom the difference is much greater:

```
ggplot(emigrants_income, aes(x = `Year`, y = `Total Female Emigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```

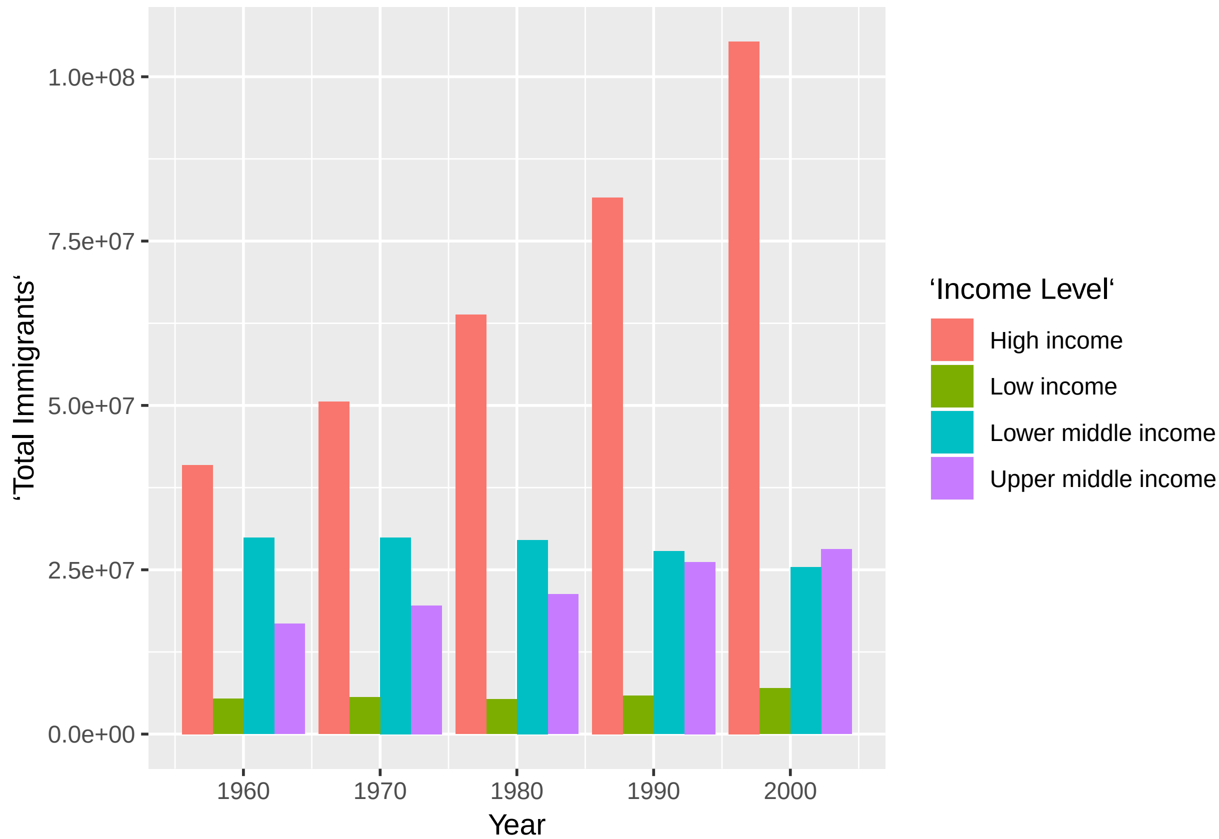



```r
ggplot(emigrants_income, aes(x = `Year`, y = `Total Male Emigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```



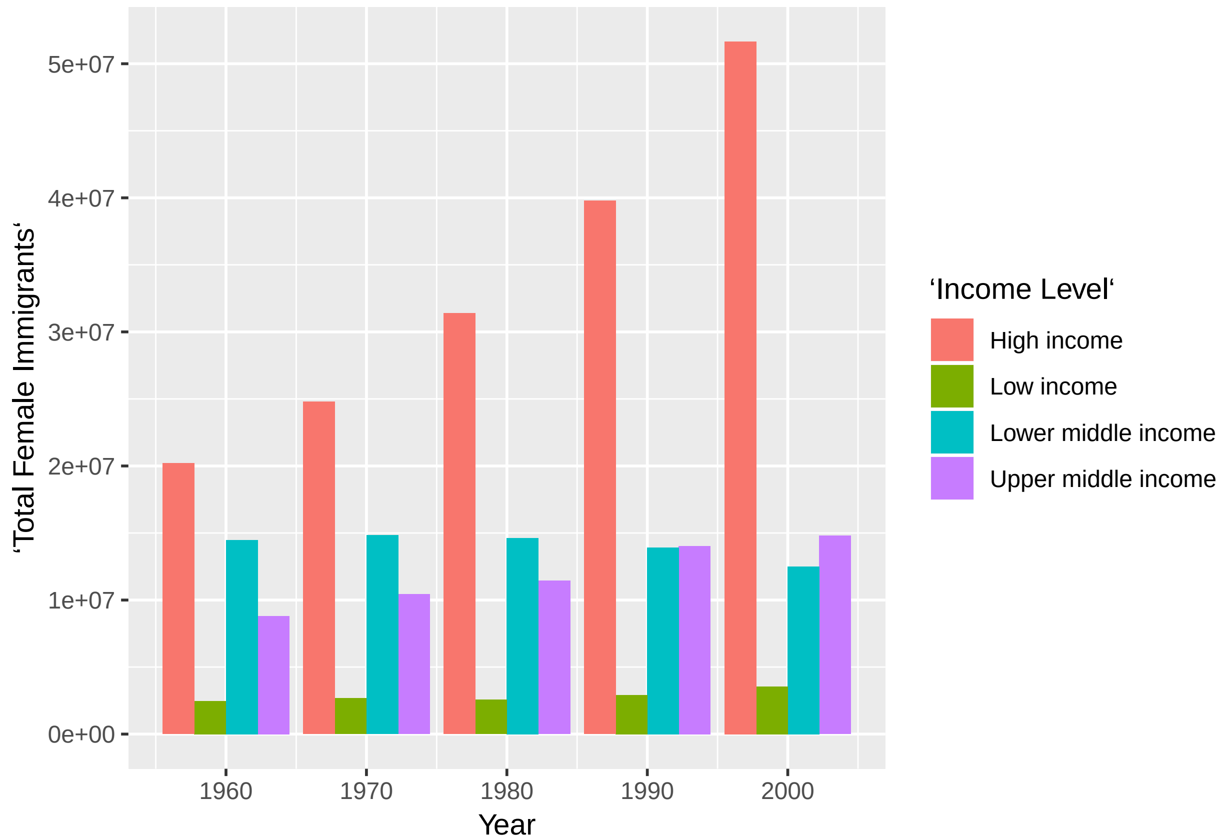
High income countries always received the lion's share of immigrants, and they seem to be the destination for most of the increase in migrants shown in the previous charts, such that their share grew larger every decade. Immigration to upper-middle income countries experienced a much less dramatic rise each decade, while immigration to lower-middle income countries actually decreased between 1960 and 2000:

```
ggplot(immigrants_income, aes(x = `Year`, y = `Total Immigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```

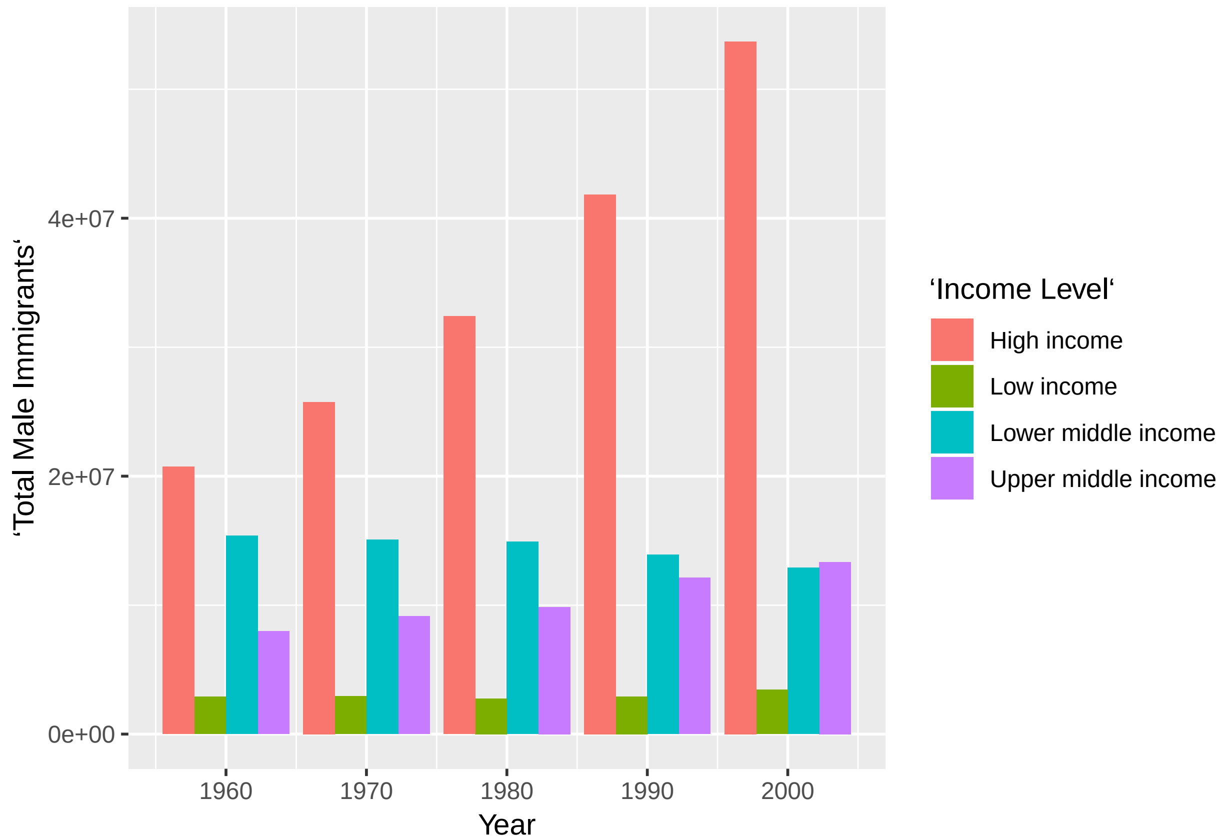


Immigration to different income-level countries followed roughly the same trends for men and women:

```
ggplot(immigrants_income, aes(x = `Year`, y = `Total Female Immigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```




```r
ggplot(immigrants_income, aes(x = `Year`, y = `Total Male Immigrants`, fill = `Income Level`)) +
  geom_bar(position = "dodge", stat = "identity")
```



Now, let's try using this data to predict the total number of immigrants to all countries with different income levels in 2010. First, we create linear regression models:

```
immigrant_income_linreg <- lm(`Total Immigrants` ~ `Income Level` + Year, immigrants_income)
emigrant_income_linreg <- lm(`Total Emigrants` ~ `Income Level` + Year, emigrants_income)
income_levels <- c("Low income", "Lower middle income", "Upper middle income", "High income")
new_data_income <- data.frame("Income Level" = income_levels, "Year" = rep(2010, 4))
colnames(new_data_income) <- c("Income Level", "Year")
summary(immigrant_income_linreg)
```

```
##
## Call:
## lm(formula = `Total Immigrants` ~ `Income Level` + Year, data = immigrants_income)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -18433135  -4810892   -632322   4739676  27835865
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -830900433  353854144  -2.348   0.0330
## `Income Level`Low income            -62628413    7147839  -8.762 2.76e-07
## `Income Level`Lower middle income   -39954819    7147839  -5.590 5.16e-05
## `Income Level`Upper middle income   -46070873    7147839  -6.445 1.10e-05
## Year                                   454220     178696   2.542   0.0226
##
## (Intercept)                        *
## `Income Level`Low income           ***
```

```
## `Income Level`Lower middle income ***
## `Income Level`Upper middle income ***
## Year                              *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11300000 on 15 degrees of freedom
## Multiple R-squared:  0.8562, Adjusted R-squared:  0.8179
## F-statistic: 22.33 on 4 and 15 DF,  p-value: 3.576e-06
```

```
summary(emigrant_income_linreg)
```

```
##
## Call:
## lm(formula = `Total Emigrants` ~ `Income Level` + Year, data = emigrants_income)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6579114 -3094973  -313376  3081755 10702913
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -843952398  158259952  -5.333 8.37e-05
## `Income Level`Low income          -30087071    3196845  -9.411 1.10e-07
## `Income Level`Lower middle income  -2478609    3196845  -0.775    0.450
## `Income Level`Upper middle income   -235356    3196845  -0.074    0.942
## Year                                 446033      79921   5.581 5.25e-05
##
## (Intercept)                       ***
## `Income Level`Low income          ***
## `Income Level`Lower middle income
## `Income Level`Upper middle income
## Year                              ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5055000 on 15 degrees of freedom
## Multiple R-squared:  0.9127, Adjusted R-squared:  0.8895
## F-statistic: 39.22 on 4 and 15 DF,  p-value: 8.938e-08
```

Notice that the residual standard error values are very high. Using these linear regression models, we will create confidence intervals for the values that we will predict:

```
predictions_2010_income <- data.frame(income_levels, predict(immigrant_income_linreg, new_data_income,
colnames(predictions_2010_income) <-
  c("Income Level", "Expected Immigrants in 2010", "Lower Bound", "Upper Bound", "Expected Emigrants in
predictions_2010_income
```

```
##          Income Level Expected Immigrants in 2010 Lower Bound Upper Bound
## 1          Low income                    19453183     3749038    35157329
## 2 Lower middle income                    42126777    26422632    57830922
## 3 Upper middle income                    36010723    20306578    51714868
## 4         High income                    82081596    66377451    97785742
##   Expected Emigrants in 2010 Lower Bound Upper Bound
## 1                   22486902    15463281    29510522
## 2                   50095363    43071743    57118984
```

```
## 3                        52338616     45314996     59362237
## 4                        52573973     45550352     59597593
```

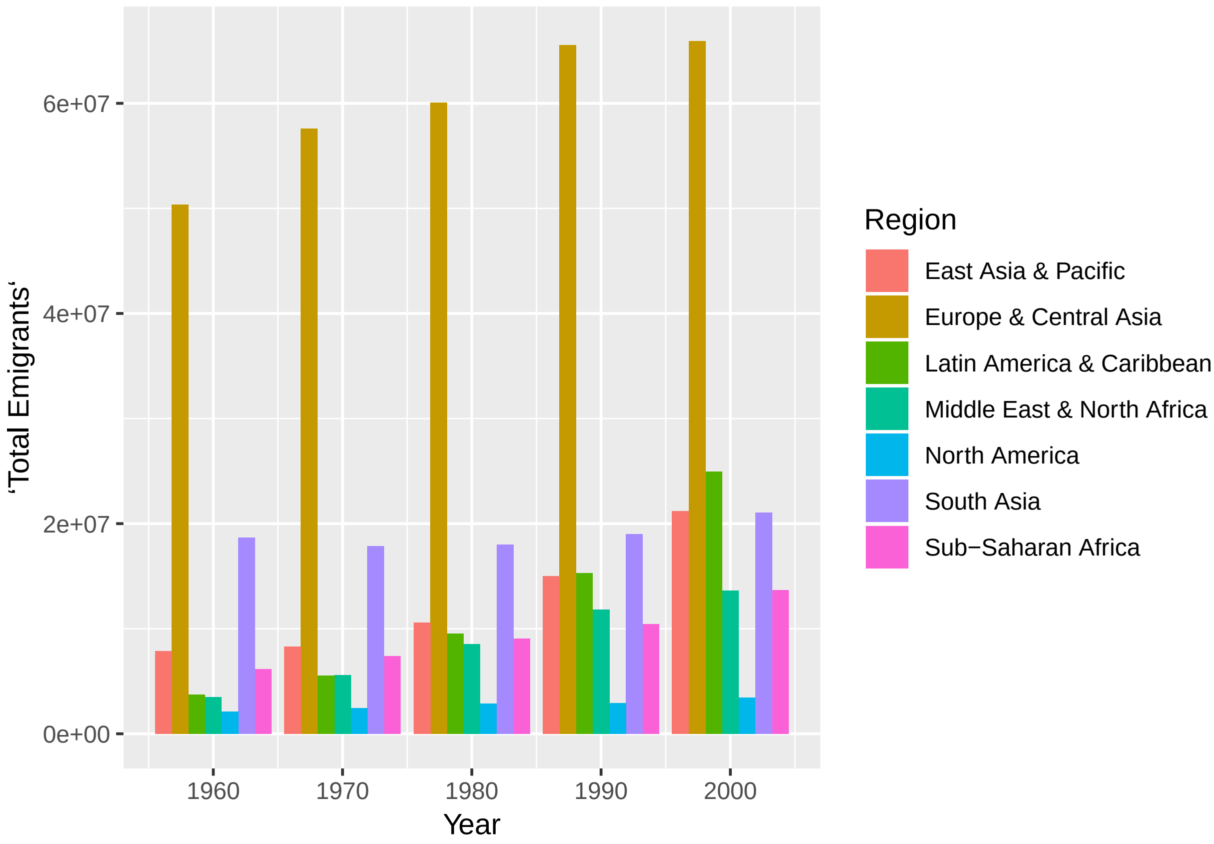
The confidence intervals for immigration are very large, so the data that is presented in these linear regressions is not much better than wild guessing.

Total Migration by Region and Gender

```r
emigrants_region <- call_neo4j('MATCH (r:Region)<-[:REGION]-(c:Country)-[:EMIGRATION]->(m:Migration)-[
  data.frame()
colnames(emigrants_region) <- c("Region", "Year", "Total Emigrants", "Total Male Emigrants", "Total Fema
immigrants_region <- call_neo4j('MATCH (r:Region)<-[:REGION]-(c:Country)<-[:IMMIGRATION]-(m:Migration)-
  data.frame()
colnames(immigrants_region) <- c("Region", "Year", "Total Immigrants", "Total Male Immigrants", "Total I
```

Emigrants leave Europe and Central Asia at a much greater rate than any other region. While emigration from South Asia and North America stayed roughly equal over time, emigration from every other region increased each decade.

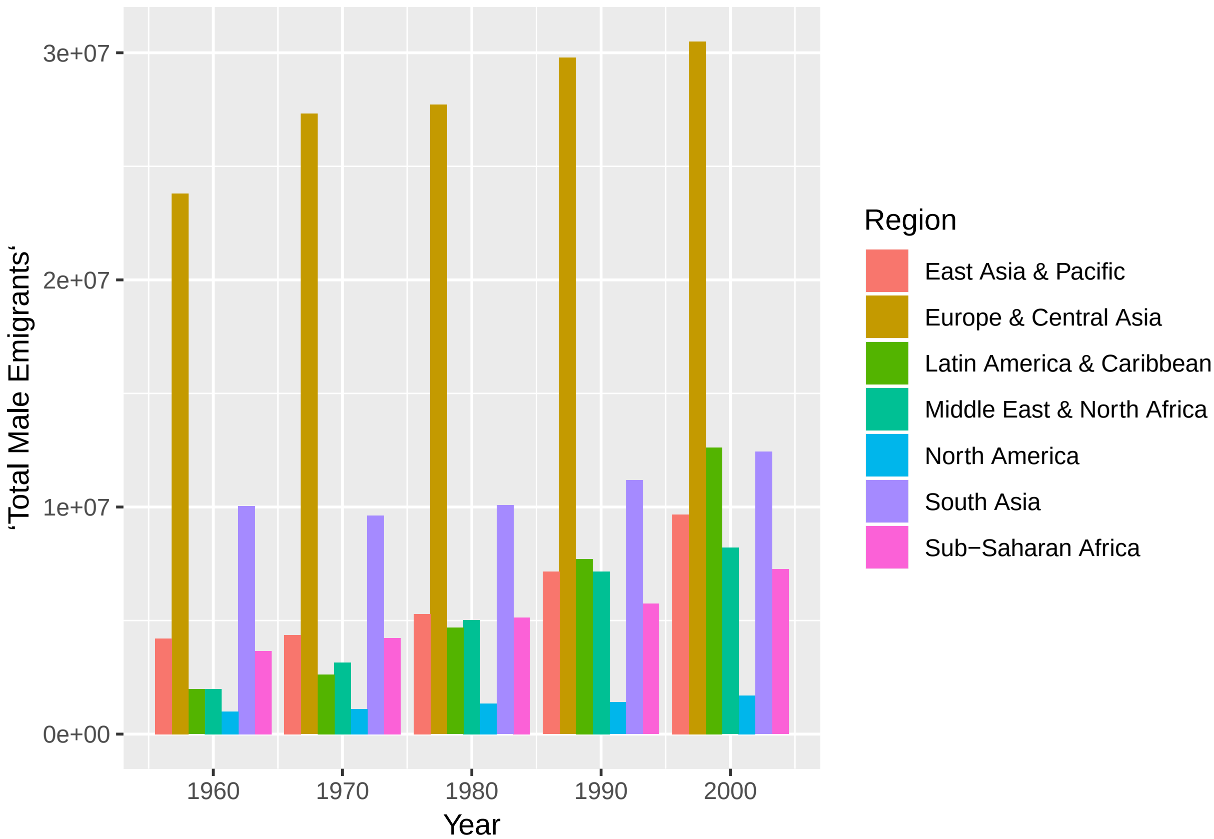
```r
ggplot(emigrants_region, aes(x = `Year`, y = `Total Emigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```



Women and men seem to emigrate from the different regions at proportional rates, with one exception - South Asian emigrants are more likely to be male:

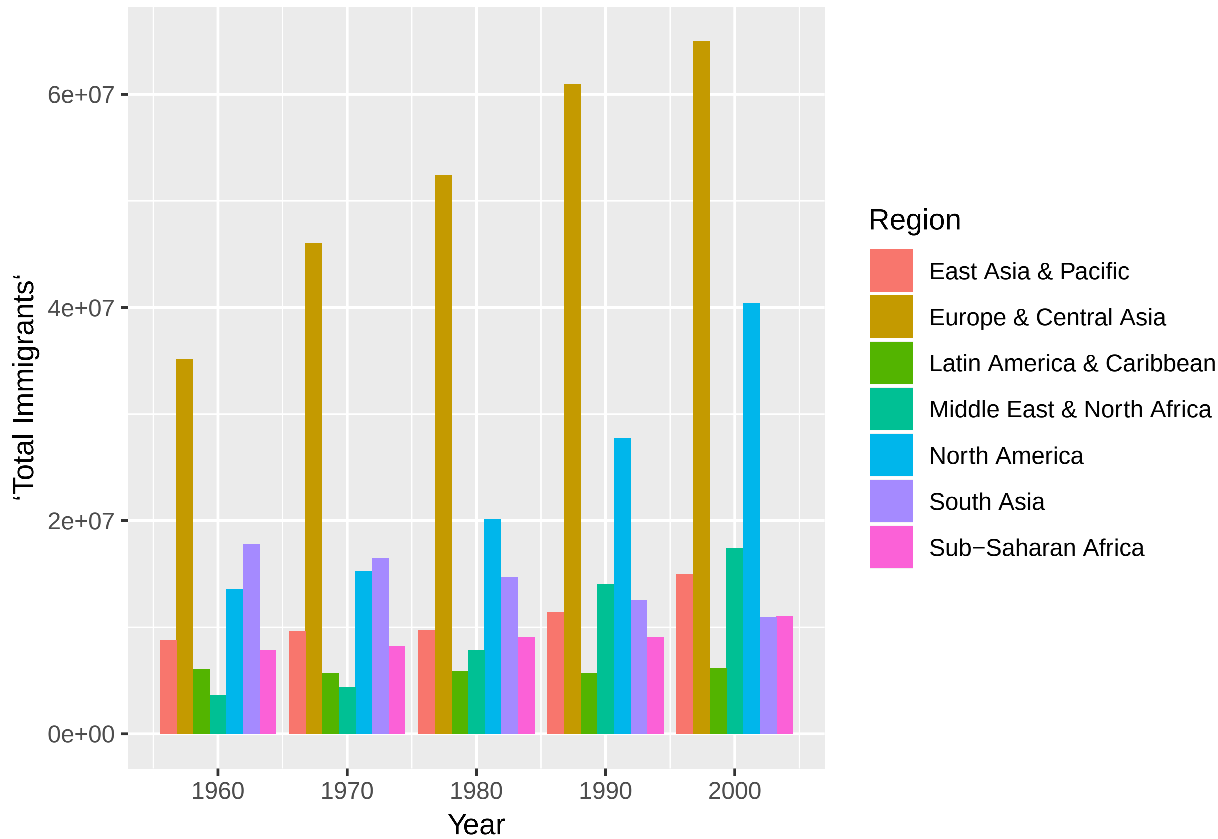
```r
ggplot(emigrants_region, aes(x = `Year`, y = `Total Female Emigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```


```r
ggplot(emigrants_region, aes(x = `Year`, y = `Total Male Emigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```



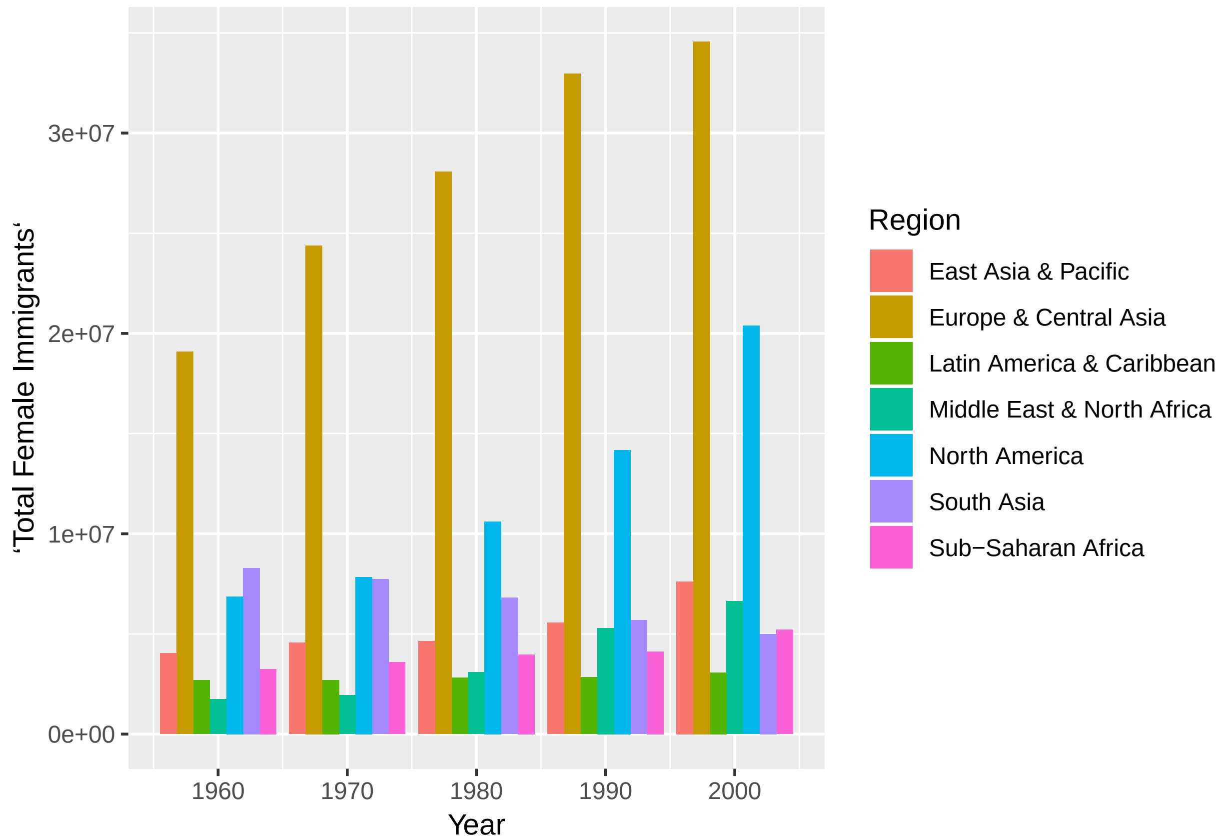
Immigration to Latin America and Caribbean countries and Sub-Saharan Africa stayed roughly equal over time. Immigration to South Asia actually decreased every decade, while immigration to Europe and Central Asia increased monotonically and the proportion immigrating to North America greatly increased from 1980-2000:

```
ggplot(immigrants_region, aes(x = `Year`, y = `Total Immigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```

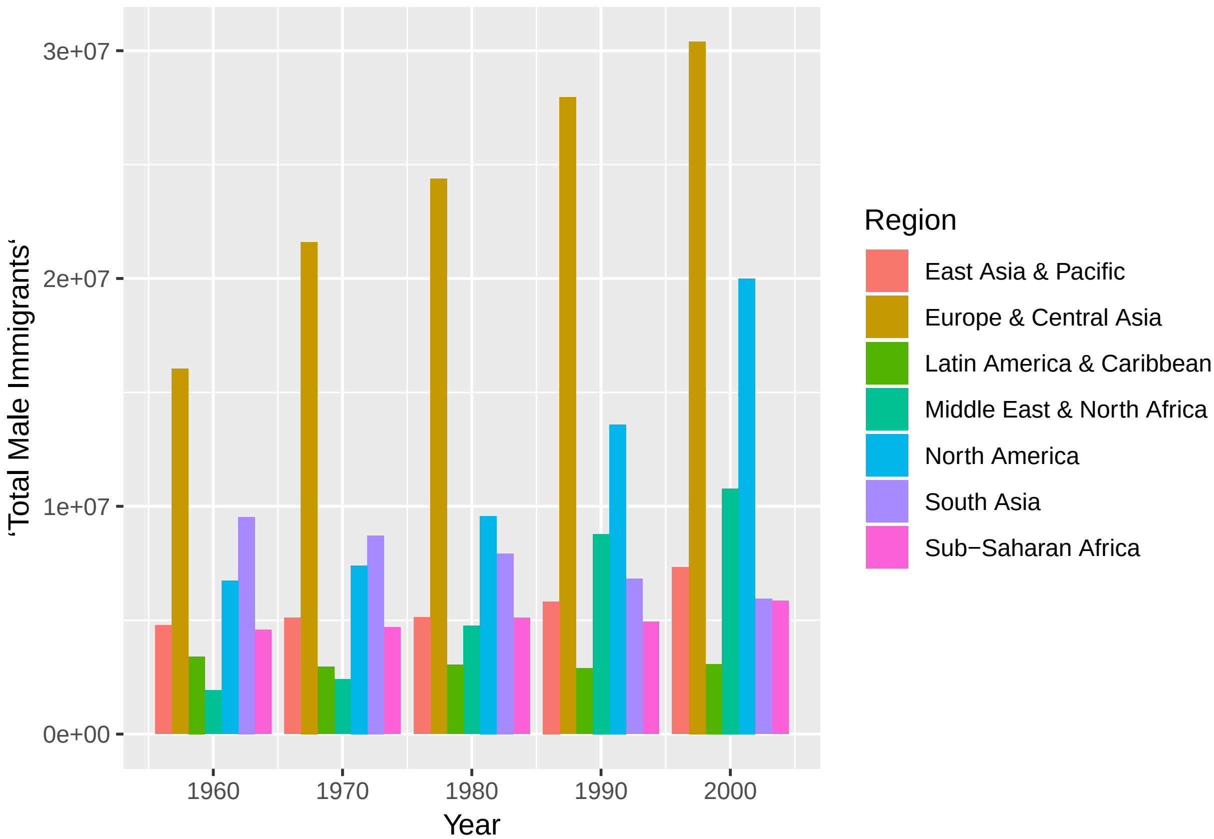


Women and men seem to immigrate to the different regions at proportional rates, with one exception - men are much more likely to immigrate to the Middle East and North Africa:

```
ggplot(immigrants_region, aes(x = `Year`, y = `Total Female Immigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```




```
ggplot(immigrants_region, aes(x = `Year`, y = `Total Male Immigrants`, fill = `Region`)) +
  geom_bar(position = "dodge", stat = "identity")
```



As before, we're going to try to predict the total number of migrants to and from each region of the world for the year 2010. We create linear regressions once more:

```r
immigrant_region_linreg <- lm(`Total Immigrants` ~ `Region` + Year, immigrants_region)
emigrant_region_linreg <- lm(`Total Emigrants` ~ `Region` + Year, emigrants_region)
regions <- unique(emigrants_region$Region)
new_data_region <- data.frame("Region" = regions, "Year" = rep(2010, 7))
colnames(new_data_income) <- c("Region", "Year")
summary(immigrant_region_linreg)
```

```
##
## Call:
## lm(formula = `Total Immigrants` ~ Region + Year, data = immigrants_region)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -11579234  -2977989    -38173   2553975  11759636
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -503003593  127765596  -3.937 0.000523
## RegionEurope & Central Asia      40980575    3413896  12.004 2.46e-12
## RegionLatin America & Caribbean  -5012983    3413896  -1.468 0.153549
## RegionMiddle East & North Africa -1440839    3413896  -0.422 0.676329
## RegionNorth America              12522218    3413896   3.668 0.001058
## RegionSouth Asia                  3576277    3413896   1.048 0.304127
## RegionSub-Saharan Africa         -1855935    3413896  -0.544 0.591149
## Year                               259554      64517   4.023 0.000416
```

```
##
## (Intercept)                    ***
## RegionEurope & Central Asia      ***
## RegionLatin America & Caribbean
## RegionMiddle East & North Africa
## RegionNorth America              **
## RegionSouth Asia
## RegionSub-Saharan Africa
## Year                           ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5398000 on 27 degrees of freedom
## Multiple R-squared:  0.9122, Adjusted R-squared:  0.8894
## F-statistic: 40.06 on 7 and 27 DF,  p-value: 1.207e-12
```

```
summary(emigrant_income_linreg)
```

```
##
## Call:
## lm(formula = `Total Emigrants` ~ `Income Level` + Year, data = emigrants_income)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6579114 -3094973  -313376  3081755 10702913
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -843952398  158259952  -5.333 8.37e-05
## `Income Level`Low income           -30087071    3196845  -9.411 1.10e-07
## `Income Level`Lower middle income   -2478609    3196845  -0.775    0.450
## `Income Level`Upper middle income    -235356    3196845  -0.074    0.942
## Year                                  446033      79921   5.581 5.25e-05
##
## (Intercept)                      ***
## `Income Level`Low income         ***
## `Income Level`Lower middle income
## `Income Level`Upper middle income
## Year                             ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5055000 on 15 degrees of freedom
## Multiple R-squared:  0.9127, Adjusted R-squared:  0.8895
## F-statistic: 39.22 on 4 and 15 DF,  p-value: 8.938e-08
```

Similarly, the residual standard errors are very large. However, we will still attempt to predict the numbers of migrants for the year 2010:

```
predictions_2010_region <- data.frame(regions, predict(immigrant_region_linreg, new_data_region, interva
colnames(predictions_2010_region) <-
  c("Region", "Expected Immigrants in 2010", "Lower Bound", "Upper Bound", "Expected Emigrants in 2010"
predictions_2010_region
```

```
##                        Region Expected Immigrants in 2010 Lower Bound
## 1                  South Asia                    22276701    15928121
```

```
## 2      Europe & Central Asia                   59680999    53332418
## 3 Middle East & North Africa                   17259585    10911005
## 4        East Asia & Pacific                   18700424    12351844
## 5         Sub-Saharan Africa                   16844489    10495908
## 6  Latin America & Caribbean                   13687441     7338861
## 7              North America                   31222642    24874062
##   Upper Bound Expected Emigrants in 2010 Lower Bound Upper Bound
## 1    28625281                   26575053    23057184    30092922
## 2    66029579                   67548824    64030955    71066693
## 3    23608165                   16273435    12755566    19791304
## 4    25049004                   20241628    16723758    23759497
## 5    23193069                   16988058    13470189    20505927
## 6    20036021                   19458509    15940640    22976378
## 7    37571222                   10409348     6891478    13927217
```

Once again, it is clear that the confidence intervals for these predicted values are very large, and the linear regression models that we created are not very accurate.

Limitations & Challenges

Below are listed some of the limitations that we discovered when it came to performing analysis on this data:

- The data only goes up to the year 2000, and does not account for significant historical events in recent history that have had a deal of impact on migration patterns (9/11, travel bans, international politics).
- The data is in 10-year increments, and therefore year-to-year prediction was not possible.
- When attempting to create linear regressions for the data, there were very large residual standard error values, which resulted in inaccurate predictions with little confidence.

Future Work

One of the possible ways to utilize this data in the future is to link it to a database of news articles that go into detail about international politics to show the different ways that certain events affected immigration patterns in the future. Our chosen graph model is designed to make best use of the pointer-chasing employed by neo4j's native graph storage and processing. It is worth noting that our strategy of modeling all facts as nodes, i.e. (:Country)-[:MIGRATION]->(:Country) actually becomes a hindrance when using graph algorithms for things like clustering, centrality, link prediction, etc., because they must treat nodes of interest (e.g. countries) as directly adjacent.