Population Projections

Guy J. Abel

Motivations

Nathan Keyfitz said:

If there is one thing that the public thinks demographers do, it is population projections; if there is one thing in which most demographers take no interest whatsoever, it is population projections.

- Population projection is the demographic technique that is most frequently requested by demography's "clients" .
- Population size in a distant future effects the behavior of many different socioeconomic and environment systems.
  - Governments to anticipate demand of services (roads, schools, medical personal, national parks, etc., etc.).
  - Providing a base for other projections essential for social and economic planning such as climate, energy or social security
  - Private business to estimate potential size of future "market".
  - Study the implications of certain demographic parameters (the projection models inputs) over time.
  - Estimate demographic impacts of past disasters or conflicts.
  - Identifying realistic goals and targets for future development.

Population Projections

Preston, Heuveline and Guillot

Calculations which show the future development of a population when certain assumptions are made about the future course of fertility, mortality and migration.

- Purely formal, developing the implication of the assumptions made.
- Can project from present day, or from a past point in time to examine "what if?" type questions.
- Projection method should be internally valid, i.e. obey demographic accounting relations

Requirements

1. Projection period.
   - When should the projection launch.
   - When should the projection period end.
   - Backwards or forwards?
2. Data.
   - Base year population
   - Future assumptions for rates of components of population change (births, deaths, migration).
   - Future assumptions for rates of transitions (e.g. marriage/divorce, enter/leave religion).
3. Projection model.
   - In most cases research question or clients require projection models that provide future population structures.
   - Additional dimensions (e.g. education, religion, marital status, ethnicity, nationality, urban/rural) might be required.
   - Defined projection periods and intervals (annual steps vs five-year steps).

Population Forecasts

Population projection and population forecasts are not the same.

Population projections are calculations which show the future development of a population when certain assumptions are made about the future course of fertility, mortality, and migration. They are in general purely formal calculations, developing the implications of the assumptions that are made. A population forecast is a projection in which the assumptions are considered to yield a realistic picture of the probable future development of a population (UN 1958)

Population Forecasts

- A population forecast is a projection in which the assumptions are considered to yield a realistic picture of the probable future development of a population.
- The underlying cohort component mechanism will be the same, but the future assumptions on the components of population change and/or transition rates are set according to a predictive mechanism.
- Predictive mechanism usually involves some form of statistical forecast.
- A projection is assessed on its ability to match the known demographic process that exist in a population.
- A forecast is assessed on its ability to predicts the future size and composition of a population.

Simple Projection Model

- The simplest population projection model is based on a population growth rate.

$$N(t+1) = N(t) \times (1+r)$$

- Assume we at time t, we know the current population $N(t) = 100$ and a crude annual growth rate $r = 5\% = 0.05$ over the next year.

$$\begin{aligned} N(t+1) &= 100 \times (1+0.05) \\ &= 105 \end{aligned}$$

$$\begin{aligned} N(t+2) &= N(t+1) \times (1+r) \\ &= 105 \times (1+0.05) \\ &= 110.25 \end{aligned}$$

Simple Projection Model

- We can write a function in R to run this projection model for as long as we want

```
> pp_rate <- function(n = NULL, N0 = NULL, rate = NULL){
+   NN <- rep(NA, times = n+1)
+   NN[1] <- N0
+   for(i in 1:n){
+     NN[i+1] <- NN[i] * (1 + rate)
+   }
+   return(NN)
+ }
>
> #testing the projection model
> pp0 <- pp_rate(n = 20, N0 = 100, rate = 0.05)
> pp0
 [1] 100.0000 105.0000 110.2500 115.7625 121.5506 127.6282 134.0096
 [8] 140.7100 147.7455 155.1328 162.8895 171.0339 179.5856 188.5649
[15] 197.9932 207.8928 218.2875 229.2018 240.6619 252.6950 265.3298
>
> pp1 <- pp_rate(n = 20, N0 = 110, rate = 0.03)
> pp1
 [1] 110.0000 113.3000 116.6990 120.2000 123.8060 127.5201 131.3458
 [8] 135.2861 139.3447 143.5251 147.8308 152.2657 156.8337 161.5387
[15] 166.3849 171.3764 176.5177 181.8132 187.2676 192.8857 198.6722
```

Plot Projections

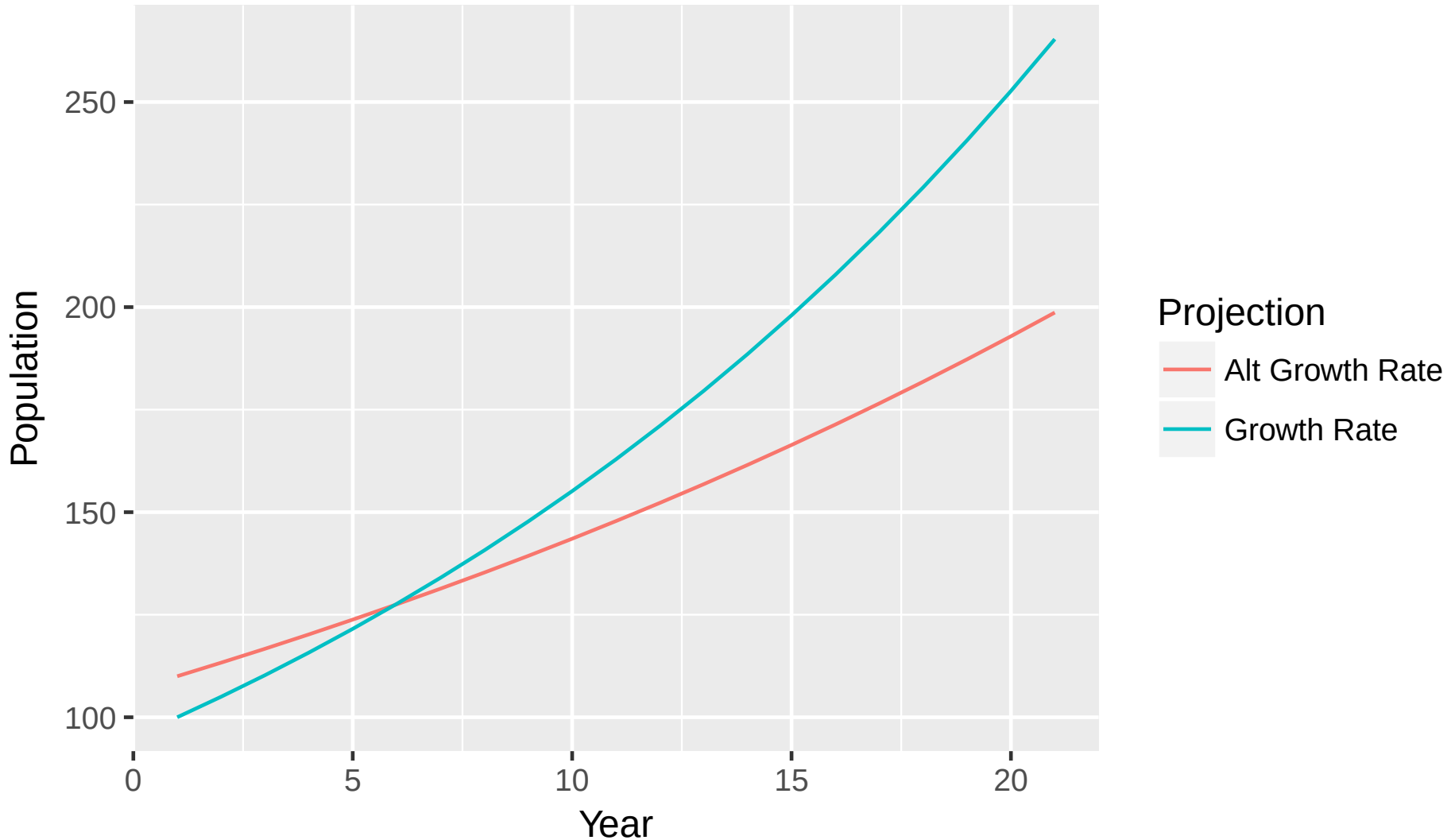
```
> library(tidyverse)
> df0 <- data_frame("Year" = 1:21, `Growth Rate` = pp0, `Alt Growth Rate` = pp1)
> df0
# A tibble: 21 x 3
    Year `Growth Rate` `Alt Growth Rate`
   <int>         <dbl>              <dbl>
 1     1      100.0000           110.0000
 2     2      105.0000           113.3000
 3     3      110.2500           116.6990
 4     4      115.7625           120.2000
 5     5      121.5506           123.8060
 6     6      127.6282           127.5201
 7     7      134.0096           131.3458
 8     8      140.7100           135.2861
 9     9      147.7455           139.3447
10    10      155.1328           143.5251
# ... with 11 more rows
```

Plot Projections

```
> # tidy format for plotting
> df1 <- gather(data = df0, key = Projection, value = Population, -Year)
> df1
# A tibble: 42 x 3
    Year  Projection Population
   <int>       <chr>      <dbl>
 1     1 Growth Rate   100.0000
 2     2 Growth Rate   105.0000
 3     3 Growth Rate   110.2500
 4     4 Growth Rate   115.7625
 5     5 Growth Rate   121.5506
 6     6 Growth Rate   127.6282
 7     7 Growth Rate   134.0096
 8     8 Growth Rate   140.7100
 9     9 Growth Rate   147.7455
10    10 Growth Rate   155.1328
# ... with 32 more rows
```

Plot Projections

```
> ggplot(data = df1, mapping = aes(x = Year, y = Population, colour = Projection))
+   geom_line()
```



Including Demographic Components

- The population growth rate r can be decomposed into crude rates of demographic components; births (b), deaths (d) and net migration (m)

$$N(t+1) = N(t) \times (1+r)$$
$$r = b - d + m$$

- Annual crude rates are based on the number of events divided by the person-years lived during the year.

Exercise 1 (ex51.R)

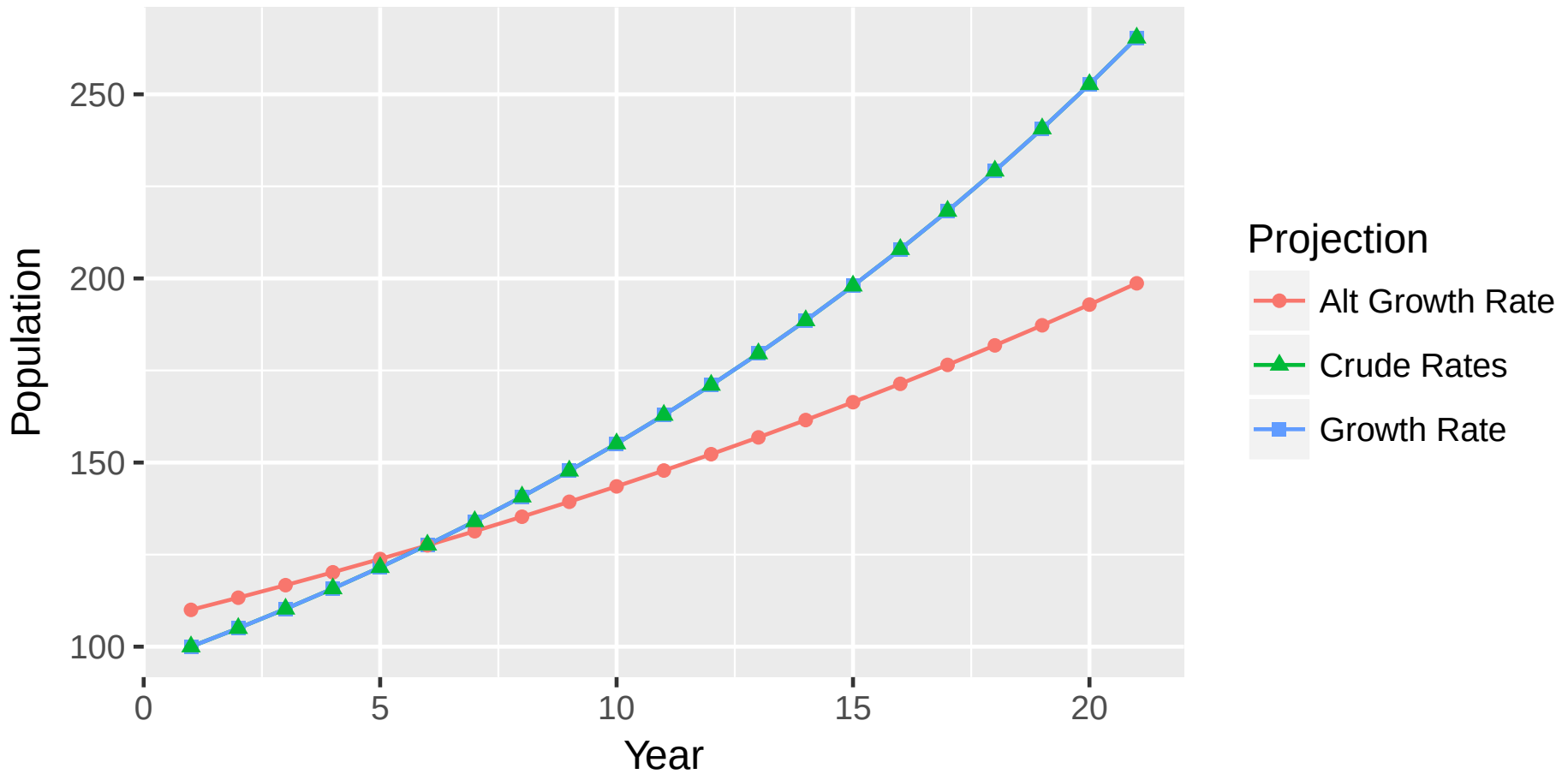
```
# 0.  Clear your workspace and set your working directory to the course folder


# 1. complete the population projection function based on crude demographic rates
pp_demo <- function(n = #####, ##### = NULL, b = NULL, d = #####, ##### = NULL){
  NN <- rep(NA, times = #####+1)
  NN[#####] <- N0
  for(i in 1:n){
   NN[i+1] <- NN[i] * (1 + ##### - ##### + #####)
  }
  return(#####)
}
# 2. Save the projection results when n = 20, N0 = 100, b = 0.04, d = 0.02, m = 0.03
#    in an object called pp2
pp2 <- pp_demo(n = 20, N0 = #####, b = 0.04, d = #####, m = 0.03)
pp2
# 3. Uncomment the code from the slides below
# pp_rate <- function(n = NULL, N0 = NULL, rate = NULL){
#   NN <- rep(NA, times = n+1)
#   NN[1] <- N0
#   for(i in 1:n){
#     NN[i+1] <- NN[i] * (1 + rate)
#   }
#   return(NN)
```

Including Demographic Components

```
> df0 <- mutate(df0, `Crude Rates` = pp2)
> df0
# A tibble: 21 x 4
    Year `Growth Rate` `Alt Growth Rate` `Crude Rates`
   <int>         <dbl>             <dbl>         <dbl>
 1     1      100.0000          110.0000      100.0000
 2     2      105.0000          113.3000      105.0000
 3     3      110.2500          116.6990      110.2500
 4     4      115.7625          120.2000      115.7625
 5     5      121.5506          123.8060      121.5506
 6     6      127.6282          127.5201      127.6282
 7     7      134.0096          131.3458      134.0096
 8     8      140.7100          135.2861      140.7100
 9     9      147.7455          139.3447      147.7455
10    10      155.1328          143.5251      155.1328
# ... with 11 more rows
> df1 <- gather(data = df0, key = Projection, value = Population, -Year)
```

Including Demographic Components

```
> ggplot(data = df1, mapping = aes(x = Year, y = Population,
+                                  colour = Projection, shape = Projection)) +
+   geom_line() +
+   geom_point()
```



Net Migration Counts

- Net migration rates are a bad idea:
  - The true denominator should be the population at risk, i.e. the global population.
  - Easier to set assumptions on net migration counts.
- Net migration can be treated as a count (M) rather than a rate in a projection model:.

$$N(t+1) = N(t) \times (1 + b - d) + M$$

- In this model, migration events occur at the end of period, rather than evenly throughout the year. A more realistic approach is to average arrivals through out the period:

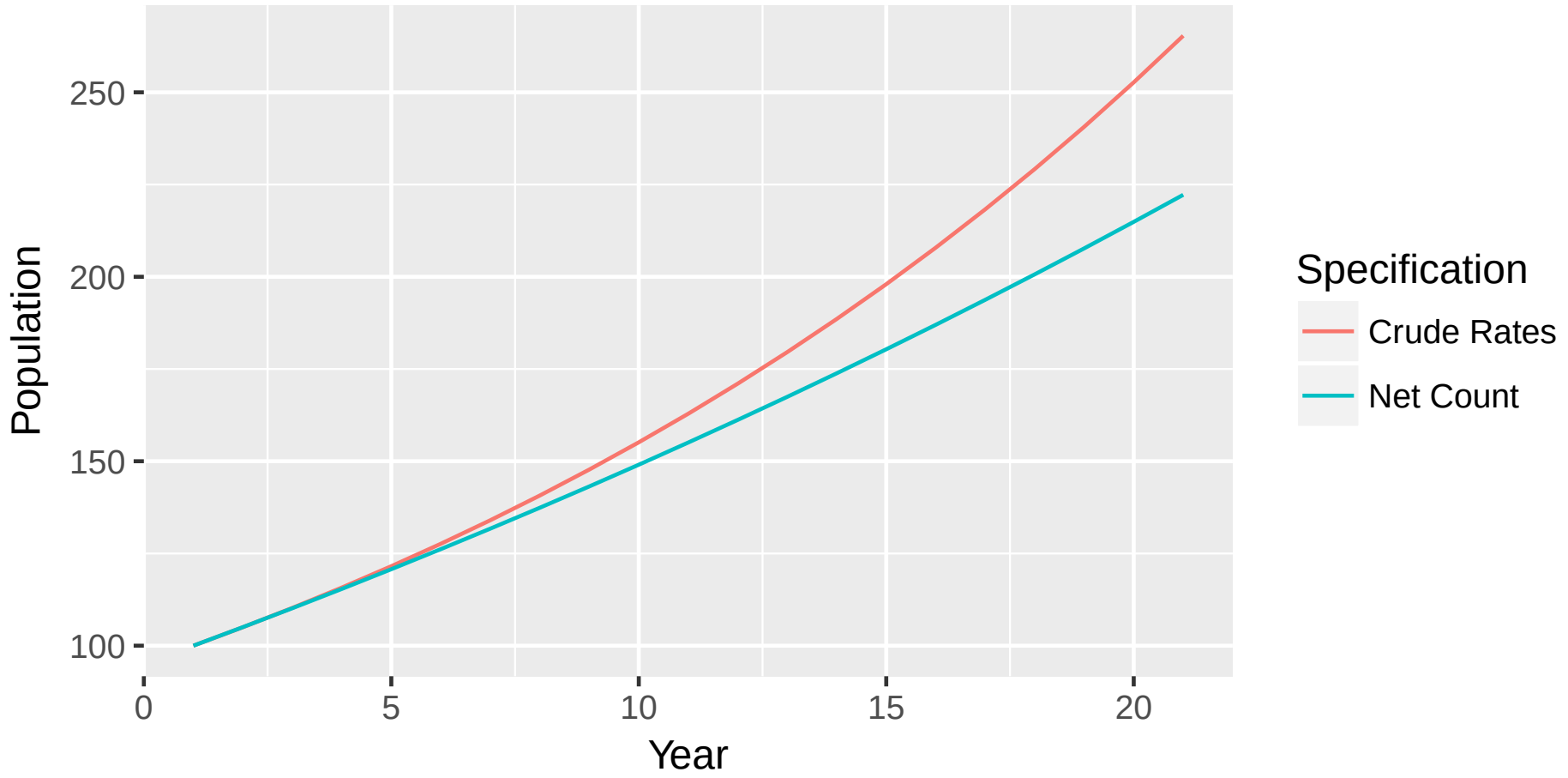
$$N(t+1) = \left(N(t) + \frac{M}{2} \right) \times (1 + b - d) + \frac{M}{2}$$

Exercise 2 (ex52.R)

```r
# 0.  a) Check your working directory is in the course folder

#     b) Souce the solution file for ex51.R

##
##
##
# 1. complete the population projection function based on net migration counts
pp_net <- function(n = #####, ##### = NULL, b = NULL, d = #####, ##### = NULL){
  NN <- rep(NA, times = ##### + 1)
  NN[#####] <- N0
  for(i in 1:n){
    NN[i+1] <- (NN[i] + #####/2) * (1 + ##### - d) + #####/#####
  }
  return(#####)
}
# 2. save the projection results when n = 20, N0 = 100, b = 0.04, d = 0.02 and M = 3
#    in an object called pp3
pp3 <- pp_net(n = 20, N0 = #####, b = 0.04, d = 0.02, M = #####)
pp3
# 3. create a new column in df0 with the results of pp3
df0 <- mutate(df0, `Net Count` = #####)
# 4. drop the growth rate projections (pp0 and pp1) from df0
df0 <- #####(df0, -contains("Growth"))
```

Net Migration Counts

```
> ggplot(data = df1,
+        mapping = aes(x = Year, y = Population, colour = Specification)) +
+   geom_line()
```



Net Migration Counts

- Different results despite setting net migration rate $m = 0.3$ and $M = 3$ with $N(0) = 100$
  - After the first projection period the net migration count model has a higher population (see `df0`)
  - Thereafter, the crude rates model has a higher projected population.
  - Caused by an increase on the population at risk in the later projection periods, leading to more migration.
- Rogers, A. (1990). *Requiem for the net migrant*. Geographical Analysis, 22(4), 283-300.
  - Difficult to set assumptions for future levels of net migration as is the product of both in and out migrations,
  - Net migration counts cannot respond to changes in the population at risk
  - Unlike many demographic rates, net migration has irregular age patterns and is volatile over time.

Net Migration Counts

- Net migration can be separated into immigration counts (I) and emigration rates (e),. . . if you have the data.

$$N(t+1) = N(t) \times (1 + b - d - e) + I$$

- Average immigration over the period:

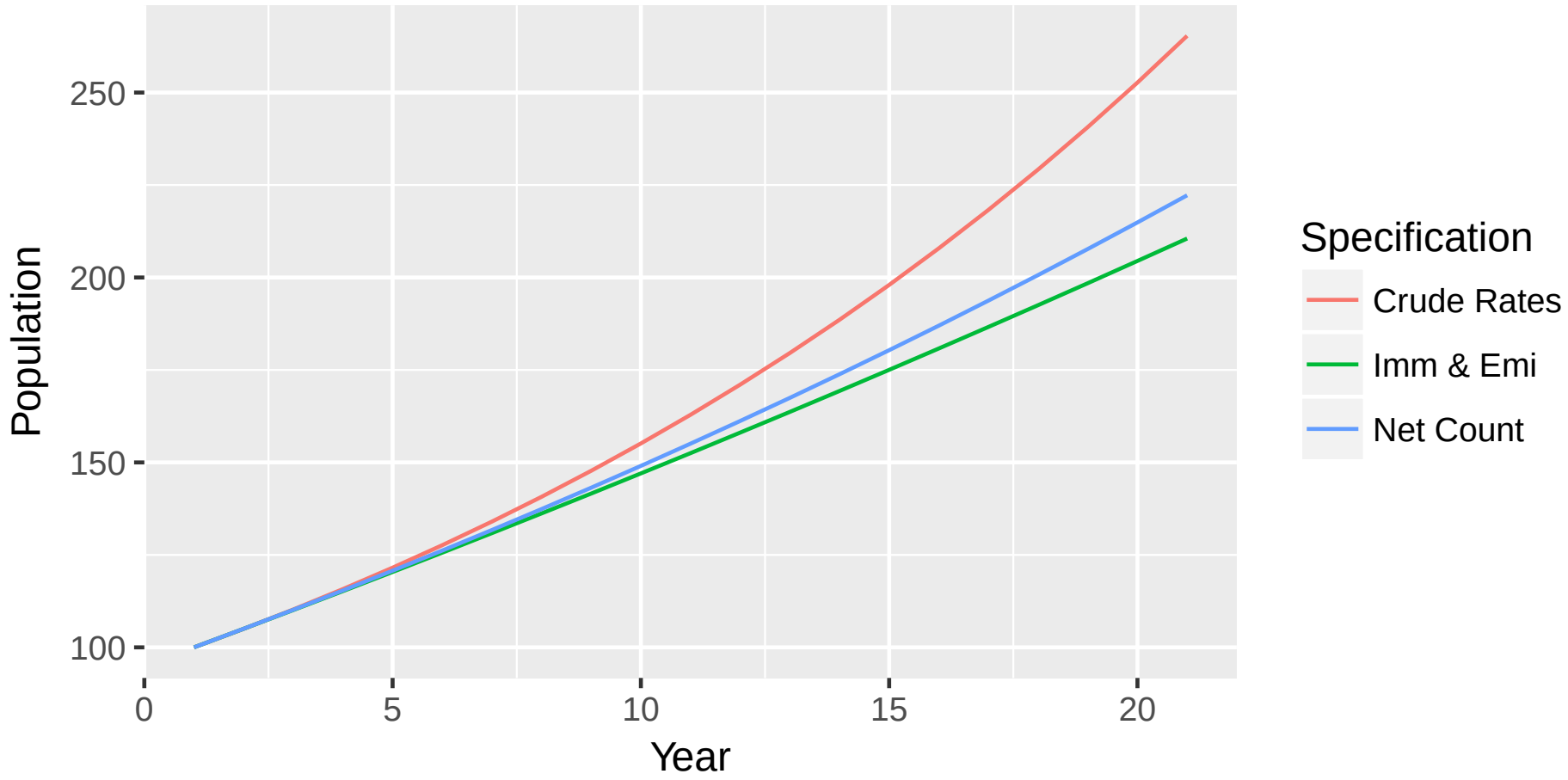
$$N(t+1) = \left(N(t) + \frac{I}{2}\right) \times (1 + b - d - e) + \frac{I}{2}$$

Exercise 3 (ex53.R)

```r
# 0.  a) Check your working directory is the course folder

#     b) Souce the solution file for ex52.R

##
##
##
# 1. complete the population projection function based on immigration counts and emi
pp_open <- function(n = NULL, N0 = NULL, b = NULL, d = NULL, e = NULL, I = NULL){
  NN <- rep(NA, times = n+1)
  NN[1] <- #####
  for(i in 1:n){
    NN[i+1] <- (NN[i] ##### #####/#####) ##### (1 + b - ##### - #####) + #####/2
  }
  return(#####)
}
# 2. Save the projection results when n = 20, N0 = 100, b = 0.04, d = 0.02, e = 0.01
#    in an object called pp4
pp4 <- pp_open(n = 20, ##### = 100, b = 0.04, ##### = 0.02, e = #####, ##### = 4)
pp4
# 3. create a new column in df0 with the results of pp4
df0 <- mutate(df0, "Imm & Emi" = #####)
df0
# 4. create a new data set df1 that is the tidy data version of df0
```

Decomposing Net Migration

```
> ggplot(data = df1,
+        mapping = aes(x = Year, y = Population, colour = Specification)) +
+   geom_line()
```



Decomposing Net Migration

- Again, different results despite setting net migration rate $m = 0.3$, net migration count to $M = 3$ or immigration count $I = 4$ and emigration rate $e = 0.01$ with $N(0) = 100$
  - The projected population in the open model is consistently lower than from the net count model.
  - Caused by the emigration rate lowering the population at risk.

Time-Specific Future Assumptions

- Typically we want to project with non-constant future rates. We can add a t subscript to bring time-specific assumptions:

$$N(t+1) = \left(N(t) + \frac{I_{t,t+1}}{2}\right) \times (1 + b_{t,t+1} - d_{t,t+1} - e_{t,t+1}) + \frac{I_{t,t+1}}{2}$$

- This specification can help answer some typical what-ifs?
  - Continuation of past trends.
  - Convergence to replacement fertility.
  - Convergence to zero migration or some other migration target.
- Set up some dynamic demographic scenarios
  1. Birth rate scenario where $b = 0.04$ initially and then linearly increases over the 20 year period to 0.06.
  2. Immigration scenario where $I = 4$ for first 10 periods and then I converges to zero (at 20 steps ahead) thereafter.

Time-Specific Future Assumptions

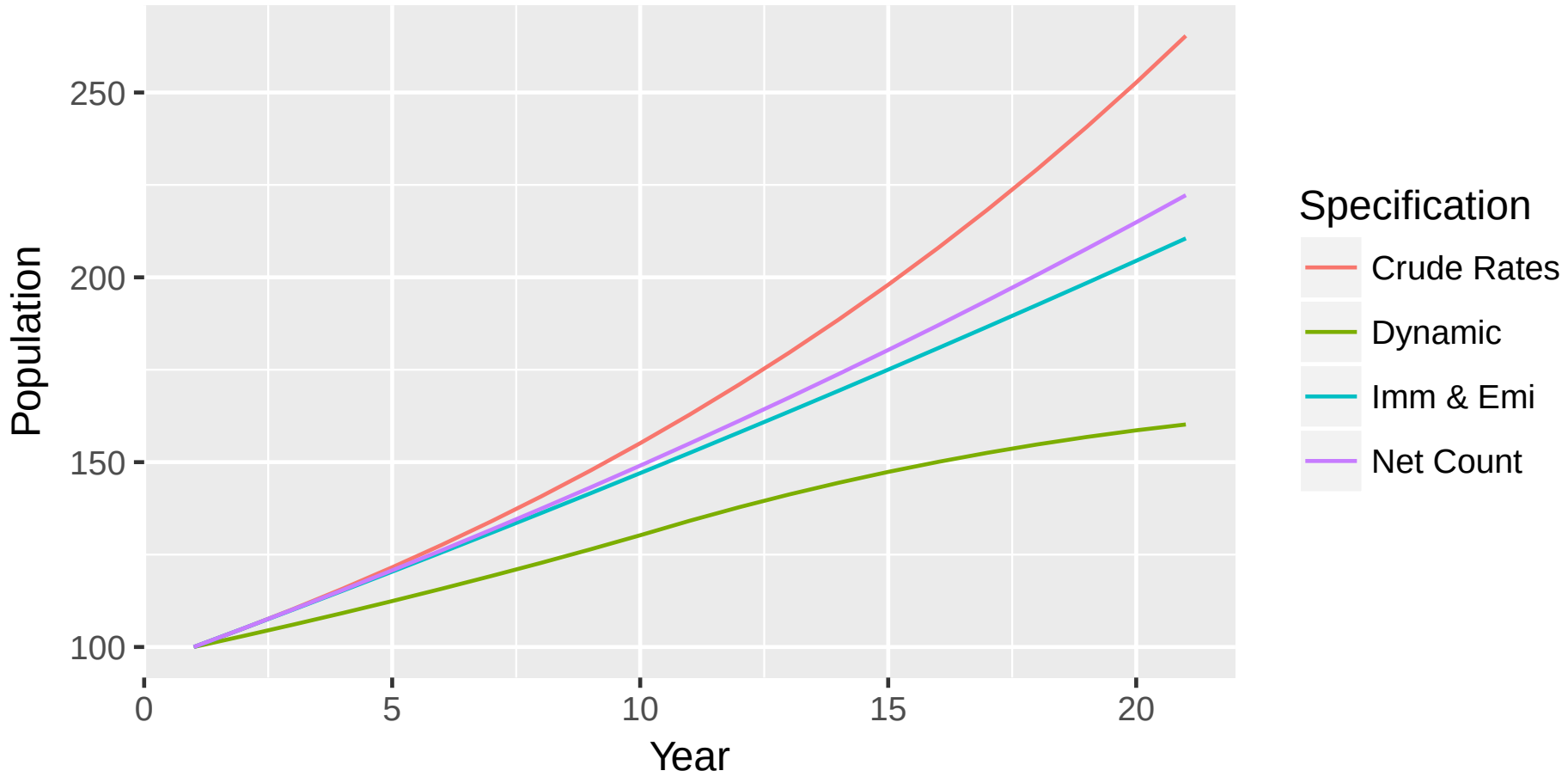
```
> #birth rate from 0.04 to 0.06
> br <- seq(from = 0.04, to = 0.06, length = 20)
> br
 [1] 0.04000000 0.04105263 0.04210526 0.04315789 0.04421053 0.04526316
 [7] 0.04631579 0.04736842 0.04842105 0.04947368 0.05052632 0.05157895
[13] 0.05263158 0.05368421 0.05473684 0.05578947 0.05684211 0.05789474
[19] 0.05894737 0.06000000
>
> #immigration counts constant for 9 periods
> imm1 <- rep(4, times = 9)
>
> #immigration decerase from 4 to 0
> imm2 <- seq(from = 4, to = 0, length = 11)
>
> #combine
> imm <- c(imm1, imm2)
> imm
 [1] 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 3.6 3.2 2.8 2.4 2.0 1.6 1.2
[18] 0.8 0.4 0.0
```

Exercise 4 (ex54.R)

```
# 0.  a) Check your working directory is the course folder

#     b) Souce the solution file for ex53.R

##
##
##
# 1. complete the dynamic population projection function based on immigration counts
pp_open <- function(n = #####, N0 = NULL, b = NULL, d = NULL, ##### = NULL, I = NUL
  NN <- rep(NA, ##### = n+1)
  NN[1] <- N0
  for(i in 1:n){
    NN[i+#####] <- (NN[i] + #####[i]/2) * (1 + b[#####] - #####[i] ##### e[i]) + I[
  }
  return(NN)
}
# 2. Uncomment and run the birth rate and scenario code
# br <- seq(from = 0.04, to = 0.06, length = 20)
# imm1 <- rep(4, times = 9)
# imm2 <- seq(from = 4, to = 0, length = 11)
# imm <- c(imm1, imm2)
# 3. Save the projection results when b = br, d = rep(0.02, 20), e = rep(0.01, 20) a
#    in an object called pp5
pp5 <- pp_open(n = 20, N0 = 100,
```

Time-Specific Future Assumptions

```
> ggplot(data = df1,
+        mapping = aes(x = Year, y = Population, colour = Specification)) +
+   geom_line()
```



Concluding Remarks

- Review of simple projection functions:

Function	Model
`pp_rate`	Simple projection with crude growth rate
`pp_demo`	Simple projection with crude demographic growth rates
`pp_net`	Simple projection with net migration counts
`pp_open`	Dynamic simple projection with immigration counts and emigration rates

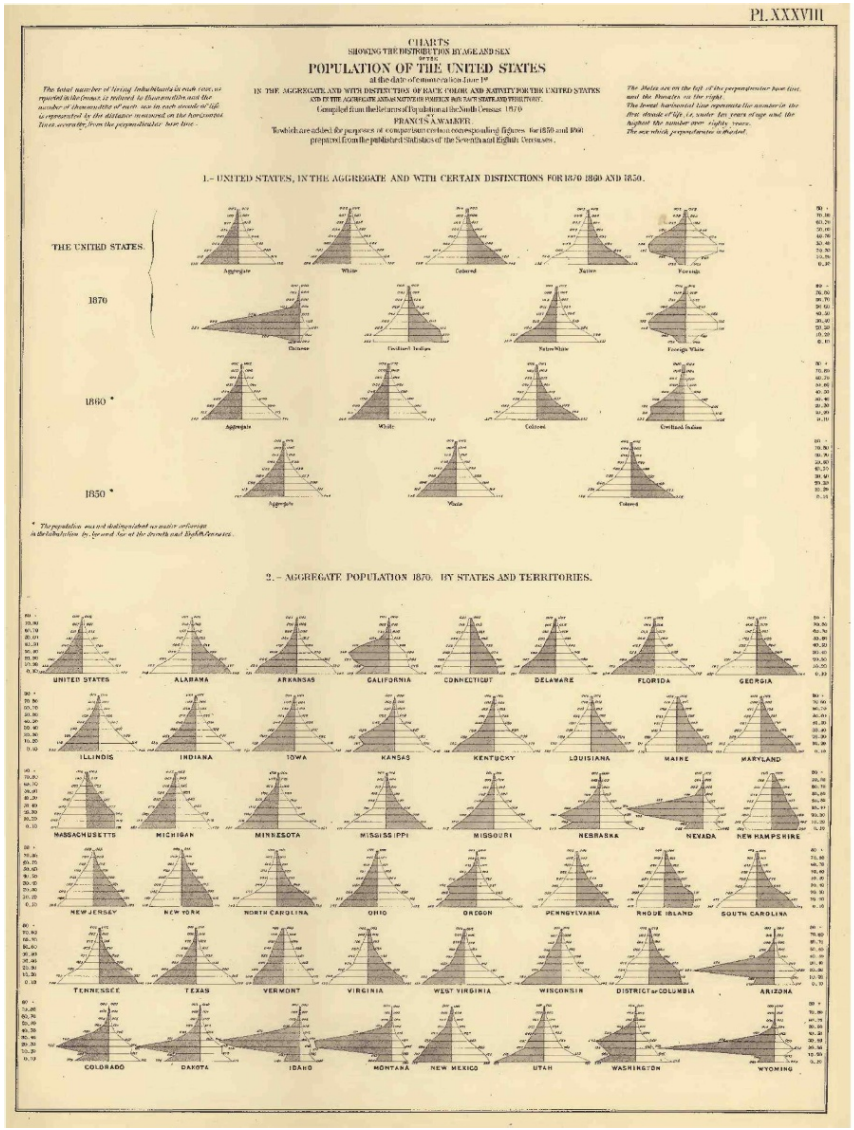
Concluding Remarks

- These simple projection models show the underlying structure of R function required to multi-step projection functions. We will build upon them for cohort component projections.
- Simple projection models are typically not used for projections.However
  - Might be the only option if data is limited.
  - Have been shown in some cases to be more accurate than more complex cohort component projection models (Stoto, 1983).
- Sensitivity to how you specify migration in the projection model.

Population Pyramids

- Standard way to visualize cohort component population projections.
  - Walker, Francis A., ed. (1874). *Statistical Atlas of the United States, Based on the Results of Ninth Census, 1870, with Contributions from Many Eminent Men of Science and Several Departments of the Federal Government.* New York: Julius Bien.

Population Pyramids

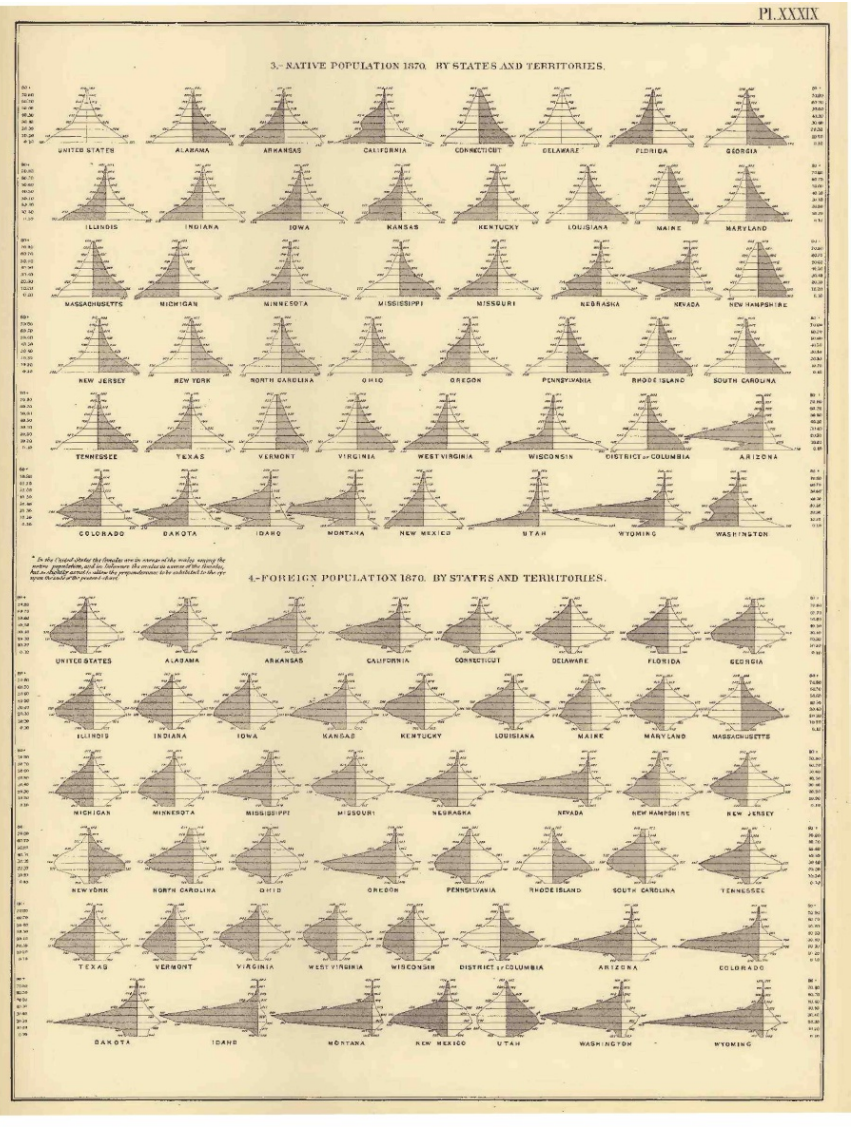

PL. XXXVIII

CHARTS
SHOWING THE DISTRIBUTION BY AGE AND SEX
OF THE
POPULATION OF THE UNITED STATES
at the date of consecutive Censuses from 1st
IN THE AGGREGATE AND WITH DISTINCTIONS OF RACE, COLOR AND NATIVITY FOR THE UNITED STATES AND BY THE AGGREGATE AND SEXES FOR FOREIGN AND NATIVE STATE AND TERRITORY.
Compiled from the figures of Publications of the Tenth Census 1870
BY
FRANCIS A. WALKER.
Tables are added for purposes of comparison of certain corresponding figures for 1870 and 1860 prepared from the published Statistics of the Seventh and Eighth Censuses.

1.- UNITED STATES, IN THE AGGREGATE AND WITH CERTAIN DISTINCTIONS FOR 1870 1860 AND 1850.

THE UNITED STATES.
1870
1860 *
1850 *

2.- AGGREGATE POPULATION 1870. BY STATES AND TERRITORIES.


PL. XXXIX

3.- NATIVE POPULATION 1870. BY STATES AND TERRITORIES.

4.-FOREIGN POPULATION 1870. BY STATES AND TERRITORIES.

Data

- Some good locations for current age-sex population data include:
  - The United Nations provides data on population by 5 year age groups for all countries.
    - Online: `http://esa.un.org/unpd/wpp/Download/`
    - R package, e.g. `wpp2017` on CRAN
  - Human Mortality Database has population by single year age groups
    - `http://www.mortality.org/`
- For a first attempt at a pyramid lets read in UN WPP data.
  - This 'tidy' data set was created from the data in the `wpp2017` package (see the script at the end of `s05_proj.R` for more details)
  - Take a subset of data to allow us to plot a pyramid for a country in a given year

Data

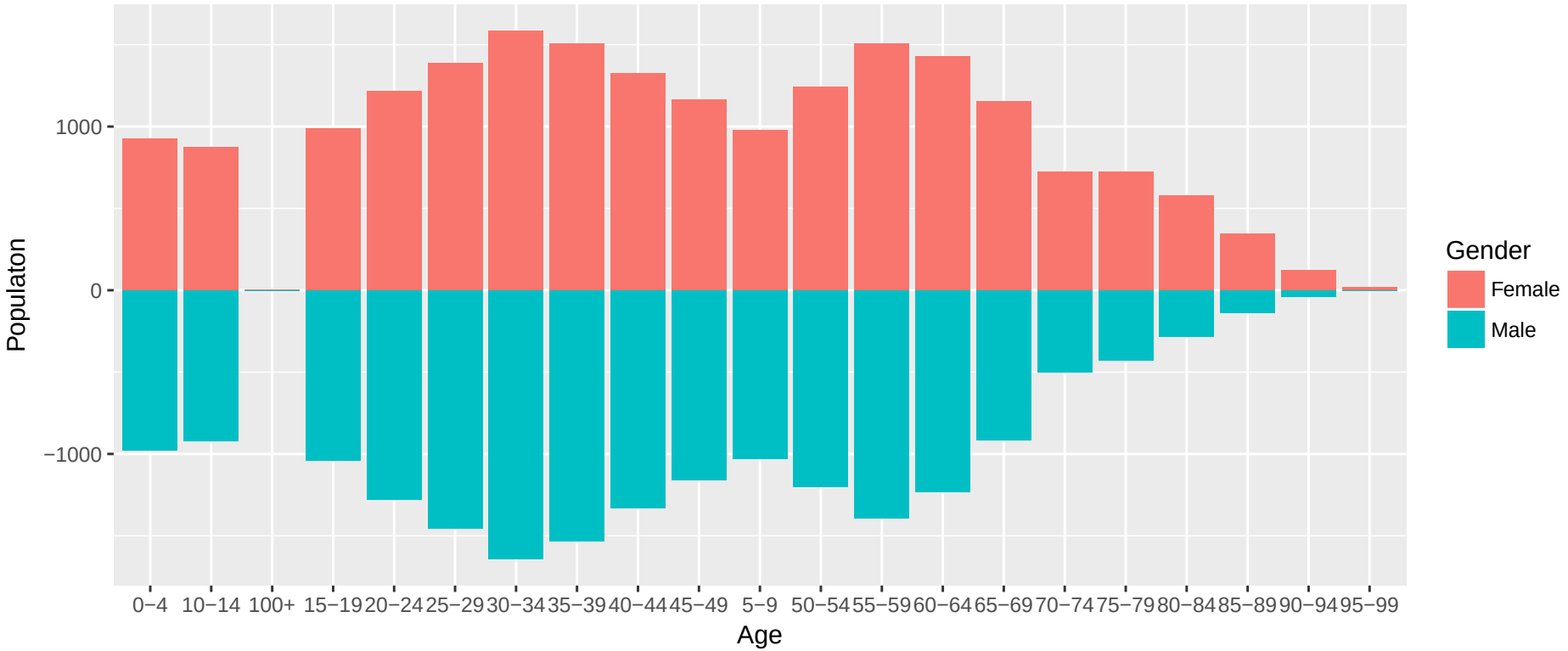
```
> setwd("C:/Users/Guy/Dropbox/SHU2017-3XS371026/slides-md/")
> df0 <- read_csv(file = "./data/wpp2017_pop.csv")
> # Poland 2015 5-year age group data from UN to plot
> df1 <- filter(df0, name == "Poland", year == 2015)
> df1
# A tibble: 42 x 6
   country_code   name   age gender  year      pop
          <int>  <chr> <chr>  <chr> <int>    <dbl>
 1          616 Poland   0-4   Male  2015  976.855
 2          616 Poland   5-9   Male  2015 1027.692
 3          616 Poland 10-14   Male  2015  918.667
 4          616 Poland 15-19   Male  2015 1037.925
 5          616 Poland 20-24   Male  2015 1277.581
 6          616 Poland 25-29   Male  2015 1452.508
 7          616 Poland 30-34   Male  2015 1642.547
 8          616 Poland 35-39   Male  2015 1531.170
 9          616 Poland 40-44   Male  2015 1329.083
10          616 Poland 45-49   Male  2015 1156.905
# ... with 32 more rows
```

Plotting Pyramids

- Using geom_bar() to build a pyramid.

```
> ggplot(data = df1,
+        mapping = aes(x = age,
+                      y = ifelse(gender == "Male", -pop, pop), fill = gender)) +
+   # bar length based on the value in the y variable not count (default)
+   geom_bar(stat = "identity") +
+   labs(x = "Age", y = "Populaton", fill = "Gender")
```

Plotting Pyramids



Plotting Pyramids

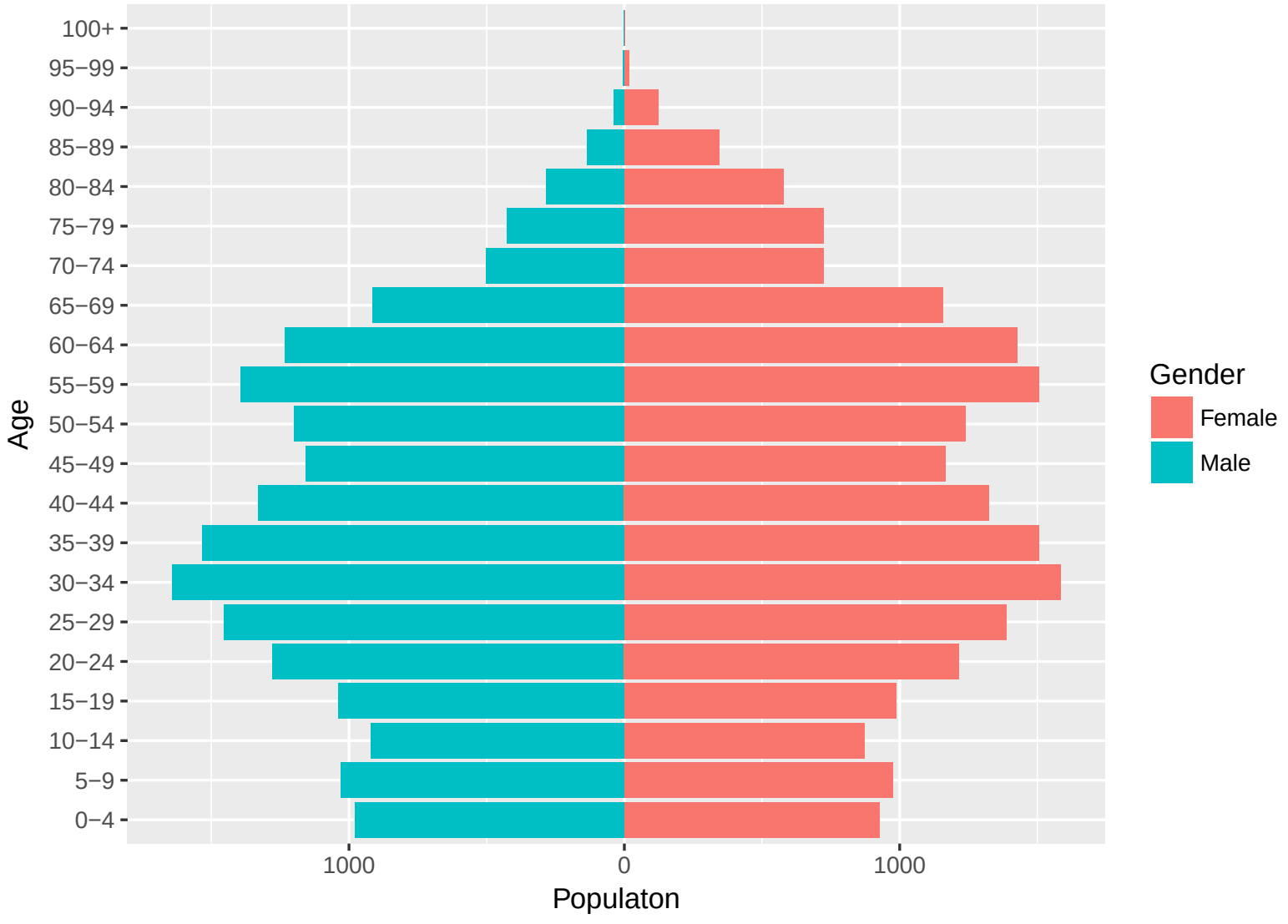
- Revisions
  1. The order of the age groups follows alpha-numeric behavior
  2. The geom_bar() only works with vertical bars
  3. The axis labels for the male population have a negative sign

```
> library(forcats)
> df0 <- mutate(df0, age = fct_inorder(age))
> df1 <- filter(df0, name == "Poland", year == 2015)
> df1
# A tibble: 42 x 6
   country_code   name    age gender  year      pop
          <int>  <chr> <fctr>  <chr> <int>    <dbl>
 1          616 Poland    0-4   Male  2015  976.855
 2          616 Poland    5-9   Male  2015 1027.692
 3          616 Poland  10-14   Male  2015  918.667
 4          616 Poland  15-19   Male  2015 1037.925
 5          616 Poland  20-24   Male  2015 1277.581
 6          616 Poland  25-29   Male  2015 1452.508
 7          616 Poland  30-34   Male  2015 1642.547
 8          616 Poland  35-39   Male  2015 1531.170
 9          616 Poland  40-44   Male  2015 1329.083
10          616 Poland  45-49   Male  2015 1156.905
# ... with 32 more rows
```

Plotting Pyramids

```
> ggplot(data = df1,
+        mapping = aes(x = age,
+                      y = ifelse(gender == "Male", -pop, pop), fill = gender)) +
+   geom_bar(stat = "identity") +
+   #positive male labels
+   scale_y_continuous(labels = abs) +
+   #bars to go vertically, rather than horizontal (deafult)
+   coord_flip() +
+   labs(x = "Age", y = "Populaton", fill = "Gender")
```

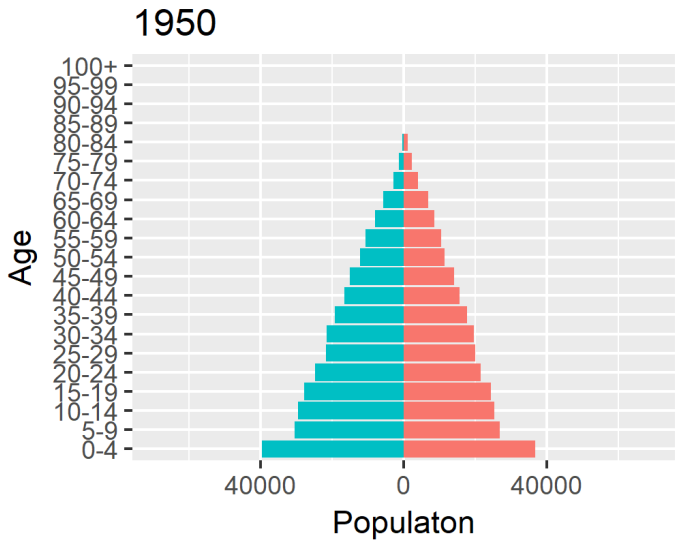
Plotting Pyramids

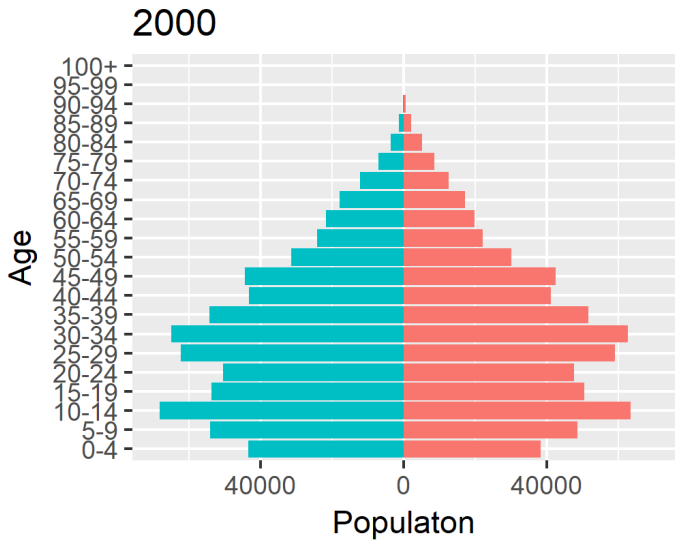


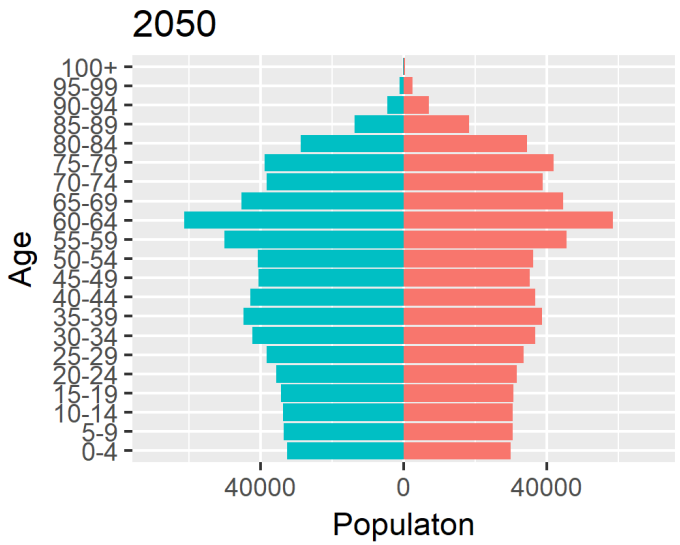
Exercise 5 (ex55.R)

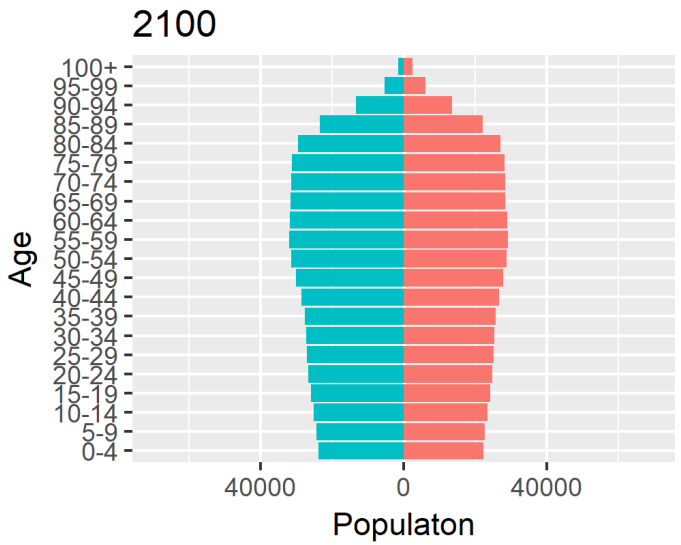
```
# 0. a) Check your working directory is the course folder

#    b) load the forcats package

##
##
##
# 1. load the data and convert the age column to a factor with levels in a sensible
df0 <- #####(file = "./exercise/data/wpp2017_pop.csv") %>%
  mutate(age = #####(age))
# 2. create a pyramid function that has the x-axis fixed to the population of the
#    largest age group in an year.
pyrm <- function(data = NULL, m = max(data$pop)){
  # m argument above defaults to maximum population in data
  # code below is same as before except for the limits on y axis
  ggplot(data = data,
         mapping = aes(x = #####, fill = gender,
                       y = ifelse(gender == "Male", yes = #####, no = pop))) +
    geom_bar(stat = #####) +
    #positive male labels
    scale_y_continuous(labels = #####, limits = c(-m, m)) +
    #bars to go vertically, rather than horizontal (deafult)
    coord_#####() +
    labs(x = "Age", y = "Populaton", fill = "Gender")
}
```

Plotting Pyramids









Plotting Pyramids

- If we want to show the outline of a past pattern we can use `geom_step()` and shift the steps to occur

```
> # all past data
> df2 <- filter(df0, name == "Poland", year <= 2015)
>
> ggplot(data = df2,
+        mapping = aes(y = ifelse(test = gender == "Male", yes = -pop, no = pop),
+                      x = age, fill = gender)) +
+   geom_bar(data = filter(df2, year == 2015),
+            stat = "identity") +
+   geom_step(data = filter(df2, year == 1950),
+             mapping = aes(x = as.numeric(age) - 0.5),
+             colour = "black") +
+   scale_y_continuous(labels = abs) +
+   labs(x = "Age", y = "Populaton", fill = "Gender") +
+   coord_flip()
```

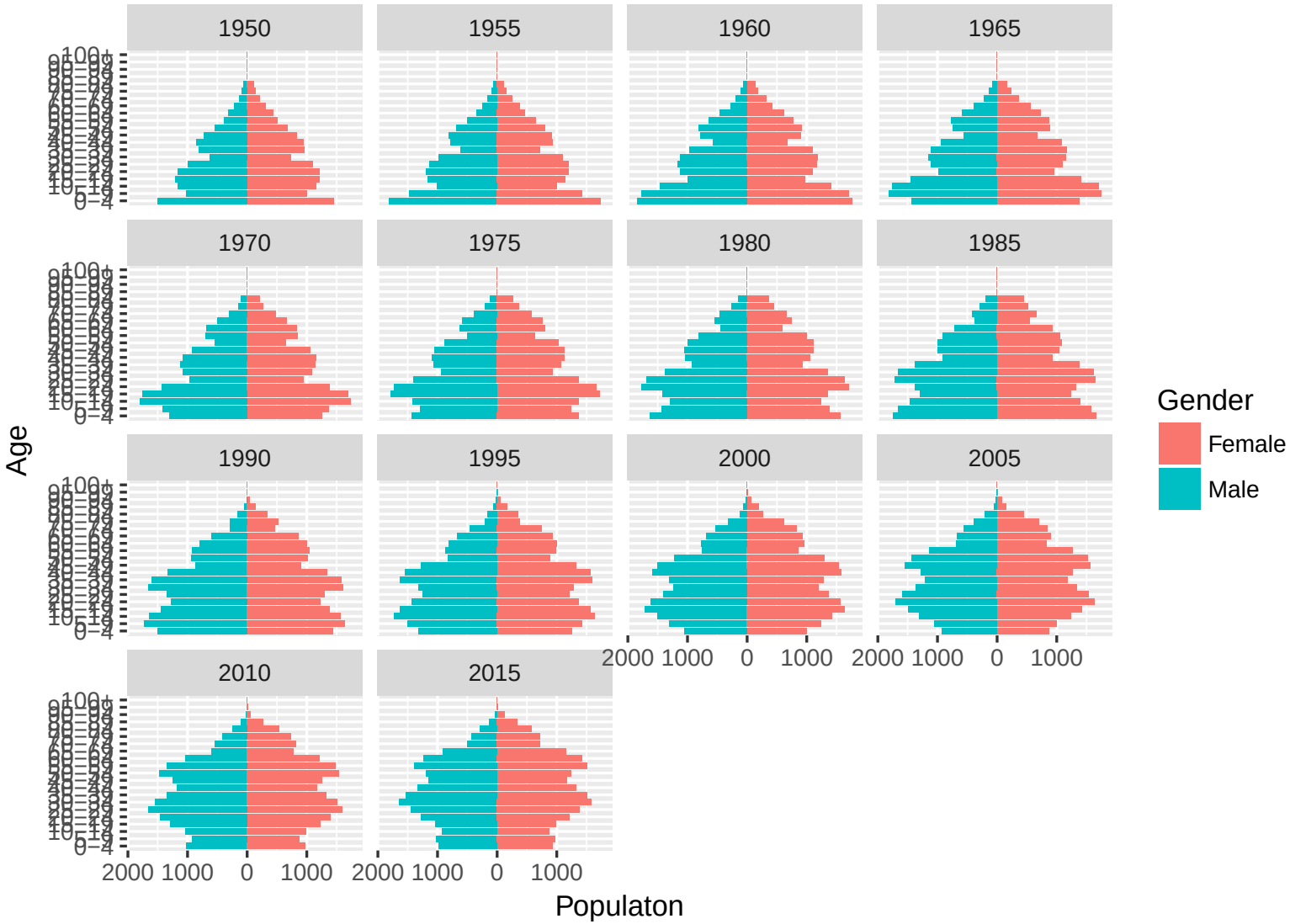
Plotting Pyramids

Plotting Pyramids

- If we want to show all pyramids we can use the facet_wrap()‘ function occur

```
> # all past data
> df2 <- filter(df0, name == "Poland", year <= 2015)
>
> ggplot(data = df2,
+        mapping = aes(y = ifelse(test = gender == "Male", yes = -pop, no = pop),
+                      x = age, fill = gender)) +
+   facet_wrap("year") +
+   geom_bar(stat = "identity") +
+   scale_y_continuous(labels = abs) +
+   labs(x = "Age", y = "Populaton", fill = "Gender") +
+   coord_flip()
```

Plotting Pyramids



Plotting Pyramids

- We can use the same pyrm() function to plot single year data.
  - Try with human mortality database in `austria.xlsx`.

```
> library(readxl)
> df3 <- read_excel("./data/austria.xlsx", sheet = 3, skip = 2)
> df3
# A tibble: 7,659 x 5
    Year   Age   Female     Male     Total
   <dbl> <chr>    <dbl>    <dbl>     <dbl>
 1  1947     0 48987.36 51872.47 100859.83
 2  1947     1 45681.51 47188.07  92869.58
 3  1947     2 50460.33 51856.40 102316.73
 4  1947     3 53938.60 55650.06 109588.66
 5  1947     4 56456.41 58449.71 114906.12
 6  1947     5 62287.84 64576.18 126864.02
 7  1947     6 66425.01 68811.12 135236.13
 8  1947     7 61339.49 63553.86 124893.35
 9  1947     8 49707.56 51523.64 101231.20
10  1947     9 42890.95 44507.52  87398.47
# ... with 7,649 more rows
```

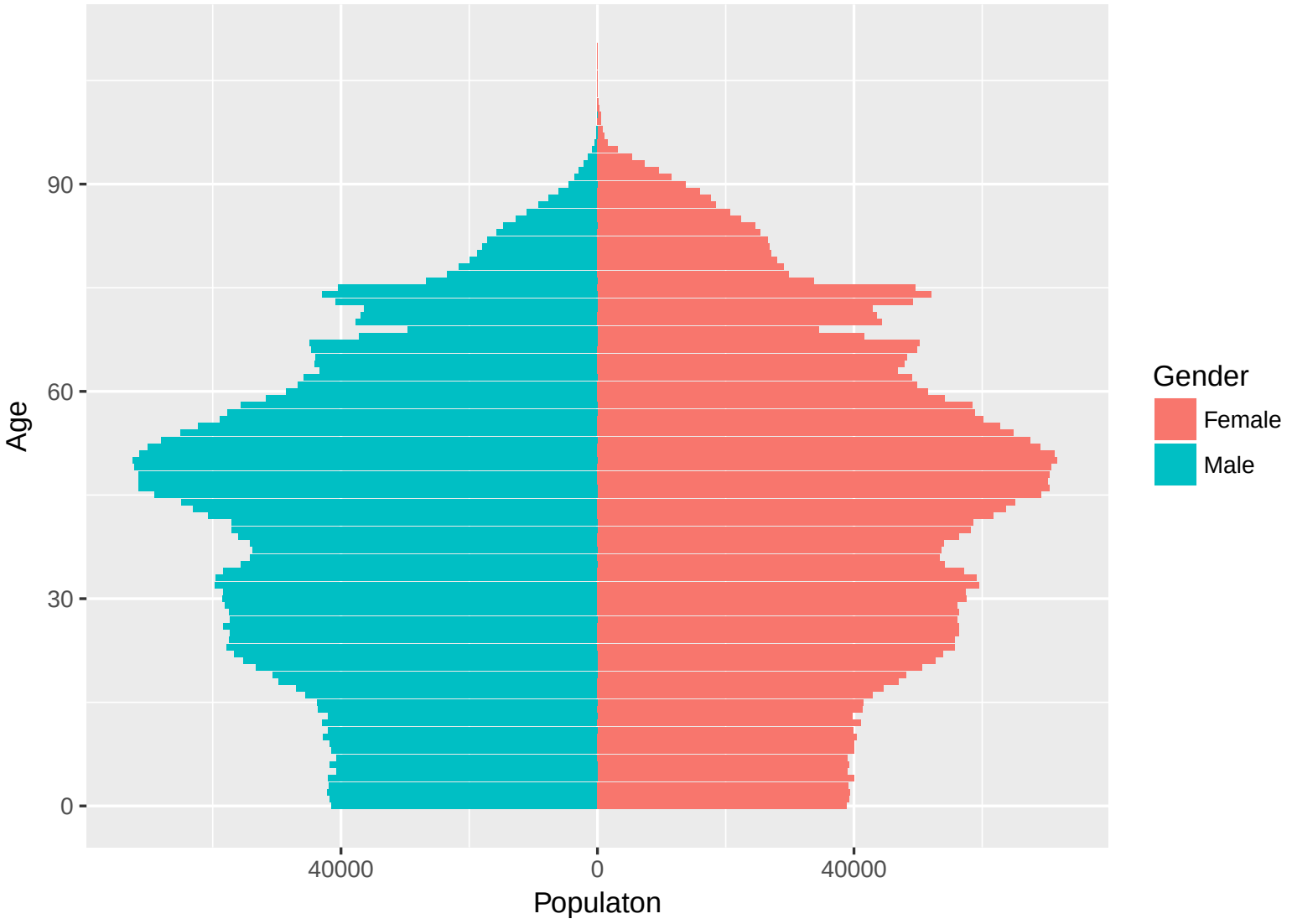
Plotting Pyramids

```
> # notice age is a character string (because of 110+)
> tail(df3, 5)
# A tibble: 5 x 5
   Year   Age Female  Male Total
  <dbl> <chr>  <dbl> <dbl> <dbl>
1  2015   106  12.55  2.37 14.92
2  2015   107   5.74  0.60  6.34
3  2015   108   4.11  0.00  4.11
4  2015   109   2.24  0.00  2.24
5  2015  110+   0.60  0.00  0.60
```

Plotting Pyramids

```
> # change Age column to numeric and format for pyramid function (tidy, lower case n
> df4 <- df3 %>%
+   select(-Total) %>%
+   mutate(Age = ifelse(Age == "110+", 110, Age),
+          Age = as.numeric(Age)) %>%
+   gather(key = gender, value = pop, -Year, -Age) %>%
+   rename(age = Age, year = Year)
> df5 <- filter(df4, year == 2015)
> df5
# A tibble: 222 x 4
    year   age gender   pop
   <dbl> <dbl>  <chr> <dbl>
 1  2015     0 Female 38920
 2  2015     1 Female 39284
 3  2015     2 Female 39381
 4  2015     3 Female 39135
 5  2015     4 Female 39995
 6  2015     5 Female 38931
 7  2015     6 Female 39221
 8  2015     7 Female 39002
 9  2015     8 Female 40030
10  2015     9 Female 40087
# ... with 212 more rows
> pyrm(data = df5)
```

Plotting Pyramids



Dynamic Pyramids

- There are a few ways in R to generate dynamic plots.
  - The `gganimate` package by David Robinson builds frames for each plot and then bounds them into a .gif
  - The `shiny` package by Winston Chang (and others) allows R users to create dynamic web applications
- The `gganimate` provides a nice ggplot functionality for the `animation` package created by Yihui Xie.
  - The animation package requires ImageMagick software to build the .gif files.
  - Install from `https://www.imagemagick.org/script/download.php`
  - Or can use the `installr` package in R (not RStudio)

```
> # install intstallr
> # install.packages("installr")
> # library(installr)
>
> # install imagemagick
> # install.imagemagick()
>
> # install gganimate from David Robinsons github (depends on animation)
> # library(devtools)
> # install_github("dgrtwo/gganimate")
```

gganimate

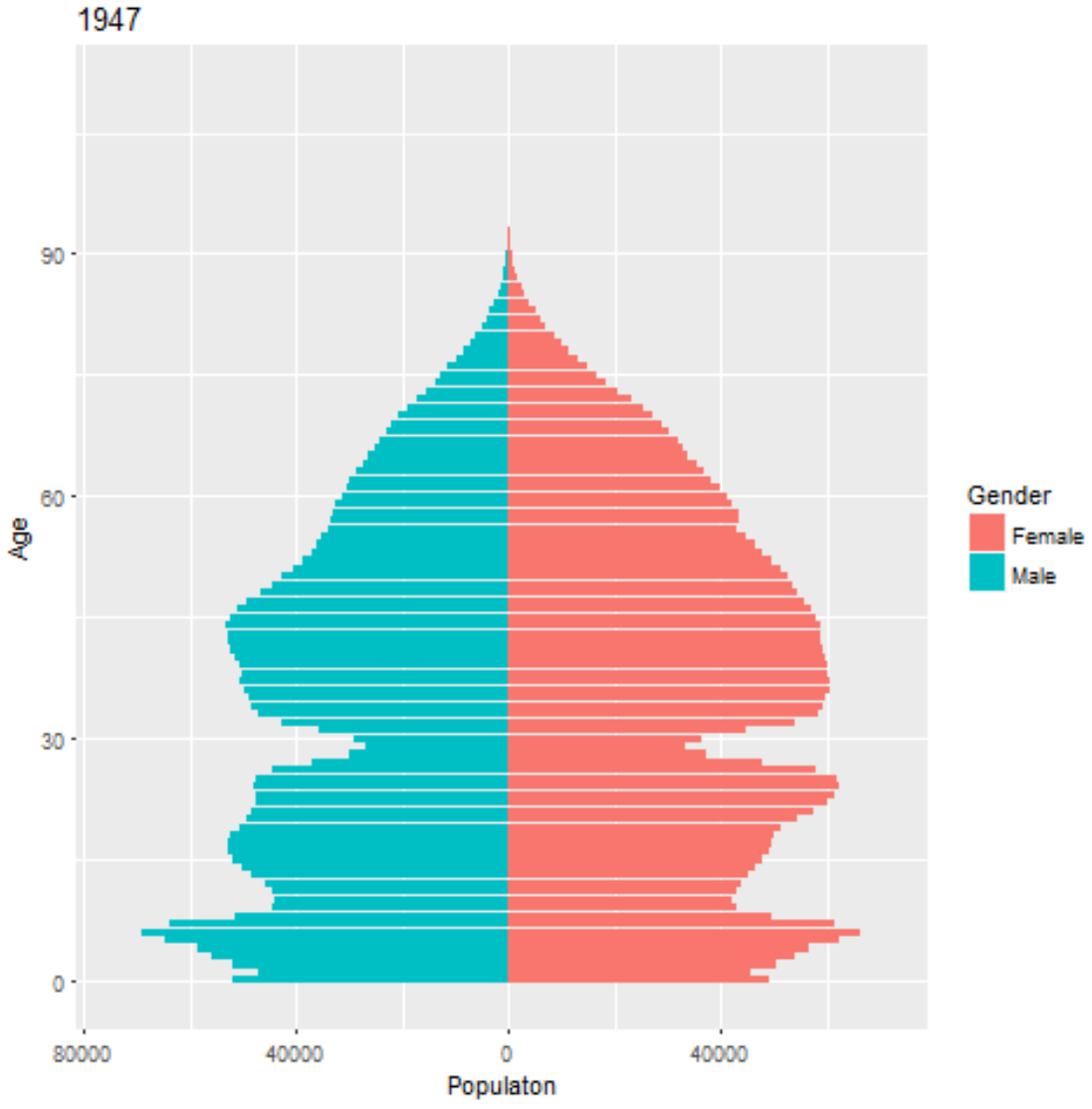
- The gganimate package adds a `frame` mapping in the aesthetic.
- The code otherwise looks much the same.
  - Need to save the plot as an R object read to pass to the `gganimate()` function
- Important `gganimate()` arguments
  - `filename`: name of `.gif` file for saved image
  - `fps`: frames per second of the gif
  - `ani.options`: animation options via the `ani.options()` function, in case function cannot find magick on your computer... requires the `animation` package is also loaded
- Generated gif will display in RStudio (not in my PDF)

```
> library(gganimate)
> g <- ggplot(data = df5,
+             mapping = aes(y = ifelse(test = gender == "Male", yes = -pop, no = po
+                           x = age, fill = gender, frame = year)) +
+   geom_bar(stat = "identity",
+            position = "identity") +
+   scale_y_continuous(labels = abs) +
+   coord_flip() +
+   labs(x = "Age", y = "Populaton", fill = "Gender")
```

gganimate

```
> library(animation)
> # tell R where magick is on your computor (not always necessary)
> magickPath <- "C:\\Program Files\\ImageMagick-7.0.6-Q16\\magick.exe"
>
> # create (displays in RStudio)
> gganimate(p = g, fps = 0.1, ani.options = ani.options(convert = magickPath))
>
> # save gif
> gganimate(p = g, filename = "./plot/austria.gif", fps = 0.1)
> file.show("./plot/austria.gif")
```

gganimate


1947
90
60
30
0
Age
80000
40000
0
40000
Populaton
Gender
Female
Male

Shiny

- Full interactive web based applications can be created using the shiny package.
- Apps need two R objects:
  1. User Interface: Provides details on the layout of the app and converts them to JavaScript. Can take different forms, we will use the `fluidPage()` layout
  2. Server that does all the R work given the user interface inputs
- The user interface and server are linked to together by two lists:
  - `input` containing elements created in the user interface
  - `output` containing elements created in the server
- The dynamic plot is created using the `shinyApp()` function.

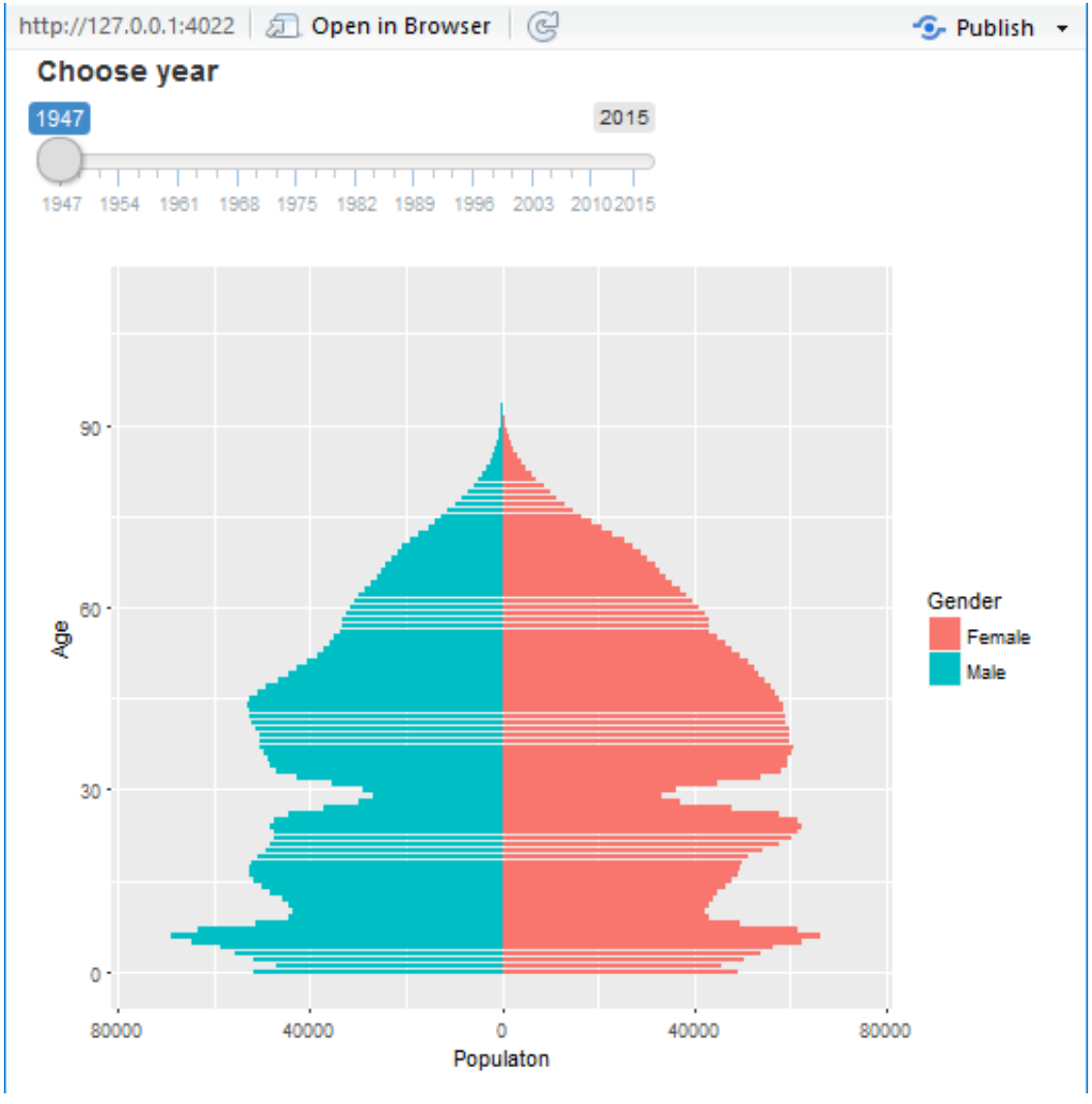
User Interface

```
> library(shiny)
>
> my_ui <- fluidPage(
+   #at the top of the page is a slider which we call year
+   sliderInput(
+     inputId = "year",
+     #label to appear:
+     label = "Choose year",
+     #on loading the slider will be set to the first year and move in 1 year steps
+     value = min(df1$year), step = 1,
+     #the slider range comes from the data
+     min = min(df1$year), max = max(df1$year),
+     #this get rid of a default `,` seperation on the slider values; "194,7"
+     sep = ""),
+   #after the slider comes the plot
+   plotOutput(outputId = "plot_pyr1")
+ )
```

Server

```
> my_server <- function(input, output){
+   #render a plot called plot_pyr1 based on our data and the year chosen by the
+   #user from the ui (slider) object called year
+   output$plot_pyr1 <- renderPlot({
+     df2 <- filter(df1, year == input$year)
+     pyrm(data = df2, m = max(df1$pop))
+   })
+ }
>
> # run the app
> shinyApp(ui = my_ui, server =  my_server)
```

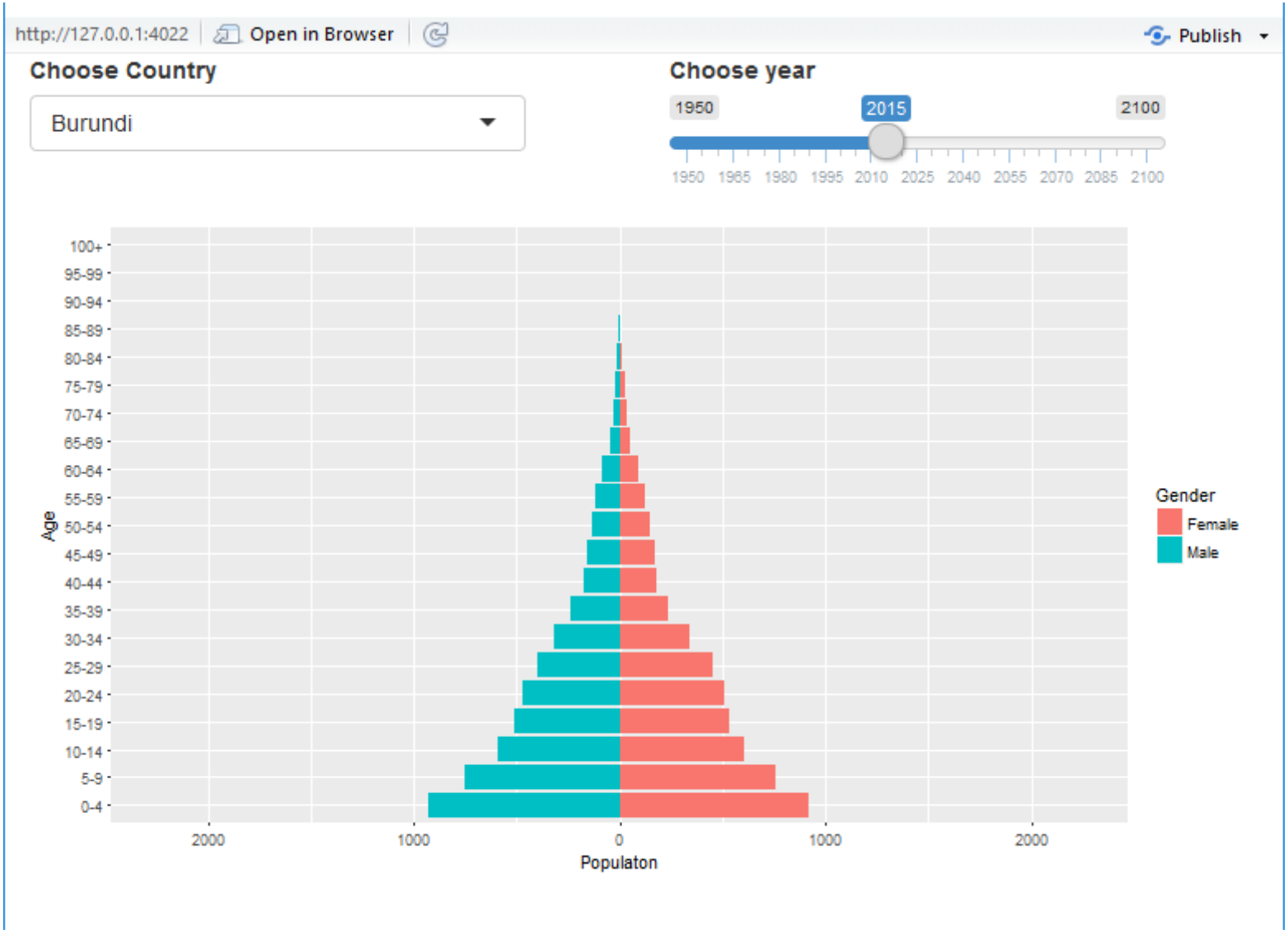
Shiny


http://127.0.0.1:4022
Open in Browser
Publish
Choose year
1947
2015
1947 1954 1961 1968 1975 1982 1989 1996 2003 2010 2015
90
60
30
0
Age
80000
40000
0
40000
80000
Populaton
Gender
Female
Male

Exercise 6 (ex56.R)

```
# 0. a) Check your working directory is the course folder

#    b) load the shiny package

##
##
##
# 1. uncomment to load the data
# df0 <- read_csv(file = "./exercise/data/wpp2017_pop.csv") %>%
#    mutate(age = factor(x = age, levels = unique(age)))
# 2. complete the user interface to show the UN data for all countries/areas
my_ui <- fluidPage(
  #a row on top of the plot
  fluidRow(
    #within the row two columns. shiny apps always based on a width of 12.
    column(
      width = 6,
      # a box of all the possible countries that the user can select from
      selectInput(inputId = 'country', label = 'Choose Country',
                  choices = unique(df0$#####))
    ),
    # slider in the second column in top row based the range of years in the data
    column(
      width = 6,
```

Shiny


http://127.0.0.1:4022
Open in Browser
Publish
Choose Country
Burundi
Choose year
1950
2015
2100
1950 1965 1980 1995 2010 2025 2040 2055 2070 2085 2100
100+
95-99
90-94
85-89
80-84
75-79
70-74
65-69
60-64
55-59
50-54
45-49
40-44
35-39
30-34
25-29
20-24
15-19
10-14
5-9
0-4
Age
2000
1000
0
1000
2000
Populaton
Gender
Female
Male

Publishing Apps Online

- You can publish shiny apps online for free to share with your projection results with the world.
  - `https://gjabel.shinyapps.io/austria-pop/`
  - `https://gjabel.shinyapps.io/wpp2017-pop/`

1. Go to `./shiny/wpp2017-pop/` folder. This folder is self-contained replication of exercise 55 including:
   1. `app.R` script with the `ui` and `server()` functions from exercise 56 with some extra lines to load the `pyrm()` function, data and libraries.
   2. `pyrm.R` script with the `pyrm()` function from exercise 55.
   3. `wpp2017_pop.csv`.
2. Open up the `app.R` file.
   - Hit the `Run` button on the top right of the script.
   - Hit the `Publish` button on the top right of the app.
3. Follow the instructions to sign up for a for shinyapps account and link the account to RStudio on your machine
   - Deploy the app
   - If you get lost, follow the guide: `http://shiny.rstudio.com/articles/shinyapps.html`

Publishing Apps Online

- Want to learn more shiny,. . .
  - Follow the tutorial to gain a better understanding of shiny apps. `http://shiny.rstudio.com/tutorial/`
  - Shiny Cheatsheet `http://shiny.rstudio.com/images/shiny-cheatsheet.pdf`
  - Check out what is possible with shiny apps `http://www.showmeshiny.com/`
  - Demographic application `http://www.oeaw.ac.at/vid/dataexplorer/`