

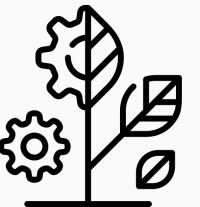
Tutorial 1

Data Visualization in R

ggplot2: plots and charts

Victoria Mironova

Associate Professor, Department of Plant Systems Physiology



Course structure

Week 5:

Lecture 1. Principles of figure design.

Quiz 1.

Week 6:

Tutorial 1. ggplot2: plots and charts.

Quiz 2.

Week 7:

Tutorial 2. ggplot2: statistics, coordinate system, facets.

Tutorial 3. ggplot2: themes and styles.

Practice 1.

Quiz 3.

Week 8:

Practice 2. Project.

Practice 3. Project.

Practice 4. Project.

Assignment.

Course structure

Week 5:

Lecture 1. Principles of figure design.

Quiz 1.

Week 6:

Tutorial 1. ggplot2: plots and charts.

Quiz 2.

Week 7:

Tutorial 2. ggplot2: statistics, coordinate system, facets.

Tutorial 3. ggplot2: themes and styles.

Practice 1.

Quiz 3.

Week 8:

Practice 2. Project.

Practice 3. Project.

Practice 4. Project.

Assignment.

Learning goals

- Understand the basic principles behind effective data visualization
- **Create data visualizations in R using ggplot2**
- Craft elegant visual presentations of data

Plan of the tutorial

- ggplot2
- Preparatory steps:
 - Install libraries
 - Get data into R
- Design basic plots with ggplot2
 - Visualize amounts
 - ... x-y relationship
 - ... distributions

DATA



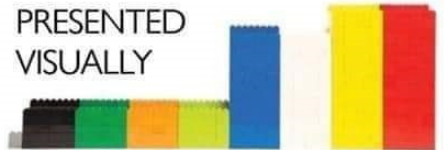
SORTED



ARRANGED



PRESENTED
VISUALLY



EXPLAINED
WITH A STORY

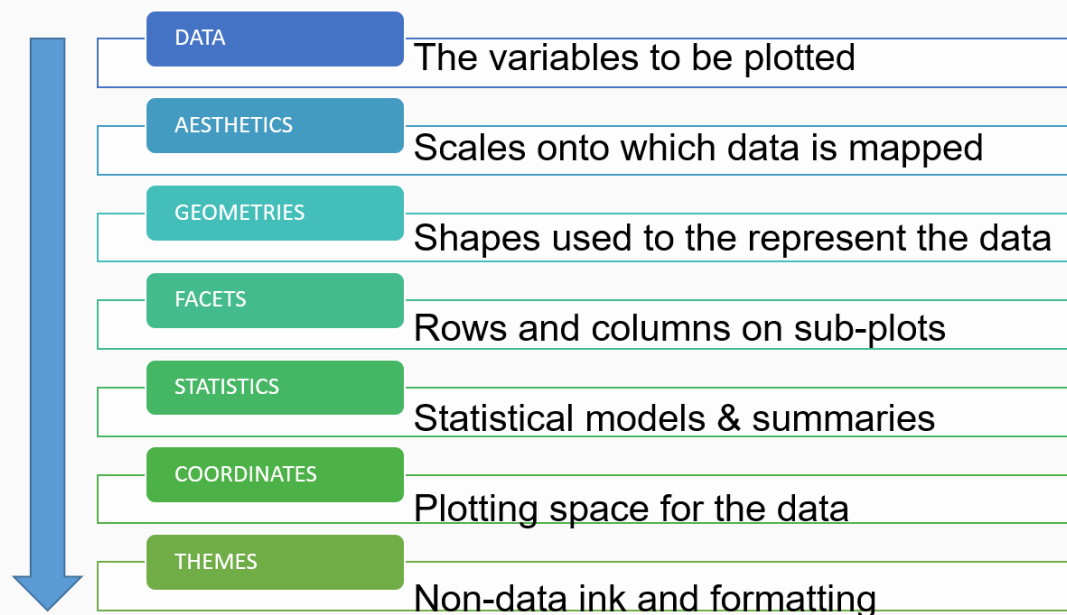


Data vizualization platforms:

- ...
- Microsoft Excel and analogs
- Matplotlib, Seaborn, and Plotly in Python
- **ggplot2 in R**



ggplot2 applies the grammar of graphics



Layers in grammar of graphics

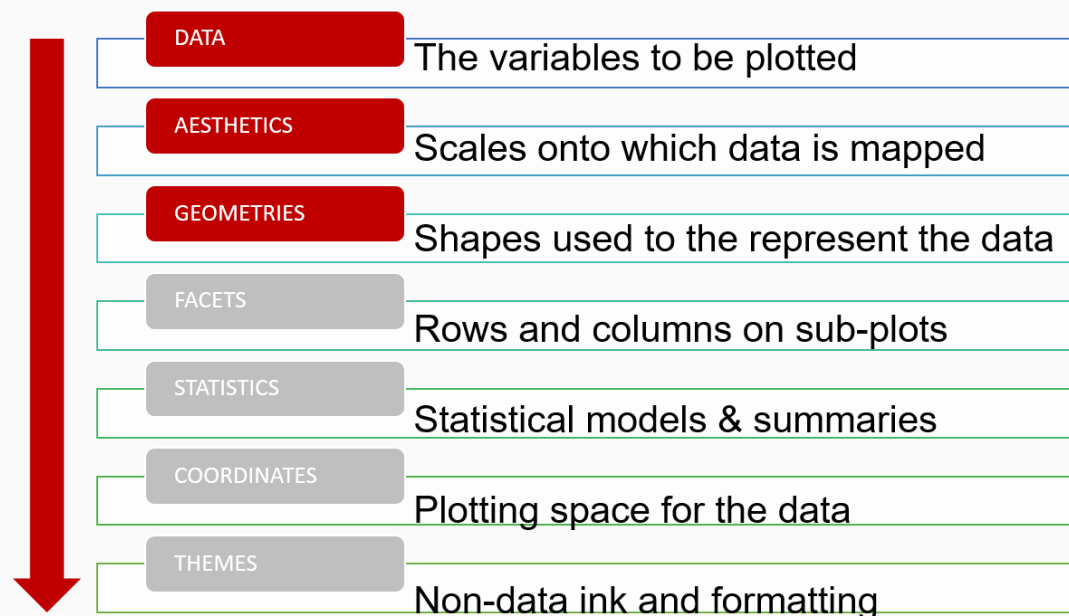
The *grammar of graphics* is a plotting framework developed by Leland Wilkinson (Grammar of Graphics, 1999) that dissects each component of a graph into individual layer.

There are two important principles:

- Graphics are made of distinct layers of grammatical elements
- Plots are built with appropriate aesthetic mappings to make these plots meaningful



Grammar of graphics



Layers in grammar of graphics

Three out of 7 layers are essential for any plot:

Data This is the dataset being plotted containing the variables to be plotted on the graph.

Aesthetics Aesthetics refers to the scales on which we map the data. Some common aesthetics to consider are axis, shape, size, and color.

Geometries Geom refers to the actual visual elements used for the data in the plot, such as points, lines, and bars.

ggplot2: a simplified graphing template

To make a graph, replace the bracketed sections in the following code with a *dataset*, a *geom* function, and a collection of *mappings*.

```
ggplot(data = <DATA>)+  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```


Preparatory steps:

- Install libraries
- Get data into R

Installing the packages

Installation of the packages in R studio:

```
my_packages <- c("tidyverse", "ggplot2")  
install.packages(my_packages)
```

Call the libraries to your file:

```
library(tidyverse)  
library(ggplot2)
```

Getting data into R

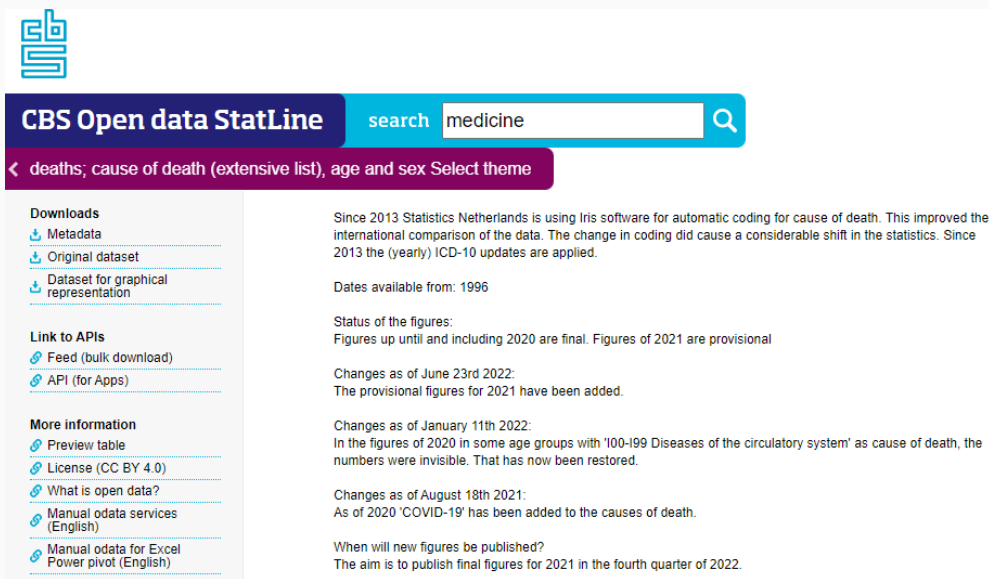
Number of deaths in the population of the Netherlands by main underlying cause of death, by age and sex, 1996-2021

https://opendata.cbs.nl/statline/portal.html?_la=en&_catalog=CBS&tableId=7233ENG&_theme=1120

Identifier: 7233ENG

Short title: deaths; cause of death (extensive list)

Reference period: 1996-2021



The screenshot displays the CBS Open data StatLine portal. At the top, there is a search bar with the text 'medicine' and a magnifying glass icon. Below the search bar, a purple banner shows the selected theme: '< deaths; cause of death (extensive list), age and sex Select theme'. The main content area is divided into two columns. The left column contains a sidebar with links under three categories: 'Downloads' (Metadata, Original dataset, Dataset for graphical representation), 'Link to APIs' (Feed (bulk download), API (for Apps)), and 'More information' (Preview table, License (CC BY 4.0), What is open data?, Manual odata services (English), Manual odata for Excel Power pivot (English)). The right column contains detailed information about the dataset, including a note about the use of Iris software for automatic coding, dates available from 1996, status of figures (up to 2020 are final, 2021 are provisional), and changes as of June 23rd 2022 and January 11th 2022.

CBS Open data StatLine search

< deaths; cause of death (extensive list), age and sex Select theme

Downloads

- Metadata
- Original dataset
- Dataset for graphical representation

Link to APIs

- Feed (bulk download)
- API (for Apps)

More information

- Preview table
- License (CC BY 4.0)
- What is open data?
- Manual odata services (English)
- Manual odata for Excel Power pivot (English)

Since 2013 Statistics Netherlands is using Iris software for automatic coding for cause of death. This improved the international comparison of the data. The change in coding did cause a considerable shift in the statistics. Since 2013 the (yearly) ICD-10 updates are applied.

Dates available from: 1996

Status of the figures:
Figures up until and including 2020 are final. Figures of 2021 are provisional

Changes as of June 23rd 2022:
The provisional figures for 2021 have been added.

Changes as of January 11th 2022:
In the figures of 2020 in some age groups with 'I00-I99 Diseases of the circulatory system' as cause of death, the numbers were invisible. That has now been restored.

Changes as of August 18th 2021:
As of 2020 'COVID-19' has been added to the causes of death.

When will new figures be published?
The aim is to publish final figures for 2021 in the fourth quarter of 2022.

Getting data into R

| Code | Table |
|------|-------|
|------|-------|

```
Death_in_NL ← read_csv2(file = "Data/7233ENG_TypedDataSet_04112022_143444.csv")
```

Getting data into R

| Code | Table |
|------|-------|
|------|-------|

```
knitr::kable(head(Death_in_NL), format = 'html')
```

| ID | Sex | Age | CausesOfDeath | Periods | Deaths_1 |
|---------|------|-------|---------------|----------|----------|
| 1022190 | 3000 | 10000 | A010668 | 1996JJ00 | 890 |
| 1022191 | 3000 | 10000 | A010668 | 1997JJ00 | 736 |
| 1022192 | 3000 | 10000 | A010668 | 1998JJ00 | 757 |
| 1022193 | 3000 | 10000 | A010668 | 1999JJ00 | 814 |
| 1022194 | 3000 | 10000 | A010668 | 2000JJ00 | 788 |
| 1022195 | 3000 | 10000 | A010668 | 2001JJ00 | 773 |

Formating the data

| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL <- Death_in_NL %>%
  mutate(Sex = recode(Sex, "3000" = "Male", "4000" = "Female")) %>%
  mutate(CausesOfDeath = recode(CausesOfDeath,
    "A010668" = "Infections", "A010840" = "Neoplasms", "A011013" = "Endocr+Metabol", "A011087" = "P
    "A011166" = "NervousSystem", "A011234" = "Eye+Adnexa", "A011307" = "CirculatorySystem", "A011384
    "A011449" = "DigestiveSystem", "A011521" = "SkinDeseases", "A011594" = "MusculSystem", "A011674
    "A011757" = "Pregnancy+Childbirth", "A011833" = "Perinatal", "A011893" = "CongenitalMalform", "
    "A050205" = "COVID-19", "A012072" = "ExternalCauses")) %>%
  mutate(Age = recode(Age,
    "10000" = "Total", "10010" = "0", "22000" = "95+", "51300" = "1-9", "70200" = "1-9", "70300" =
    "70600" = "25-29", "70700" = "30-34", "70800" = "35-39", "70900" = "40-44", "71000" = "45-49",
    "71400" = "65-69", "71500" = "70-74", "71600" = "75-79", "71700" = "80-84", "71800" = "85-89",
  mutate(Periods = recode(Periods,
    "1996JJ00" = "1996", "1997JJ00" = "1997", "1998JJ00" = "1998", "1999JJ00" = "1999", "2000JJ00"
    "2004JJ00" = "2004", "2005JJ00" = "2005", "2006JJ00" = "2006", "2007JJ00" = "2007", "2008JJ00"
    "2012JJ00" = "2012", "2013JJ00" = "2013", "2014JJ00" = "2014", "2015JJ00" = "2015", "2016JJ00"
    "2020JJ00" = "2020", "2021JJ00" = "2021")) %>%
  rename("Year" = "Periods", "Deaths" = "Deaths_1") %>%
  drop_na(Deaths) %>%
  mutate(Year = as.numeric(Year))
```

Formating the data

Code

Output

| ID | Sex | Age | CausesOfDeath | Year | Deaths |
|---------|--------|-------|--------------------|------|--------|
| 2957560 | Female | 85-89 | Perinatal | 2004 | 0 |
| 1424313 | Male | 30-34 | Perinatal | 2003 | 0 |
| 1457025 | Male | 35-39 | CirculatorySystem | 2007 | 110 |
| 1377865 | Male | 25-29 | Perinatal | 2017 | 0 |
| 2327624 | Female | 20-24 | Neoplasms | 1996 | 27 |
| 1985711 | Male | 90-94 | UnclassifiedAbnorm | 2009 | 179 |
| 2995890 | Female | 90-94 | SkinDeseases | 2010 | 48 |
| 2308669 | Female | 15-19 | CongenitalMalform | 2021 | 4 |
| 2575825 | Female | 45-49 | DigestiveSystem | 2001 | 53 |
| 2473438 | Female | 35-39 | Psychological | 2002 | 4 |

Data types:

| ID | Sex | Age | CausesOfDeath | Year | Deaths |
|---------|--------|-------|-------------------|------|--------|
| 1291232 | Male | 15-19 | ExternalCauses | 2016 | 61 |
| 2618583 | Female | 50-54 | CirculatorySystem | 2015 | 180 |
| 1317633 | Male | 20-24 | CirculatorySystem | 2001 | 13 |
| 2897348 | Female | 80-84 | CirculatorySystem | 2008 | 4317 |
| 2334070 | Female | 20-24 | Psychological | 2020 | 6 |

- **ID**: Numeric ordered;
- **Sex**: Factor, unordered;
- **Age**: Factor, unordered;
- **CausesOfDeath**: Factor, unordered;
- **Year**: Numeric, ordered;
- **Deaths**: Numeric, unordered.

Build your first plot

Code

Output

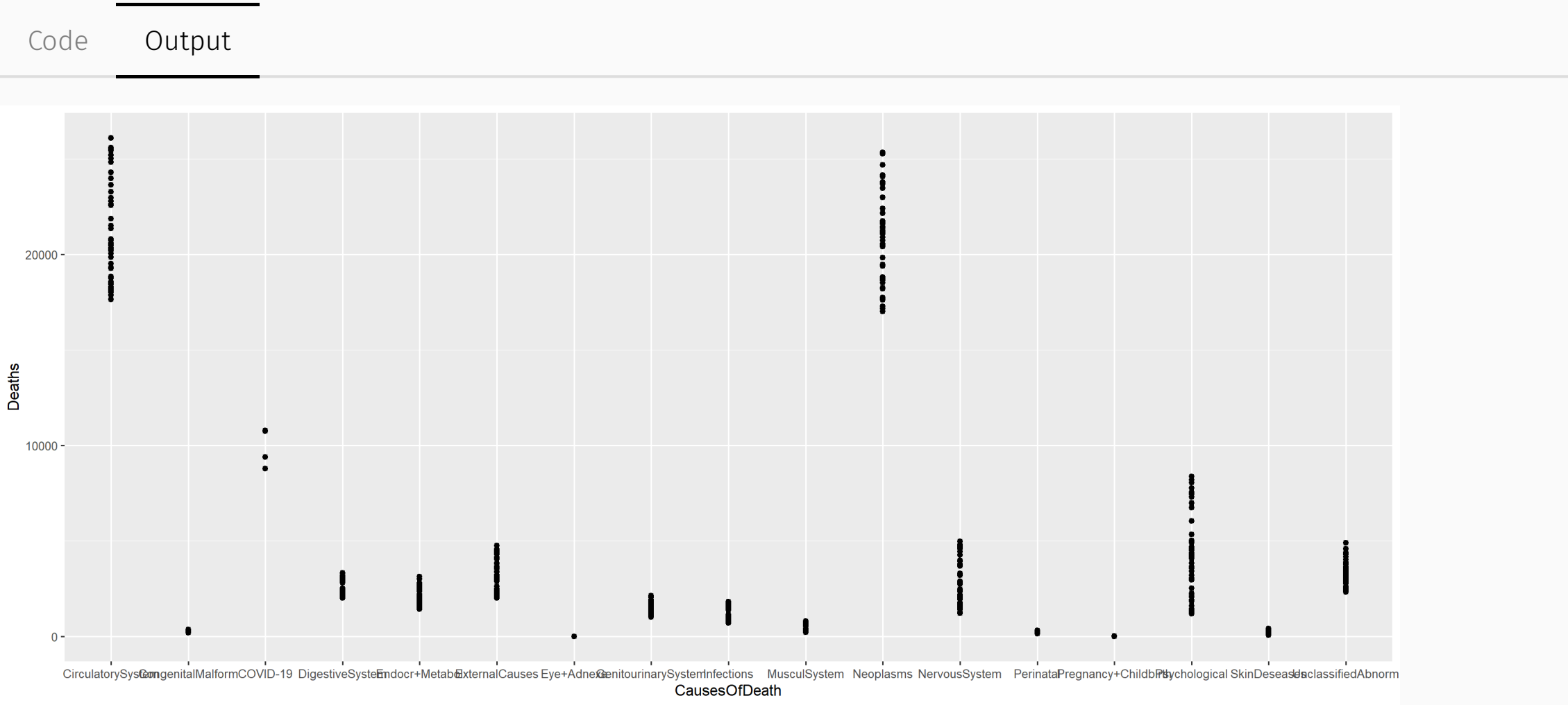
Graphing template:

```
ggplot(data = <DATA>)+  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Applied to the data:

```
Total ← filter(Death_in_NL, Age == "Total")  
ggplot(data = Total)+  
  geom_point(mapping = aes(x = CausesOfDeath, y = Deaths))
```

Build your first plot



Build your second plot

| Code | Output |
|------|--------|
|------|--------|

Graphing template:

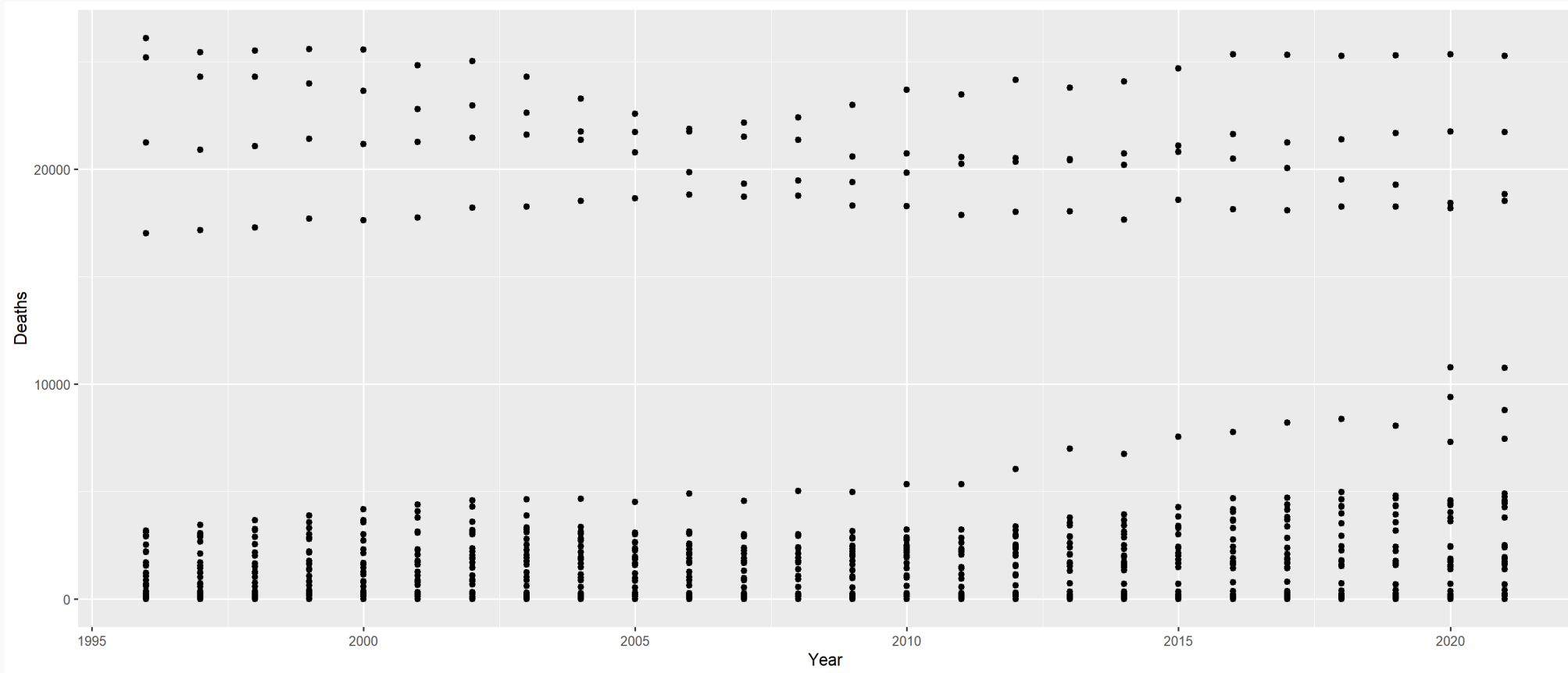
```
ggplot(data = <DATA>)+  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Applied to the data:

```
Total ← filter(Death_in_NL, Age == "Total")  
ggplot(data = Total)+  
  geom_point(mapping = aes(x = Year, y = Deaths))
```

Build your second plot

Code Output

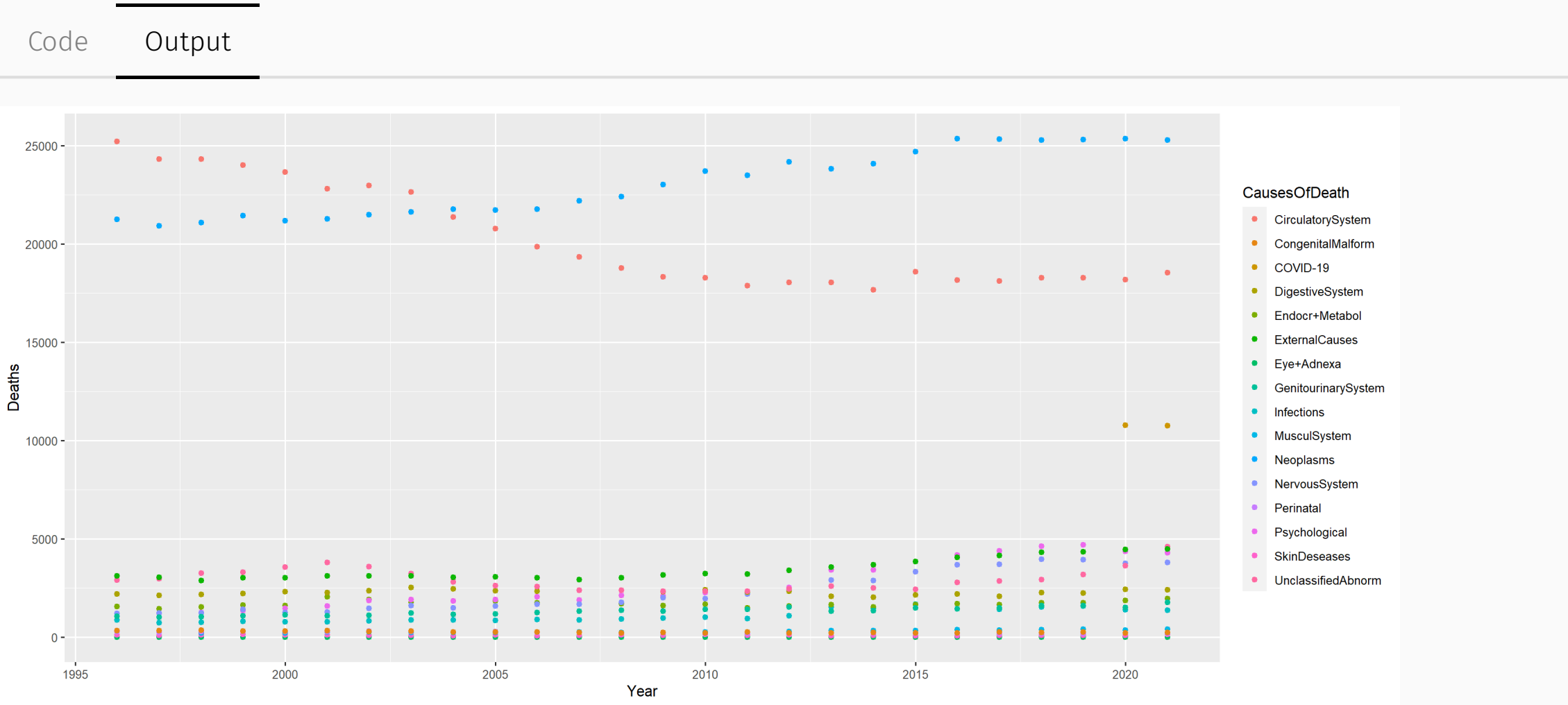


Build your second plot: edited version

| Code | Output |
|------|--------|
|------|--------|

```
Total_men <- filter(Death_in_NL, Age == "Total", Sex == "Male")
ggplot(data = Total_men)+
  geom_point(mapping = aes(x = Year, y = Deaths, color = CausesOfDeath))
```

Build your second plot: edited version



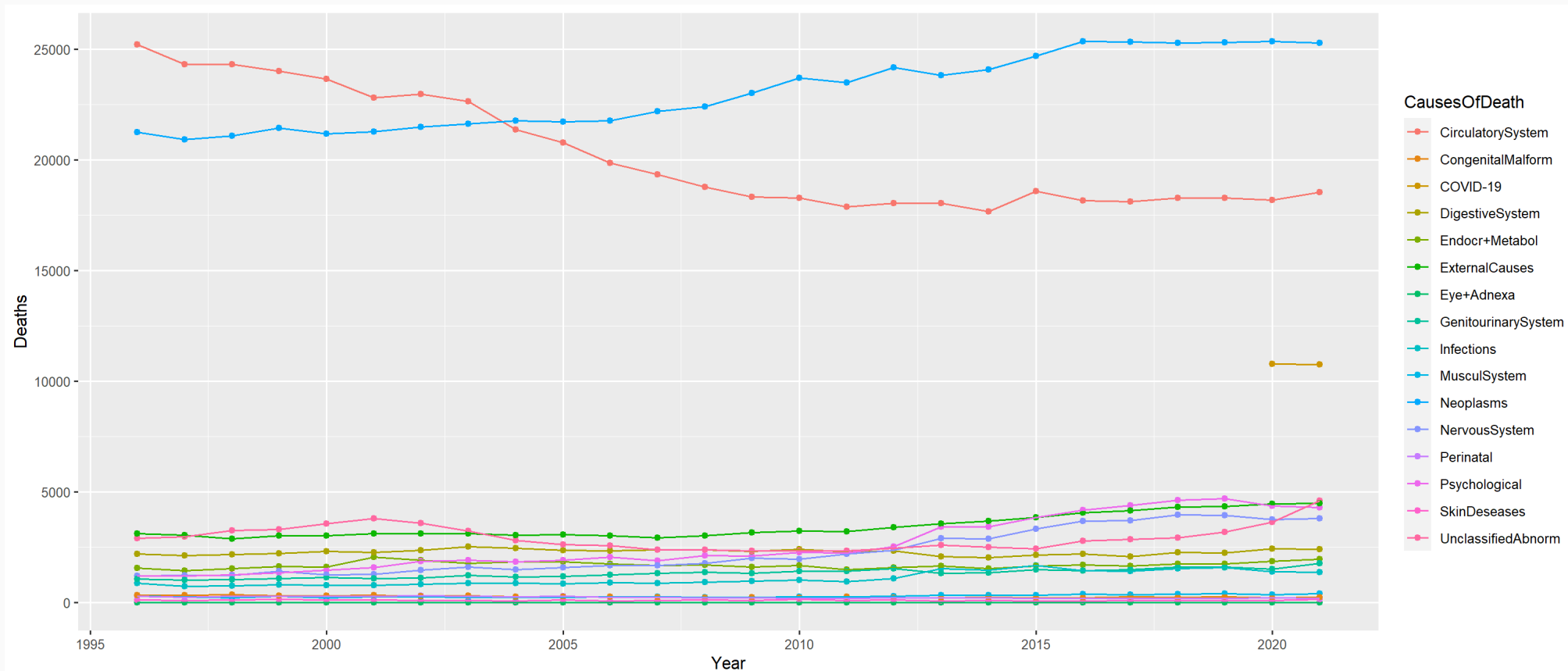
Build your second plot: edited version 2

| Code | Output |
|------|--------|
|------|--------|

```
Total_men <- filter(Death_in_NL, Age == "Total", Sex == "Male")
ggplot(data = Total_men)+
  geom_point(mapping = aes(x = Year, y = Deaths, color = CausesOfDeath))+
  geom_path(mapping = aes(x = Year, y = Deaths, color = CausesOfDeath))
```

Build your second plot: edited version 2

Code Output



Polishing the code

| Code | Output |
|------|--------|
|------|--------|

These three chunks of code generate the same plot:

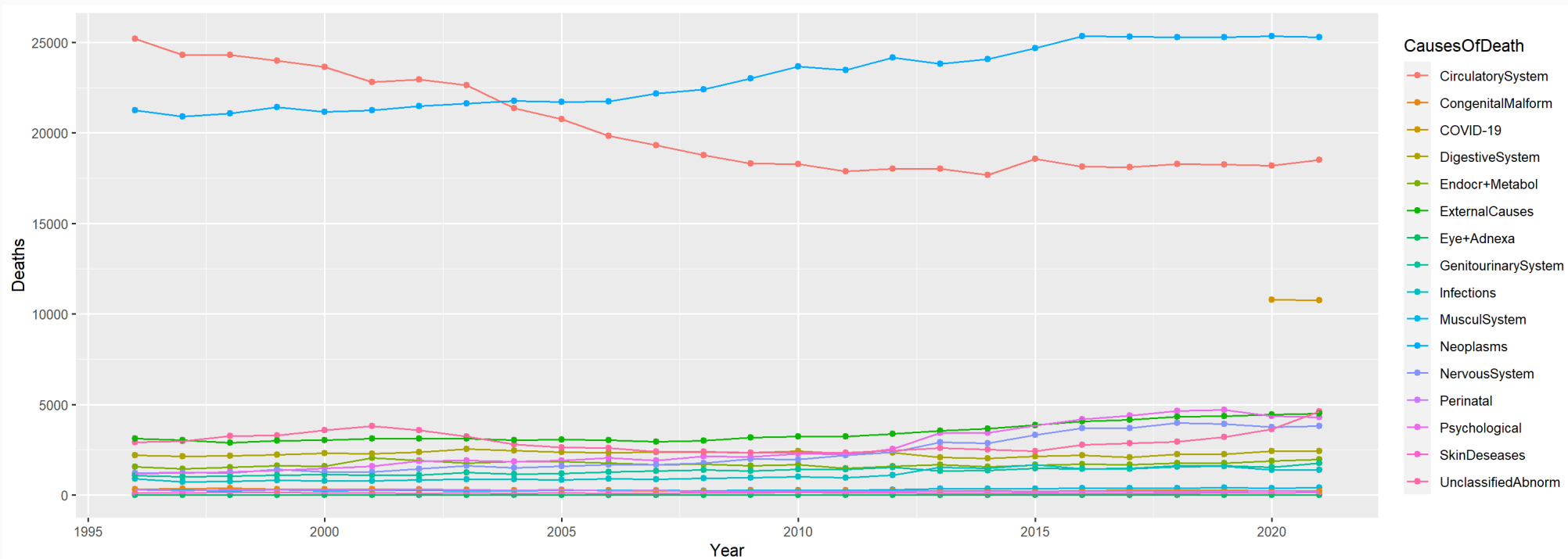
```
Total_men <- filter(Death_in_NL, Age == "Total", Sex == "Male")
ggplot(data = Total_men)+
  geom_point(mapping = aes(x = Year, y = Deaths, color = CausesOfDeath))+
  geom_path(mapping = aes(x = Year, y = Deaths, color = CausesOfDeath))
```

```
Total_men <- filter(Death_in_NL, Age == "Total", Sex == "Male")
ggplot(Total_men, aes(x = Year, y = Deaths, color = CausesOfDeath))+
  geom_point()+
  geom_path()
```

```
Death_in_NL %>%
  filter(Age == "Total", Sex == "Male") %>%
  ggplot(aes(x = Year, y = Deaths, color = CausesOfDeath))+
  geom_point()+
  geom_path()
```

Polishing the code

Code Output



Graphing template: updated

Minimal template:

```
ggplot(data = <DATA>)+  
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))</br>
```

Updated template:

```
ggplot(<DATA>)+  
<GEOM_FUNCTION1>(aes(<MAPPINGS1>))+  
<GEOM_FUNCTION2>(aes(<MAPPINGS2>))+  
<GEOM_FUNCTION3>(aes(<MAPPINGS3>))
```

or if geoms use the same mappings:

```
ggplot(<DATA>, aes(<MAPPINGS>))+  
<GEOM_FUNCTION1>()+  
<GEOM_FUNCTION2>()+  
<GEOM_FUNCTION3>()
```

or you can pipe (%>%) the data into ggplot:

```
DATA %>%  
ggplot()+  
<GEOM_FUNCTION1>(aes(<MAPPINGS1>))+  
<GEOM_FUNCTION2>(aes(<MAPPINGS2>))+  
<GEOM_FUNCTION3>(aes(<MAPPINGS3>))
```

Build different vizualizations

Visualizing amounts

Vizualizing amounts: bar plot

In many scenarios, we are interested in the magnitude of some set of numbers. The standard visualization in this scenario is the bar plot, which comes in several variations, including simple bars as well as grouped and stacked bars.

| Code | Output |
|------|--------|
|------|--------|

```
Neoplasms <- filter(Death_in_NL, CausesOfDeath = "Neoplasms",  
                    Year = "2020", Sex = "Male", Age != "Total")  
ggplot(data = Neoplasms)+  
  geom_col(mapping = aes(x = Age, y = Deaths))
```

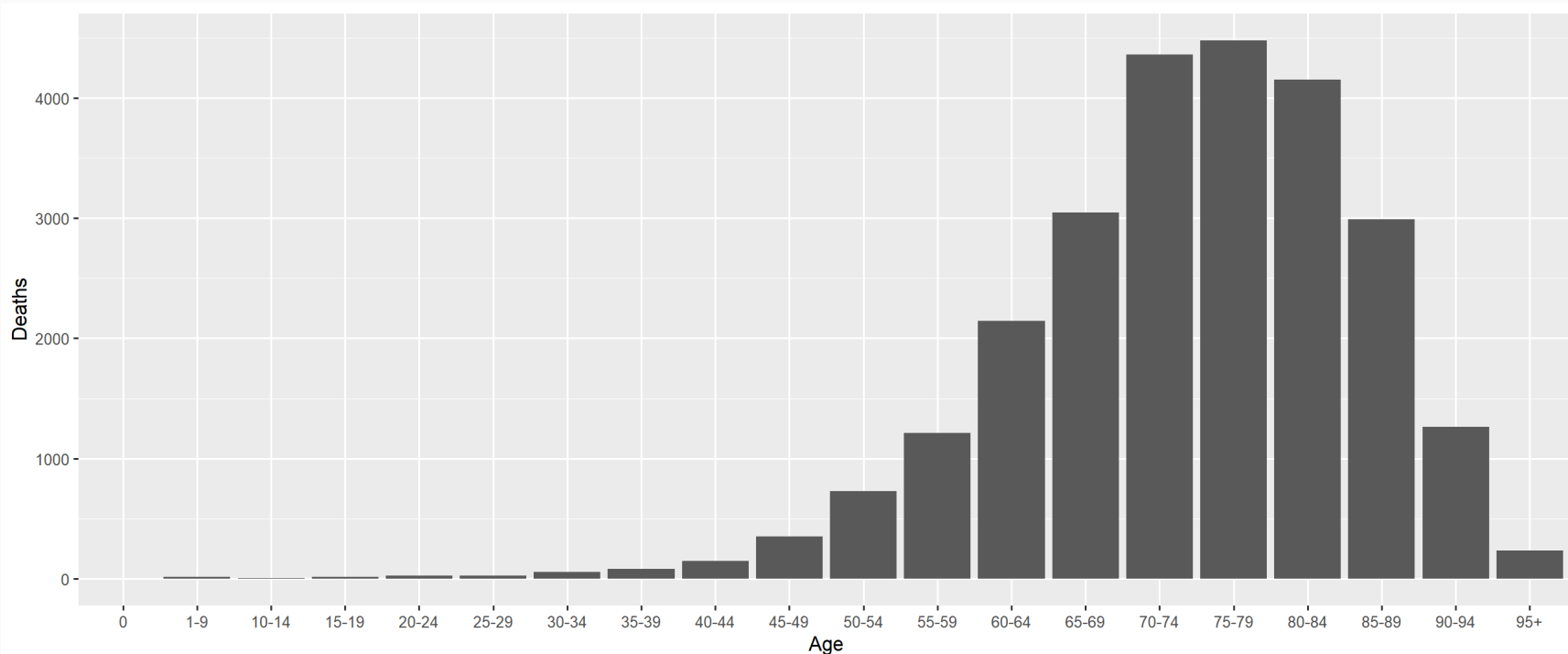
or we can write the same concise:

```
Death_in_NL %>%  
  filter(CausesOfDeath = "Neoplasms",  
         Year = "2020", Sex = "Male", Age != "Total")%>%  
  ggplot(aes(x = Age, y = Deaths))+  
    geom_col()
```

Vizualizing amounts: bar plot

In many scenarios, we are interested in the magnitude of some set of numbers. The standard visualization in this scenario is the bar plot, which comes in several variations, including simple bars as well as grouped and stacked bars.

| Code | Output |
|------|--------|
|------|--------|



Vizualizing amounts: grouped bar plot

In a grouped bar plot, we draw a group of bars at each position along the x axis, determined by one categorical variable, and then we draw bars within each group according to the other categorical variable.

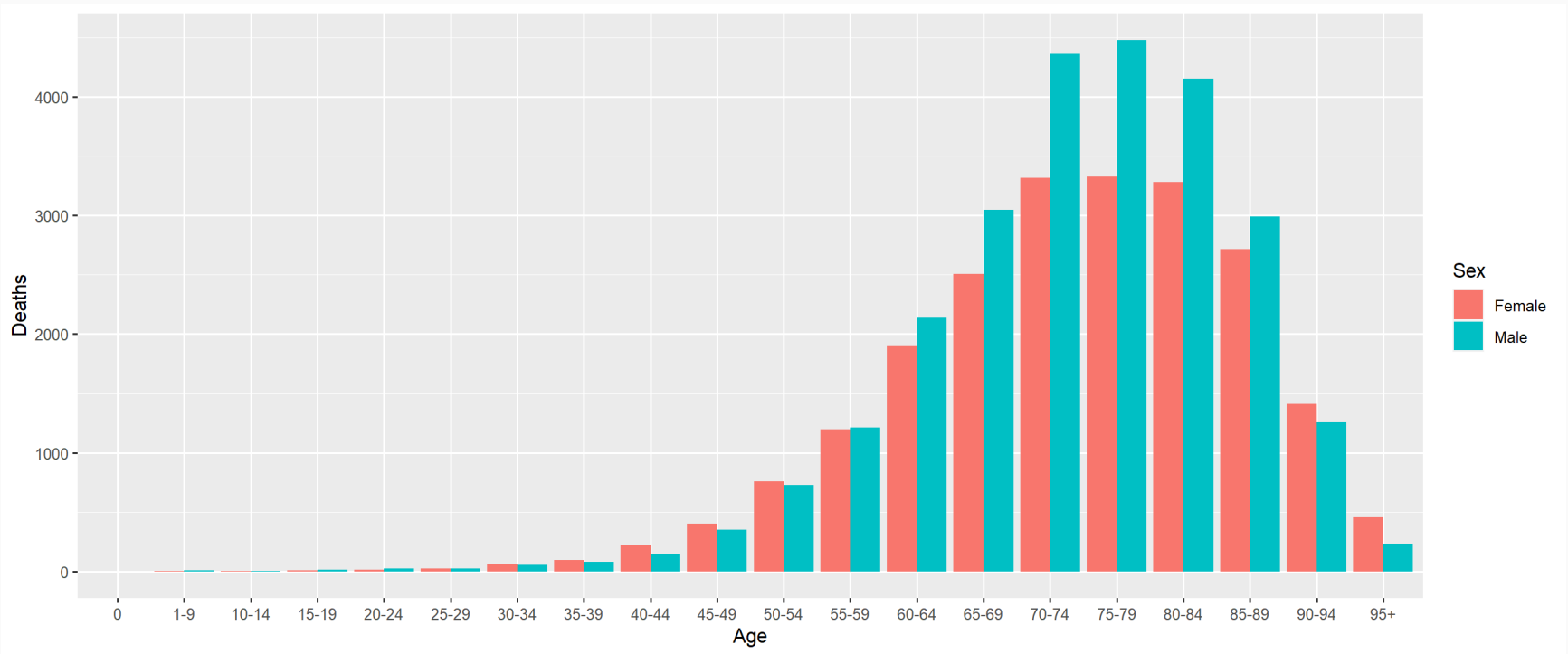
| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL %>%  
  filter(CausesOfDeath = "Neoplasms",  
         Year = "2020", Age ≠ "Total")%>%  
  ggplot(aes(x = Age, y = Deaths, fill = Sex))+  
    geom_col(position = "dodge")
```

Vizualizing amounts: grouped bar plot

In a grouped bar plot, we draw a group of bars at each position along the x axis, determined by one categorical variable, and then we draw bars within each group according to the other categorical variable.

| Code | Output |
|------|--|
| |  |



Vizualizing amounts: stacked bar plot

Instead of drawing groups of bars side-by-side, it is sometimes preferable to stack bars on top of each other. Stacking is useful when the sum of the amounts represented by the individual stacked bars is in itself a meaningful amount.

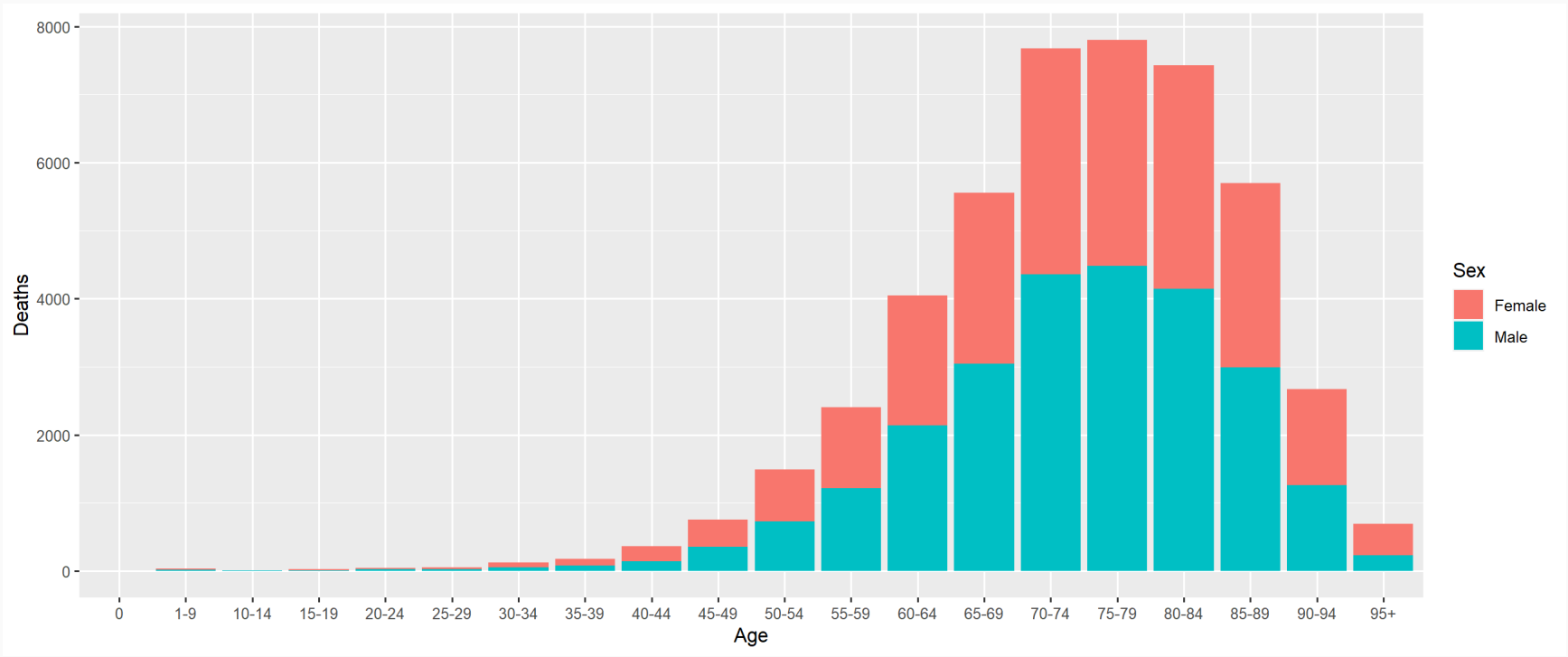
| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL %>%  
  filter(CausesOfDeath = "Neoplasms",  
         Year = "2020", Age != "Total")%>%  
  ggplot(aes(x = Age, y = Deaths, fill = Sex))+  
    geom_col(position = "stack")
```

Vizualizing amounts: stacked bar plot

Instead of drawing groups of bars side-by-side, it is sometimes preferable to stack bars on top of each other. Stacking is useful when the sum of the amounts represented by the individual stacked bars is in itself a meaningful amount.

| Code | Output |
|------|--|
| |  |



Vizualizing amounts: heatmap

As an alternative to mapping data values onto positions via bars or dots, we can map data values onto colors. Such a figure is called a heatmap.

| Code | Output |
|------|--------|
|------|--------|

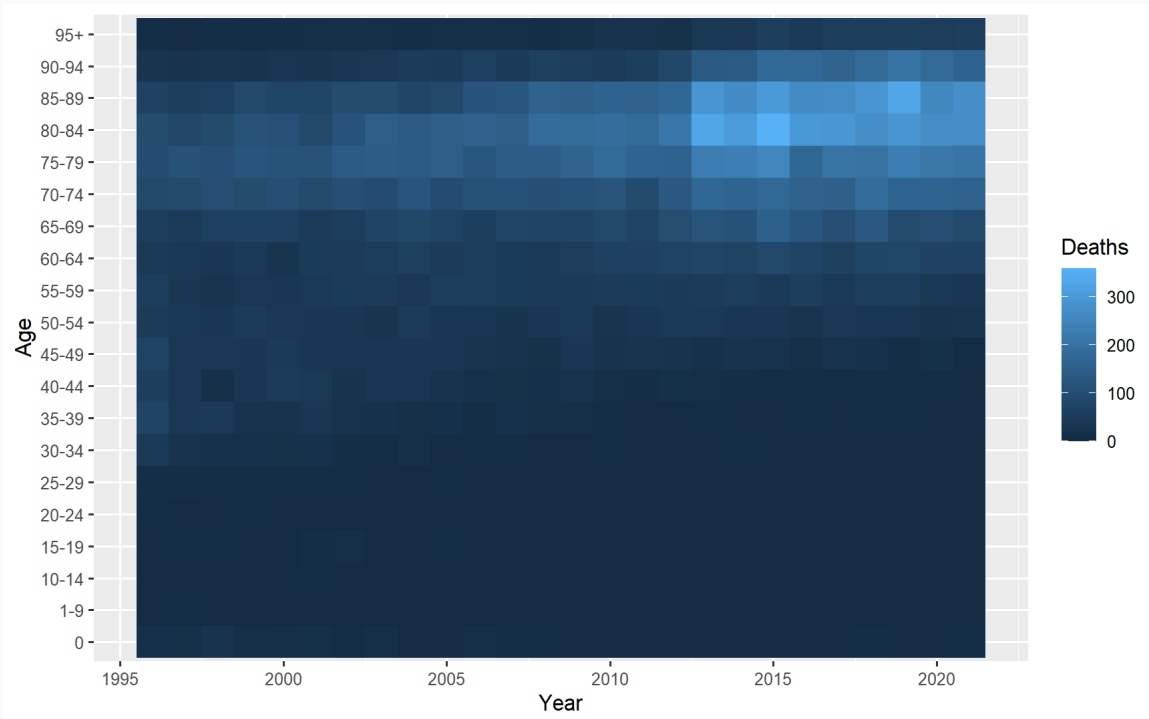
| | |
|---|--|
| <pre>Death_in_NL %>% filter(Sex = "Male", CausesOfDeath = "Infections", Age ≠ "Total") %>% ggplot(mapping = aes(x = Year, y = Age, fill = Deaths))+ geom_tile()</pre> | |
|---|--|

Vizualizing amounts: heatmap

As an alternative to mapping data values onto positions via bars or dots, we can map data values onto colors. Such a figure is called a heatmap.

Code

Output



Build different vizualizations

Visualizing x-y relationships

Scatterplot

A Scatterplot displays the relationship between two numeric variables.

| Code | Output |
|------|--------|
|------|--------|

```
Neoplasms <- filter(Death_in_NL, CausesOfDeath == "Neoplasms", Age == "Total")
p <- ggplot(data = Neoplasms)+
  geom_point(mapping = aes(x = Year, y = Deaths, color = Sex))
```

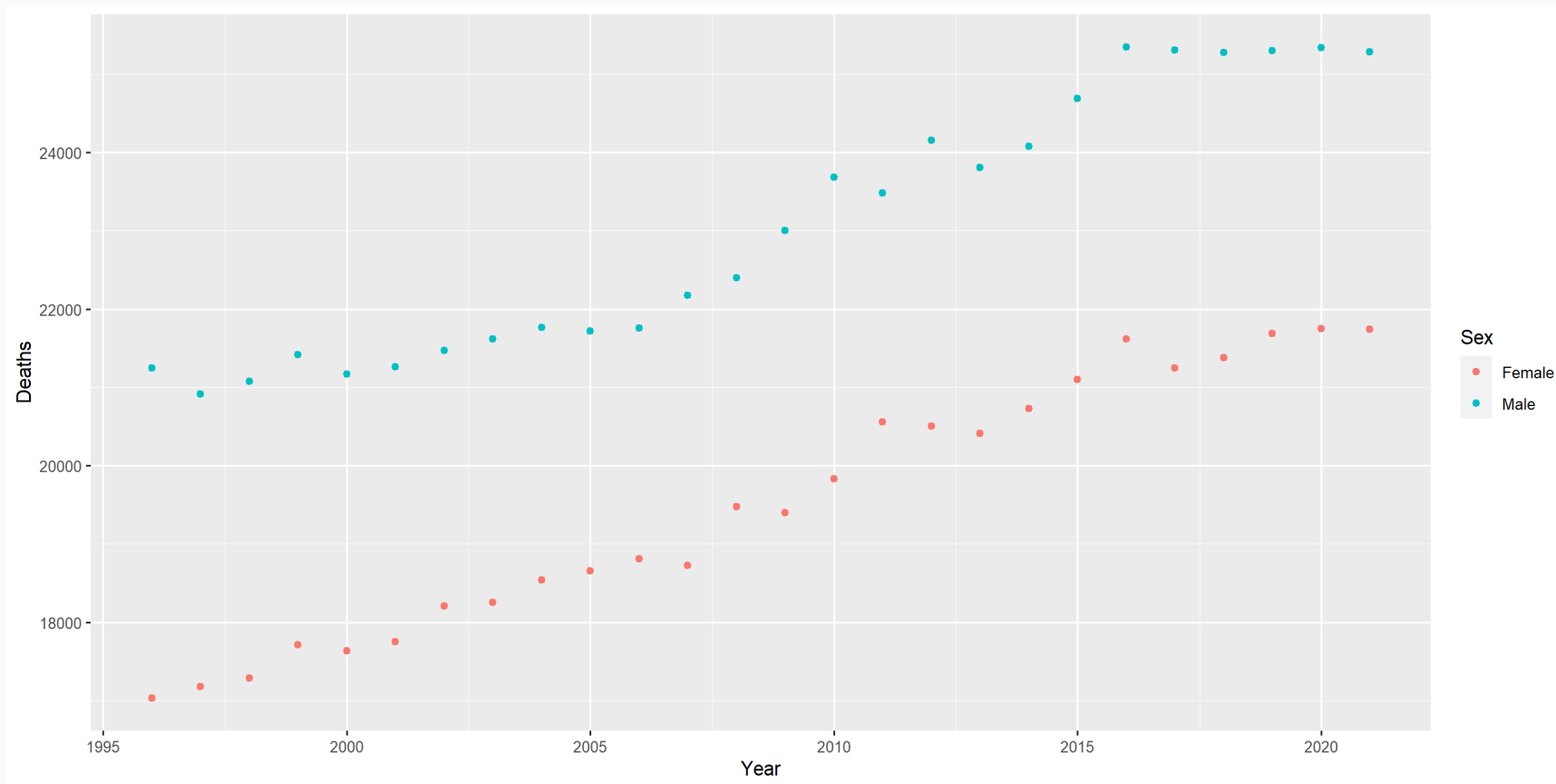
or we can write the same more concise:

```
Death_in_NL %>%
  filter(CausesOfDeath == "Neoplasms", Age == "Total")%>%
  ggplot(aes(x = Year, y = Deaths, color = Sex))+
    geom_point()
```

Scatterplot

A Scatterplot displays the relationship between two numeric variables.

Code Output



Line chart

A line chart or line graph displays the evolution of one or several numeric variables. The input data frame requires at least two columns: an ordered numeric variable for the X axis; (2) another numeric variable for the Y axis.

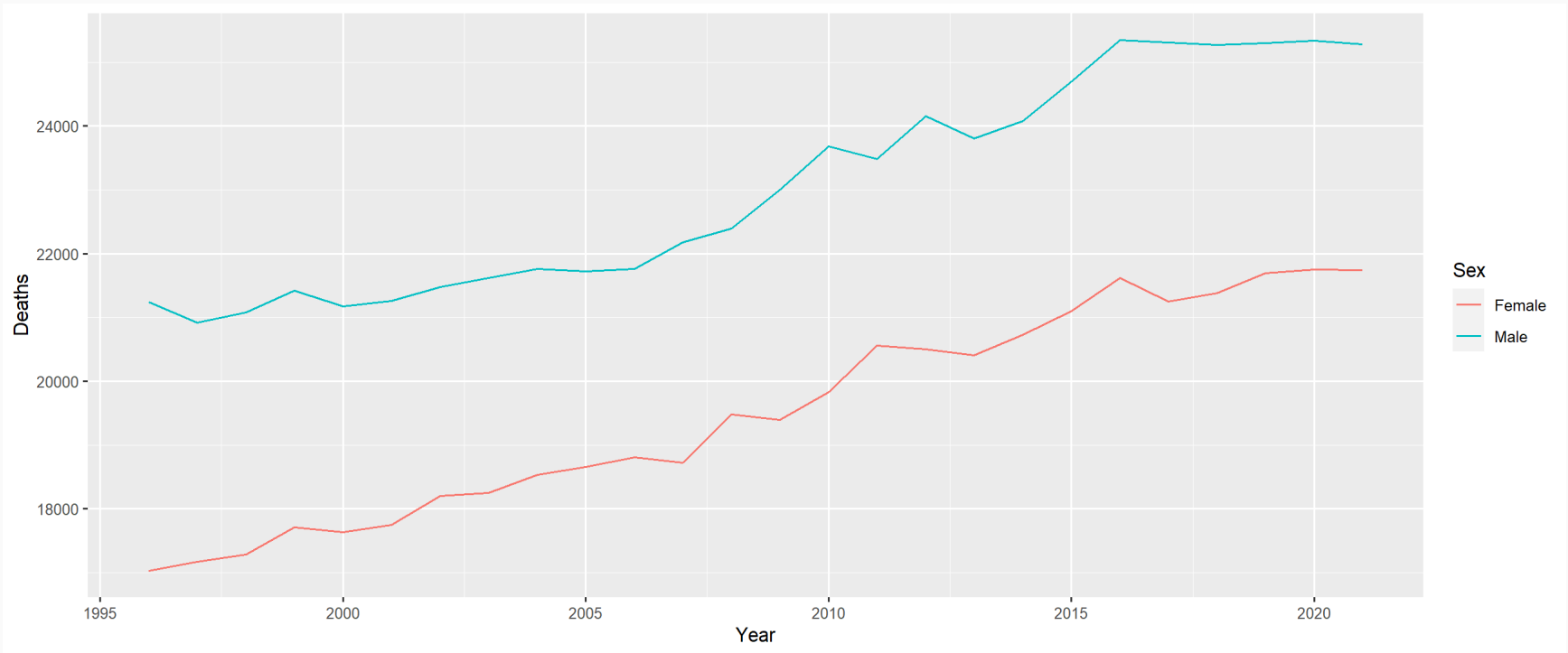
| Code | Output |
|------|--------|
|------|--------|

| | |
|--|--|
| <pre>Death_in_NL %>% filter(CausesOfDeath = "Neoplasms", Age = "Total")%>% ggplot(aes(x = Year, y = Deaths, color = Sex))+ geom_line()</pre> | |
|--|--|

Line chart

A line chart or line graph displays the evolution of one or several numeric variables. The input data frame requires at least two columns: an ordered numeric variable for the X axis; (2) another numeric variable for the Y axis.

| Code | Output |
|------|--|
| |  |



Line chart: updated

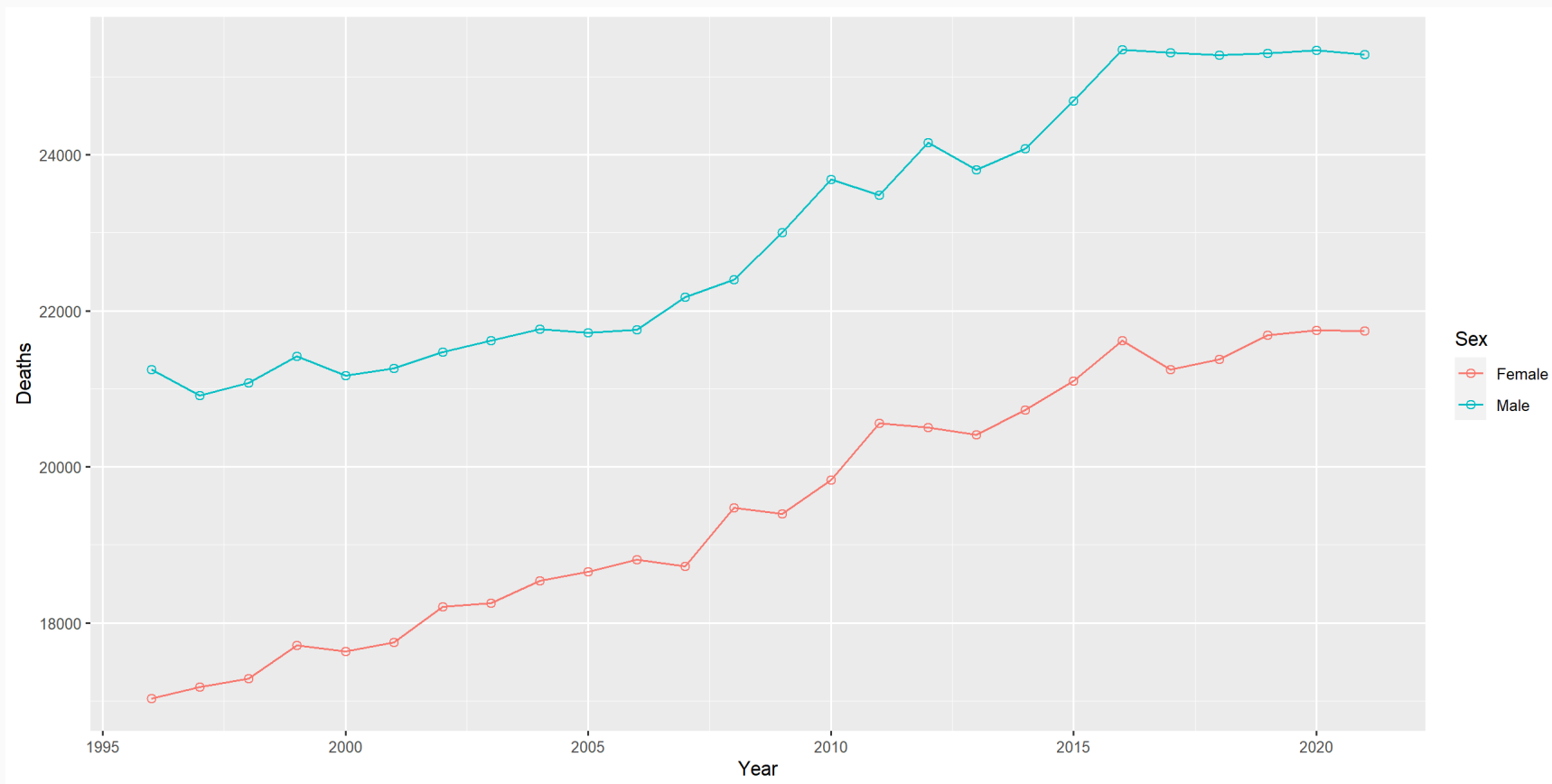
| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL %>%  
  filter(CausesOfDeath = "Neoplasms", Age = "Total")%>%  
  ggplot(aes(x = Year, y = Deaths, color = Sex))+  
    geom_line()+  
    geom_point(shape=21, size=2)
```

Line chart: updated

Code

Output



Smooth line graph

We can use smooth lines to represent trends in a larger dataset. The act of smoothing produces a function that captures key patterns in the data while removing irrelevant minor detail or noise.

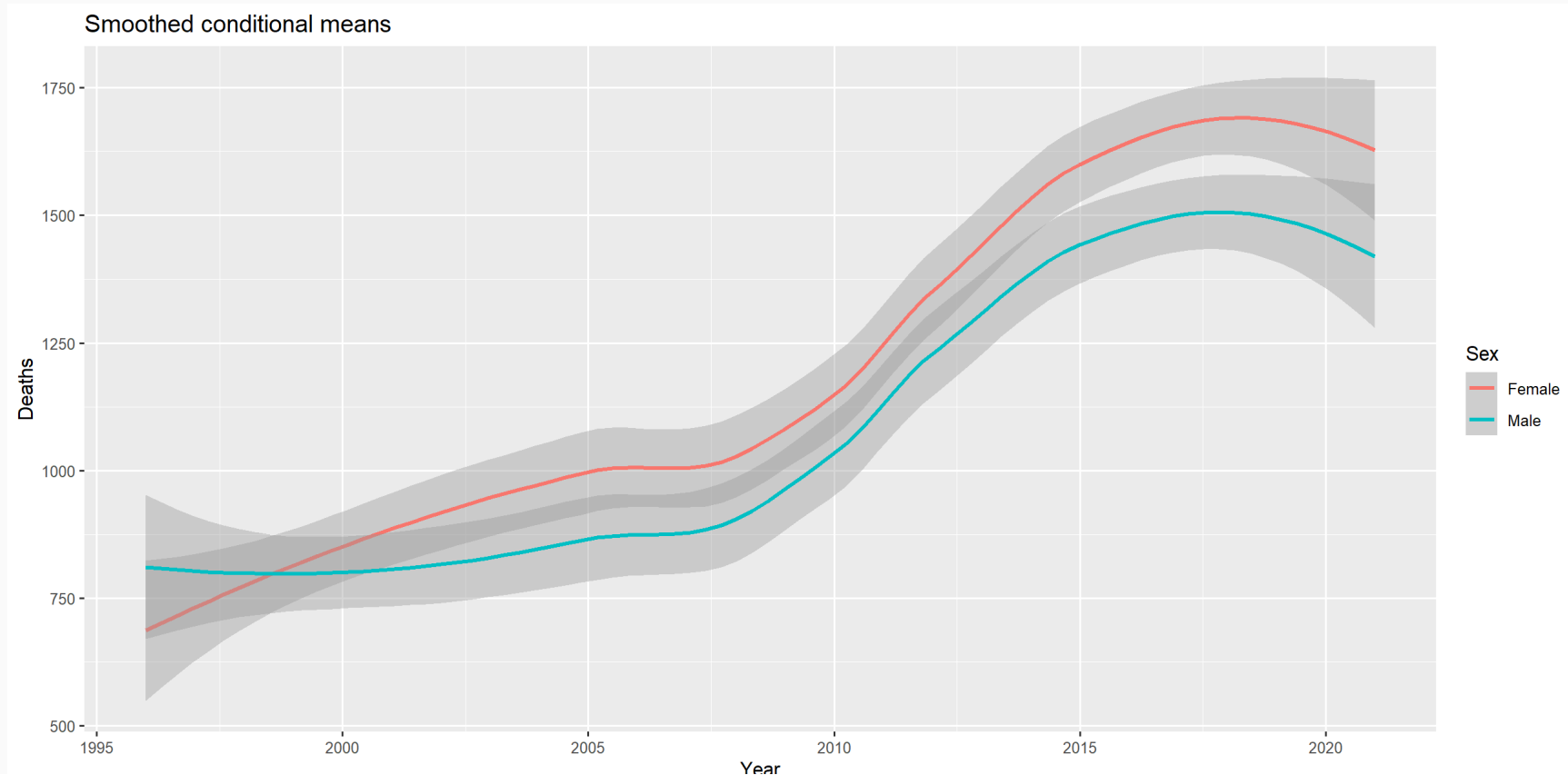
| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL %>%  
  filter(CausesOfDeath = "Infections", Age = "Total") %>%  
  ggplot(mapping = aes(x = Year, y = Deaths, color = Sex))+  
  geom_smooth()+  
  ggtitle("Smoothed conditional means")
```

Smooth line graph

We can use smooth lines to represent trends in a larger dataset. The act of smoothing produces a function that captures key patterns in the data while removing irrelevant minor detail or noise.

Code Output



Build different visualizations

Visualizing distributions

Histogram

A histogram takes as input a *numeric* variable only. The variable is cut into several *bins*, and the number of observation per *bin* is represented by the height of the bar.

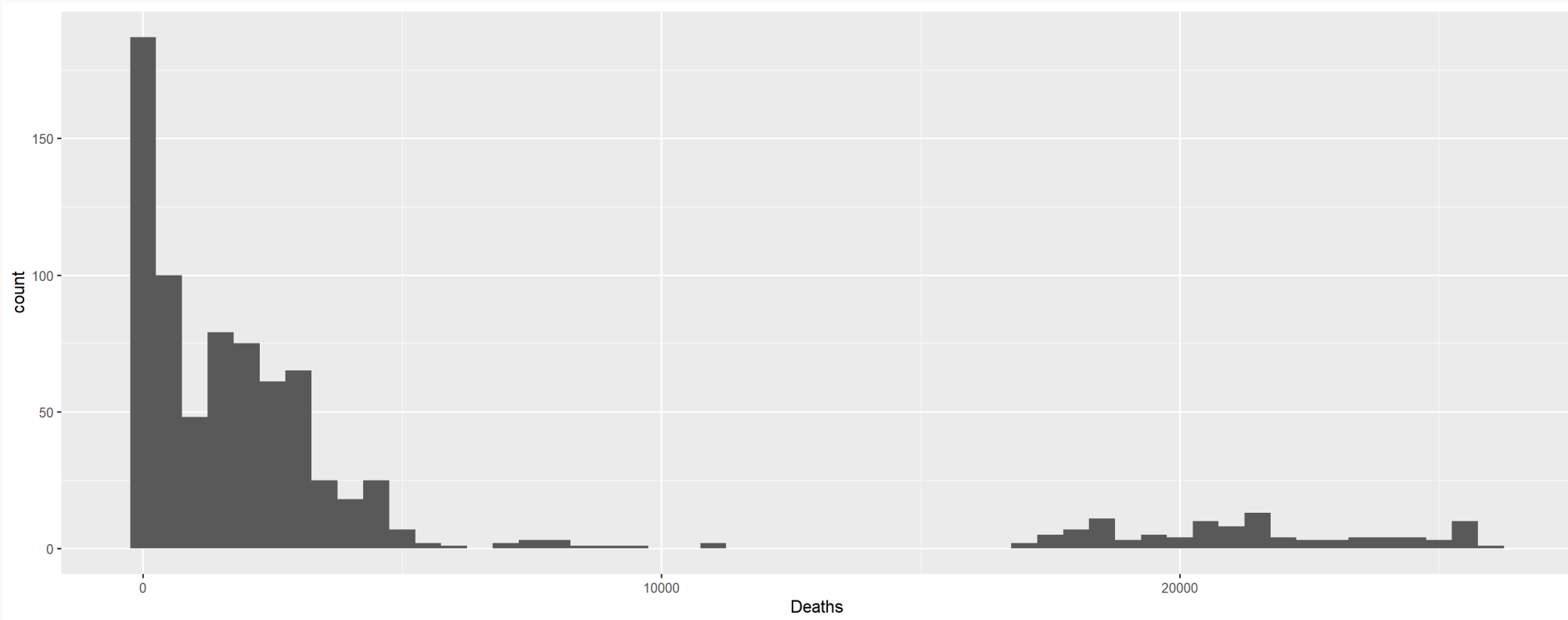
| Code | Output |
|------|--------|
|------|--------|

| | |
|--|--|
| <pre>Death_in_NL %>% filter(Age == "Total")%>% ggplot(aes(x = Deaths))+ geom_histogram(binwidth=500)</pre> | |
|--|--|

Histogram

A histogram takes as input a *numeric* variable only. The variable is cut into several *bins*, and the number of observation per *bin* is represented by the height of the bar.

| Code | Output |
|------|--------|
|------|--------|



Histogram: binwidth

Play with the *bin* size, it can give different insight.

| Code | Output |
|------|--------|
|------|--------|

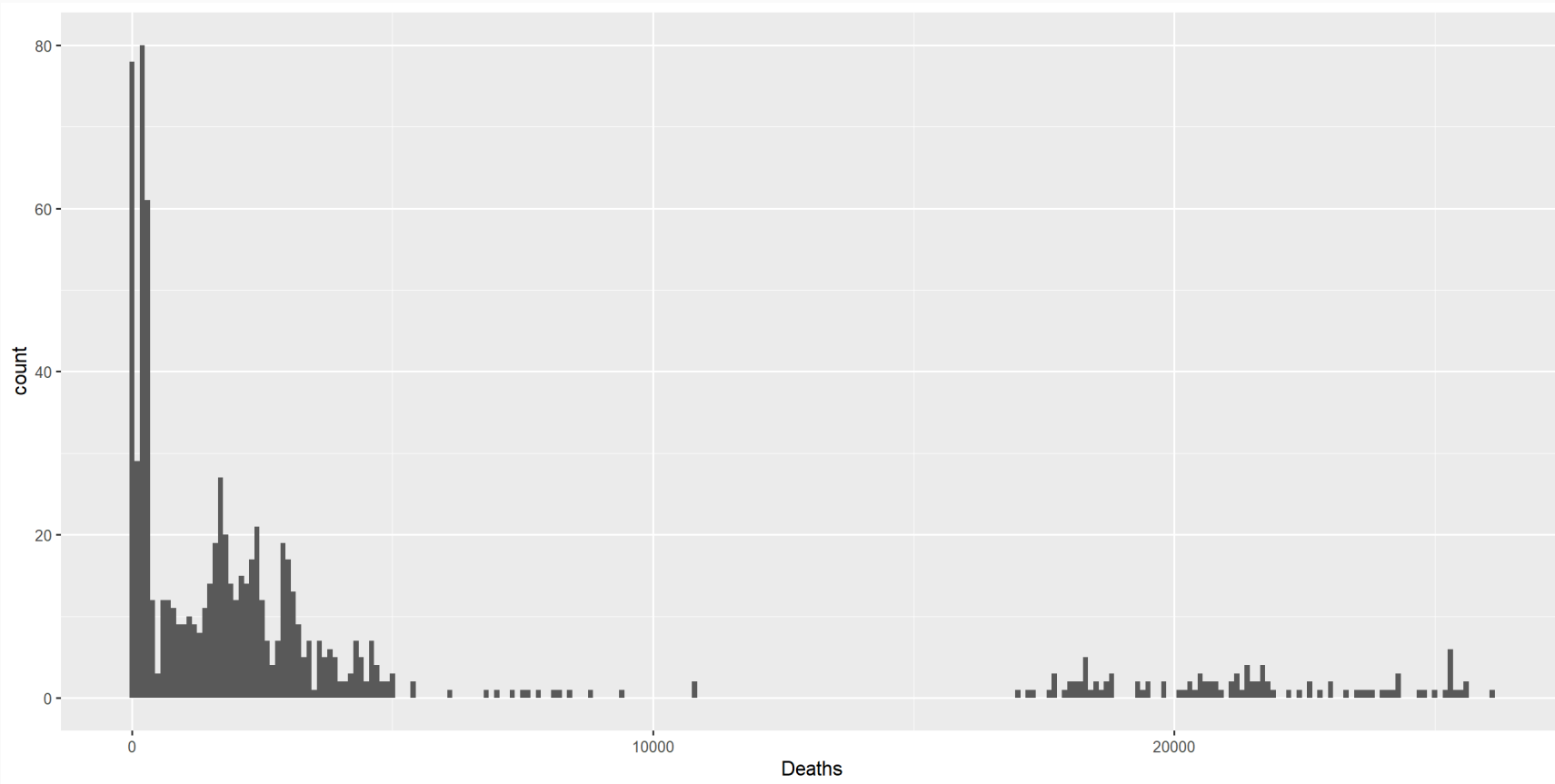
| | |
|--|--|
| <pre>Death_in_NL %>% filter(Age == "Total")%>% ggplot(aes(x = Deaths))+ geom_histogram(binwidth=100)</pre> | |
|--|--|

Histogram: binwidth

Play with the *bin* size, it can give different insight.

Code

Output



Histogram: two variables

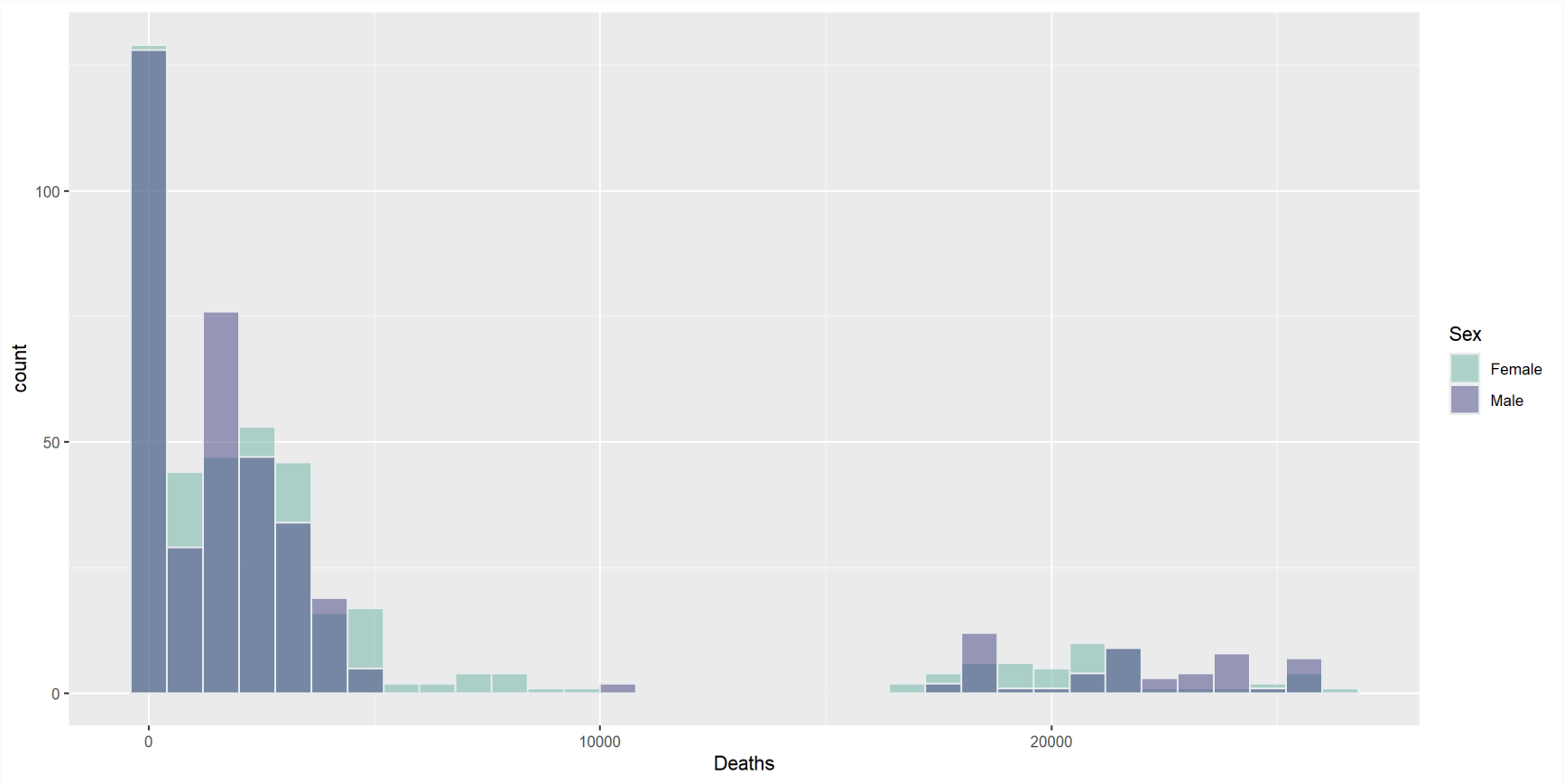
It is possible to represent the distribution of *several variables* on the same axis using this technique.

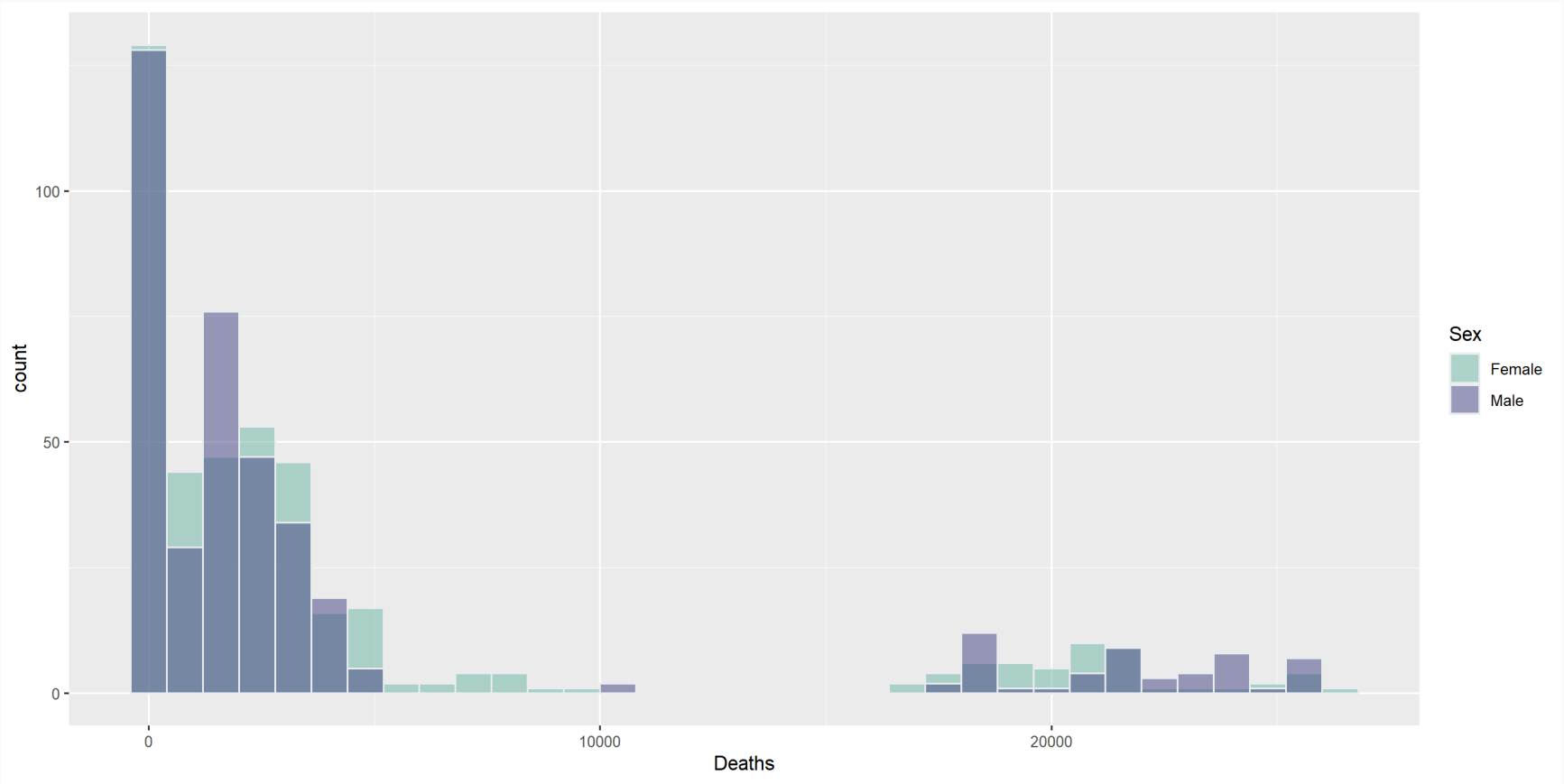
| Code | Output |
|------|--------|
|------|--------|

| | |
|---|--|
| <pre>Death_in_NL %>% filter(Age == "Total")%>% ggplot(aes(x = Deaths, fill = Sex))+ geom_histogram(color="#e9ecf", alpha=0.5, position="identity", binwidth = 800)+ scale_fill_manual(values=c("#69b3a2", "#404080"))</pre> | |
|---|--|

Histogram: two variables

It is possible to represent the distribution of *several variables* on the same axis using this technique.

| Code | Output |
|------|--|
| |  |



Density plot

In a density plot, we attempt to visualize the underlying probability distribution of the data by drawing an appropriate continuous curve estimated from the data. The most commonly used method for this estimation procedure is called kernel density estimation.

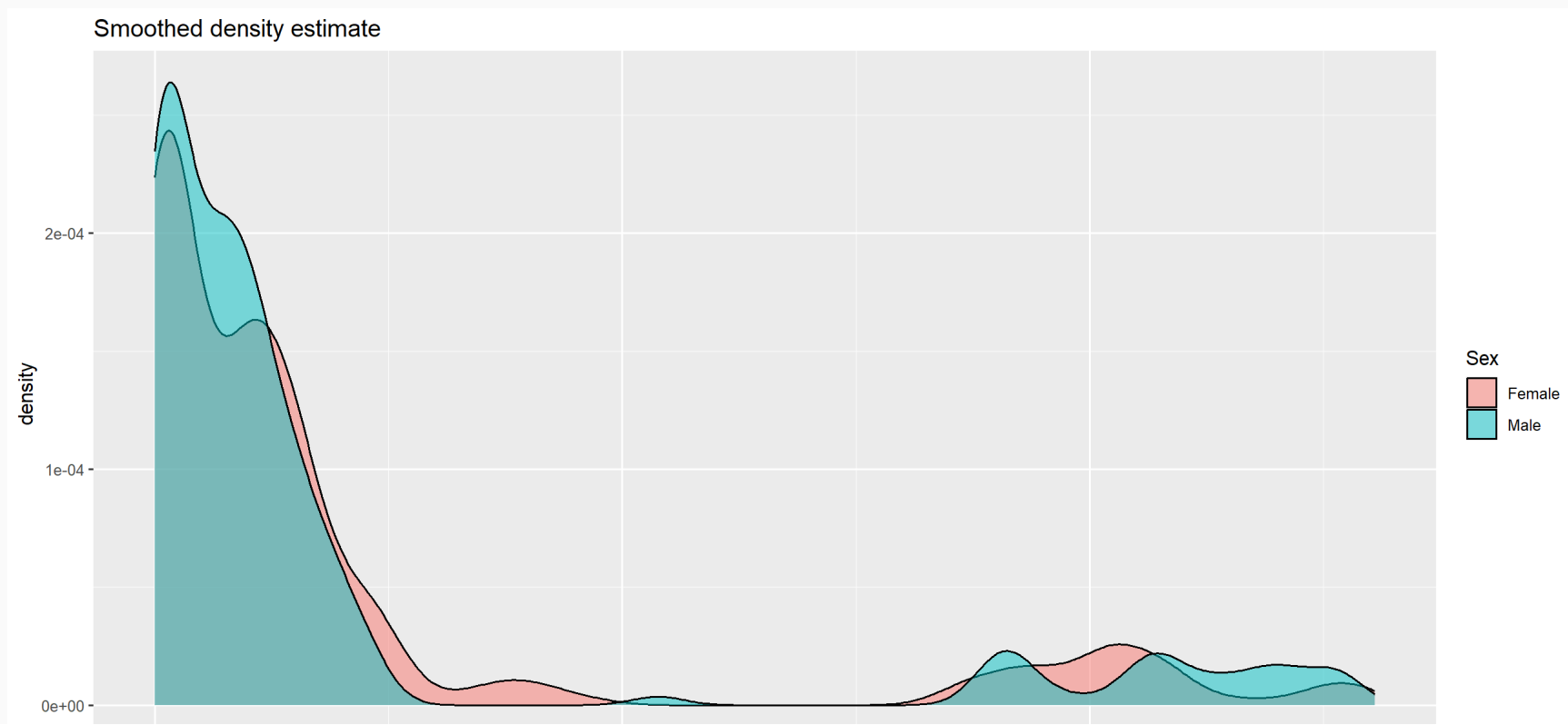
| Code | Output |
|------|--------|
|------|--------|

```
Death_in_NL %>%  
  filter(Age == "Total") %>%  
  ggplot(aes(x = Deaths, fill = Sex))+  
  geom_density(alpha = 0.5)+  
  ggtitle("Smoothed density estimate")
```

Density plot

In a density plot, we attempt to visualize the underlying probability distribution of the data by drawing an appropriate continuous curve estimated from the data. The most commonly used method for this estimation procedure is called kernel density estimation.

Code Output



Basic Boxplot

A boxplot divides the data into quartiles and visualizes them in a standardized manner. You need to specify a *quantitative variable* for the Y axis, and a *qualitative variable* for the X axis (a group).

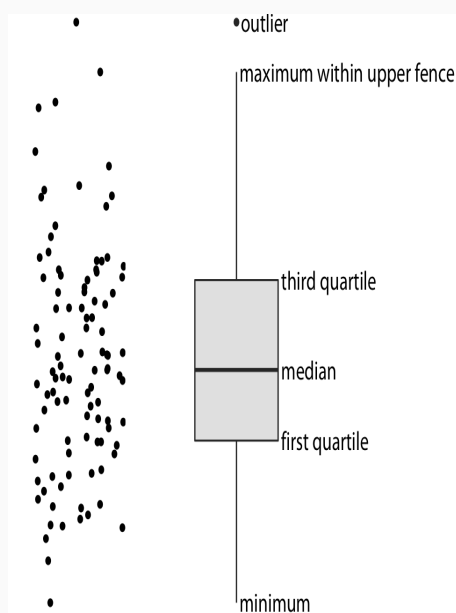
| Code | Output |
|------|--------|
|------|--------|

| | |
|--|--|
| <pre>Death_in_NL %>% filter(Age ≠ "Total", CausesOfDeath = "Neoplasms")%>% ggplot(aes(x = Age, y = Deaths))+ geom_boxplot(alpha=0.5)</pre> | |
|--|--|

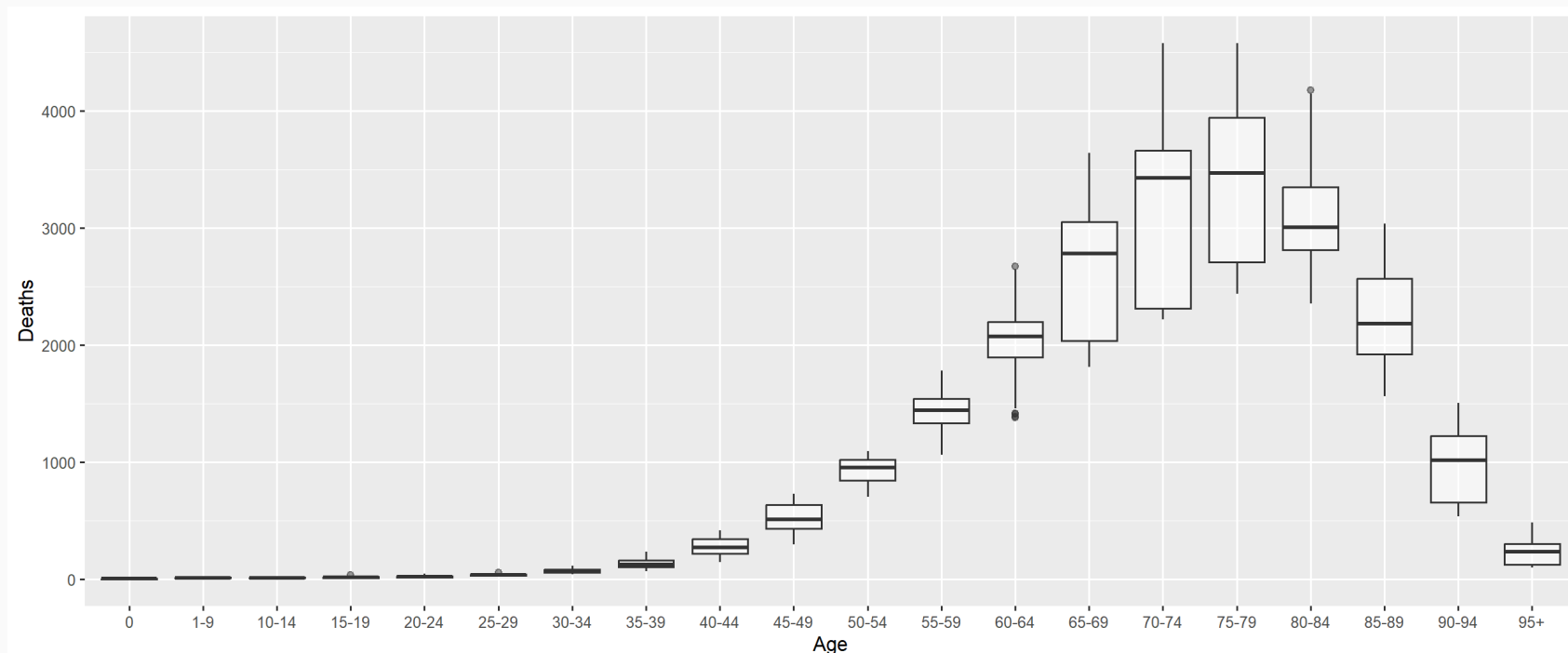
Basic Boxplot

A boxplot divides the data into quartiles and visualizes them in a standardized manner. You need to specify a *quantitative variable* for the Y axis, and a *qualitative variable* for the X axis (a group).

| Code | Output |
|------|--------|
|------|--------|



Anatomy of a boxplot



A grouped boxplot

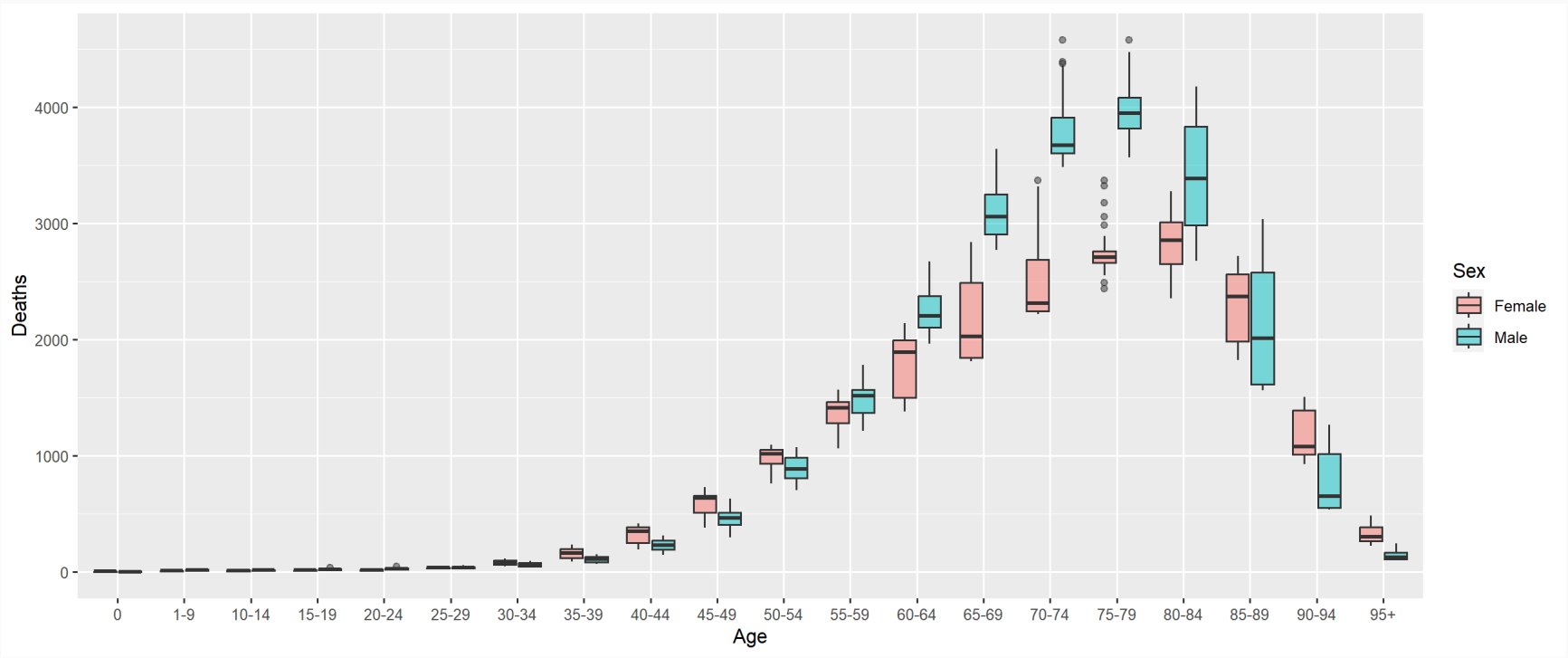
Here we visualize the distribution of 21 groups (Age intervals) and 2 subgroups (Male and Female). The group must be called in the X argument of ggplot2. The subgroup is called in the fill or color argument.

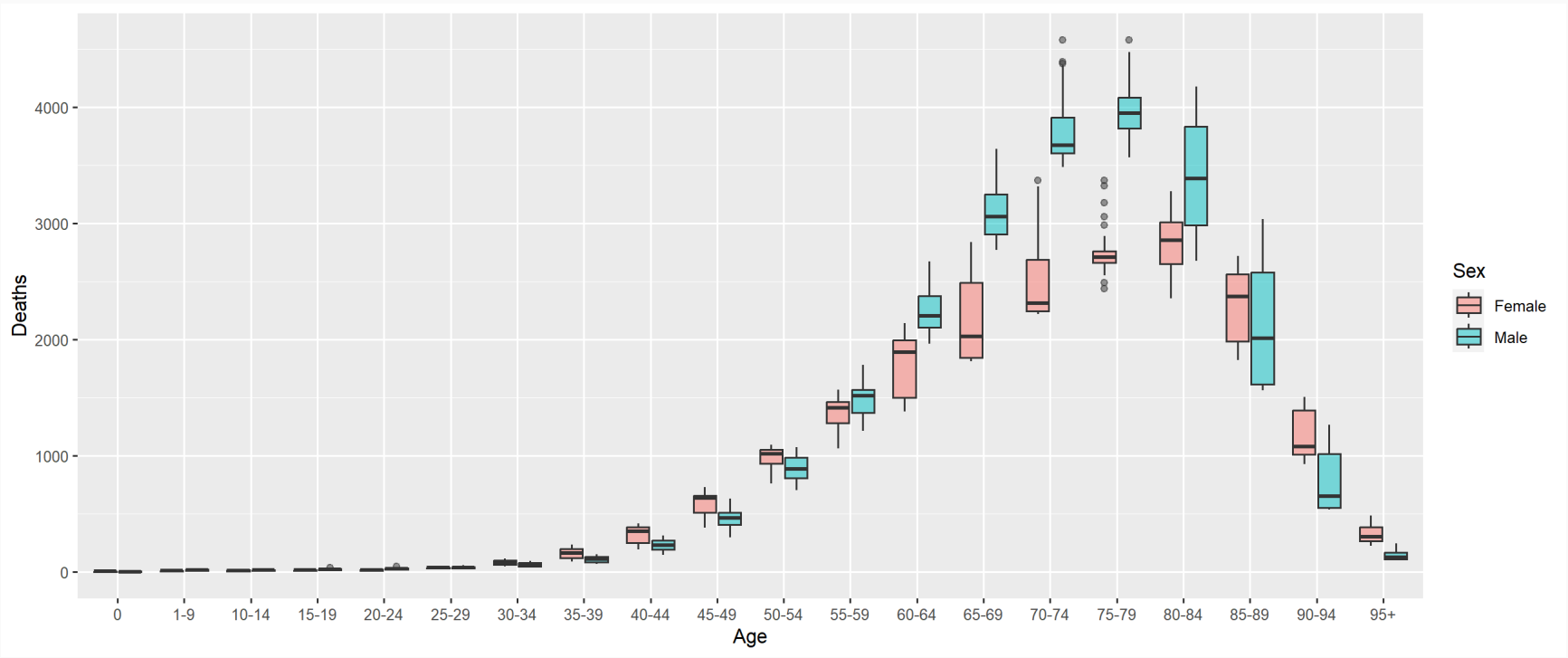
| Code | Output |
|------|--------|
|------|--------|

| | |
|--|--|
| <pre>Death_in_NL %>% filter(Age ≠ "Total", CausesOfDeath = "Neoplasms")%>% ggplot(aes(x = Age, y = Deaths, fill = Sex))+ geom_boxplot(alpha=0.5)</pre> | |
|--|--|

A grouped boxplot

Here we visualize the distribution of 21 groups (Age intervals) and 2 subgroups (Male and Female). The group must be called in the X argument of ggplot2. The subgroup is called in the fill or color argument.

| Code | Output |
|------|--|
| |  |



Boxplot with individual data points

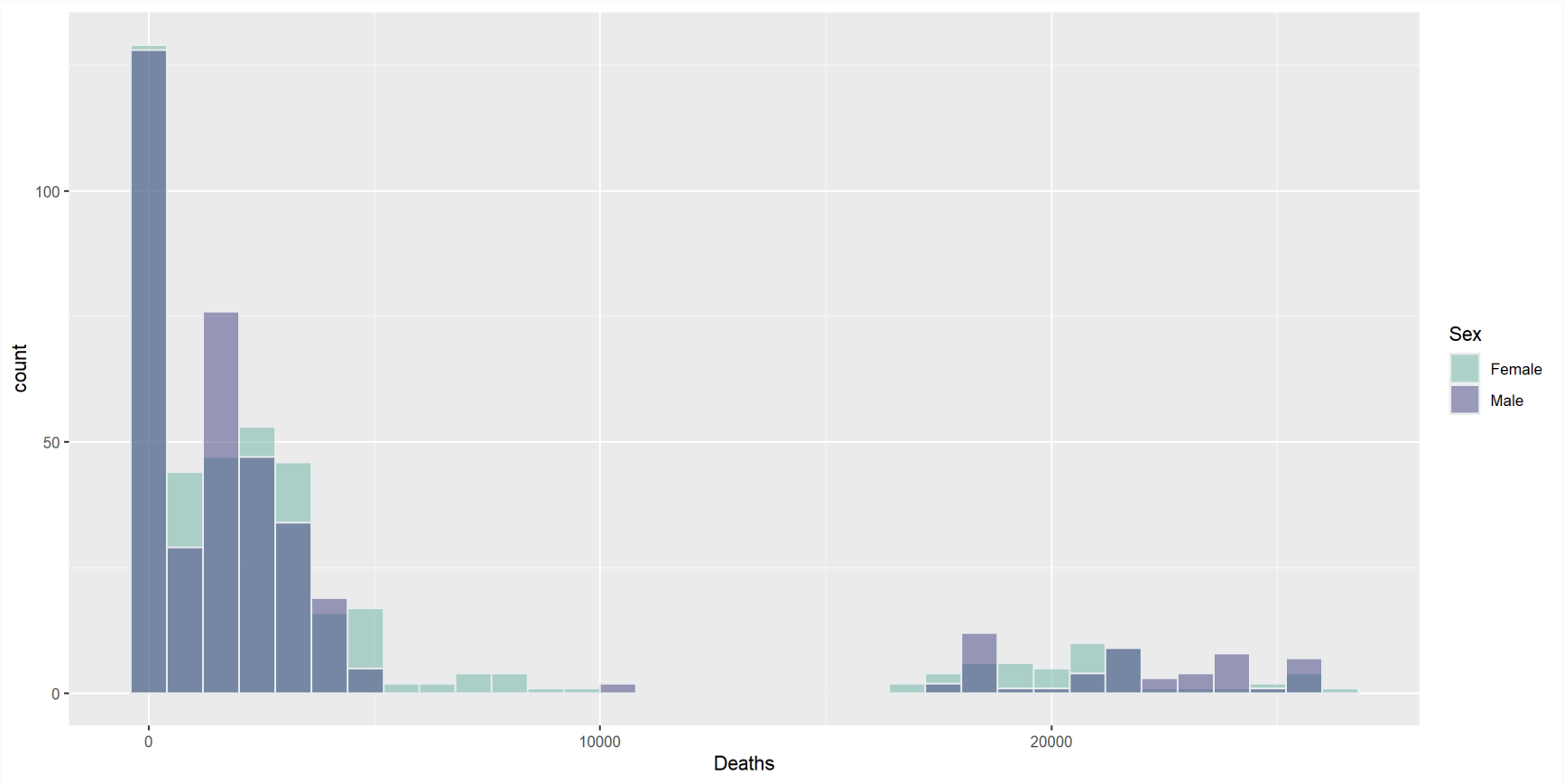
A boxplot summarizes the distribution of a continuous variable. It is often criticized for hiding the underlying data. Therefore, showing individual observations using jitter on top of boxes is a good practice.

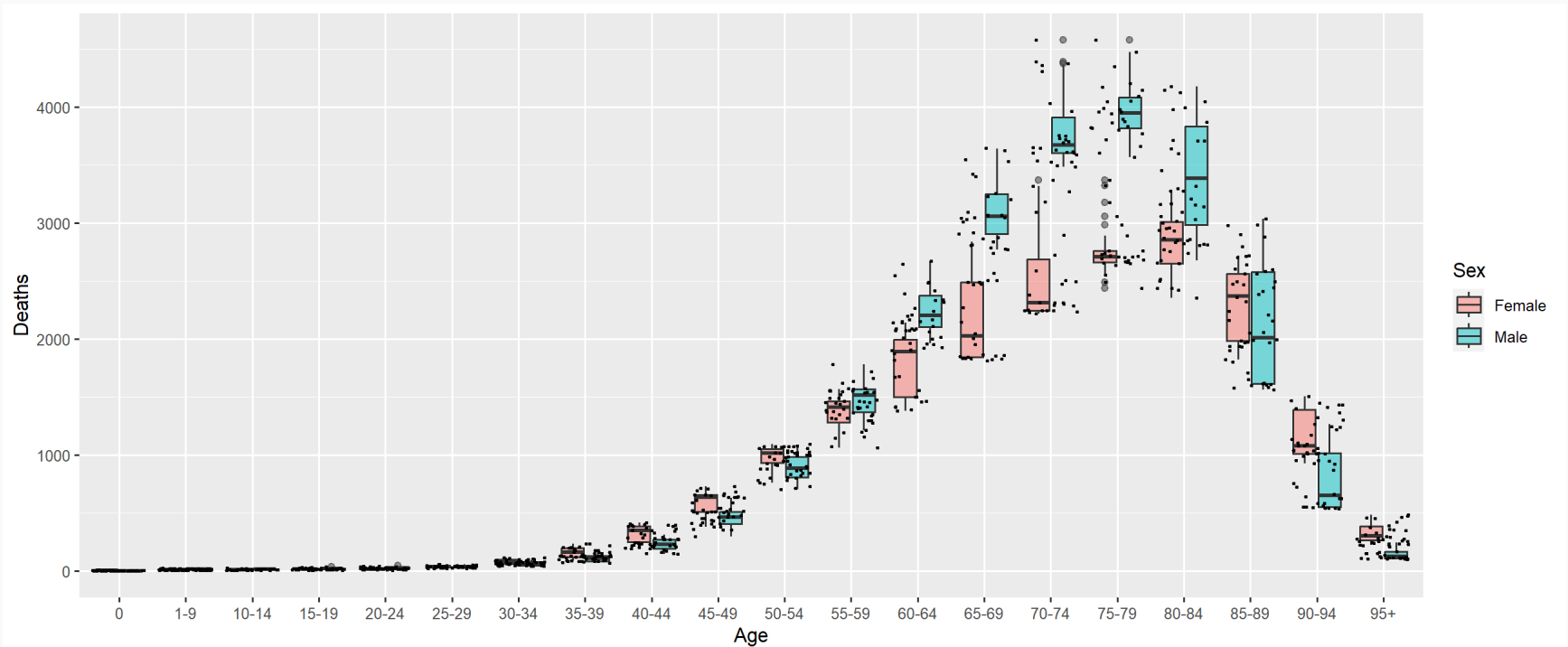
| Code | Output |
|------|--------|
|------|--------|

| | |
|--|--|
| <pre>Death_in_NL %>% filter(Age != "Total", CausesOfDeath == "Neoplasms")%>% ggplot(aes(x = Age, y = Deaths))+ geom_boxplot(aes(fill = Sex), alpha=0.5)+ geom_jitter(color="black", size=0.5, alpha=0.9)</pre> | |
|--|--|

Boxplot with individual data points

A boxplot summarizes the distribution of a continuous variable. It is often criticized for hiding the underlying data. Therefore, showing individual observations using jitter on top of boxes is a good practice.

| Code | Output |
|------|--|
| |  |



Violin plot

Violin plot is really close from a boxplot, but allows a deeper understanding of the distribution. Violins are particularly adapted when the amount of data is huge and showing individual observations gets impossible.

| Code | Output |
|------|--------|
|------|--------|

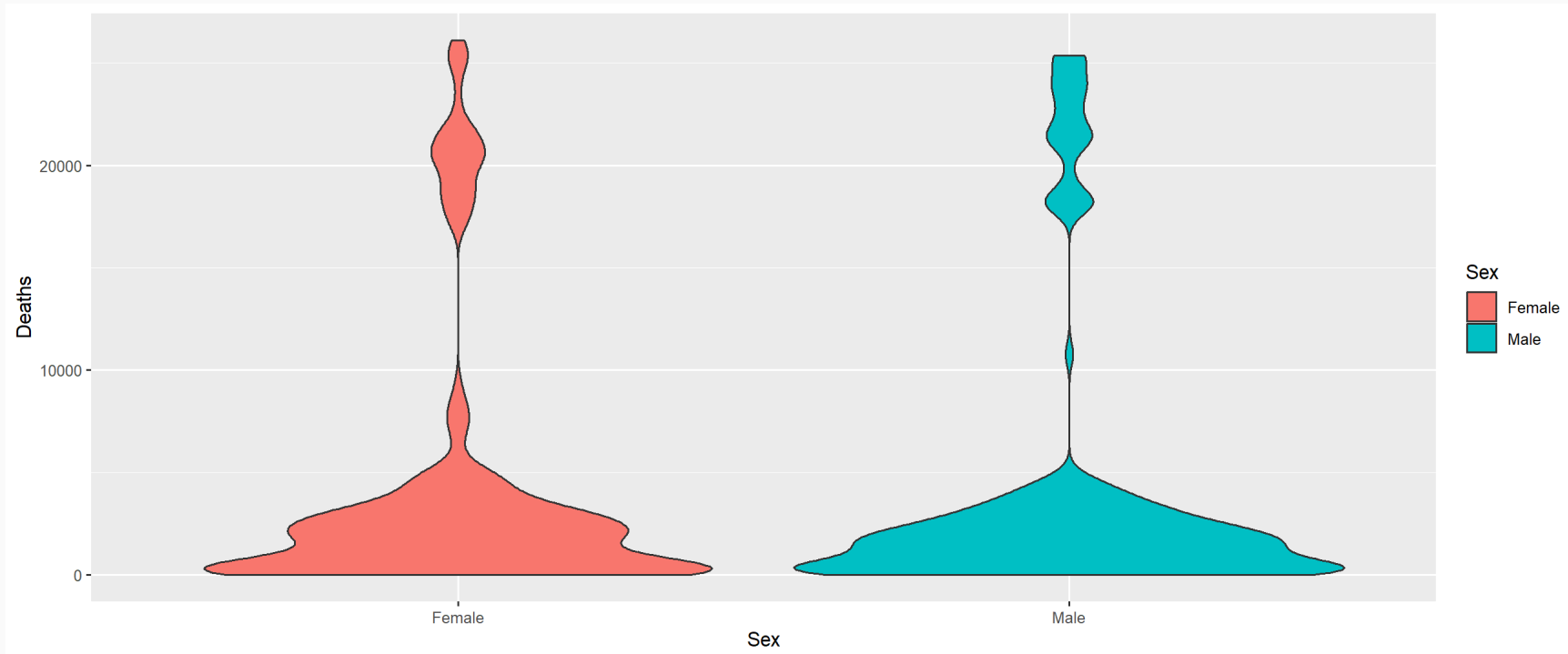
| | |
|---|--|
| <pre>Death_in_NL %>% filter(Age == "Total")%>% ggplot(aes(x = Sex, y = Deaths))+ geom_violin(aes(fill = Sex))</pre> | |
|---|--|

Violin plot

Violin plot is really close from a boxplot, but allows a deeper understanding of the distribution. Violins are particularly adapted when the amount of data is huge and showing individual observations gets impossible.

Code

Output



Your turn

1. Plot 1 from slide 14 demonstrates distributions in the number of deceased per year over the period from 1996 to 2021. And a dot plot is not the best geom to demonstrate distributions. Modify the plot using the geoms that fit better for the task (see lecture 1_5_geometries).
2. The plot from slide 17 looks good enough, although the amounts for many lines are not recognizable. Can you visualize the number of deceased from different causes of death for a particular year (2015).
3. The plot on slide 26 shows the progression in the number of deceased over time. It uses the total number of deceased, irrespective of their age. Make a new plot to visualize the number of deceased men and women depending on their age; make it for a particular year (2015) and for a particular cause of death (neoplasms).

