EJERCICIO 7

Enunciado:

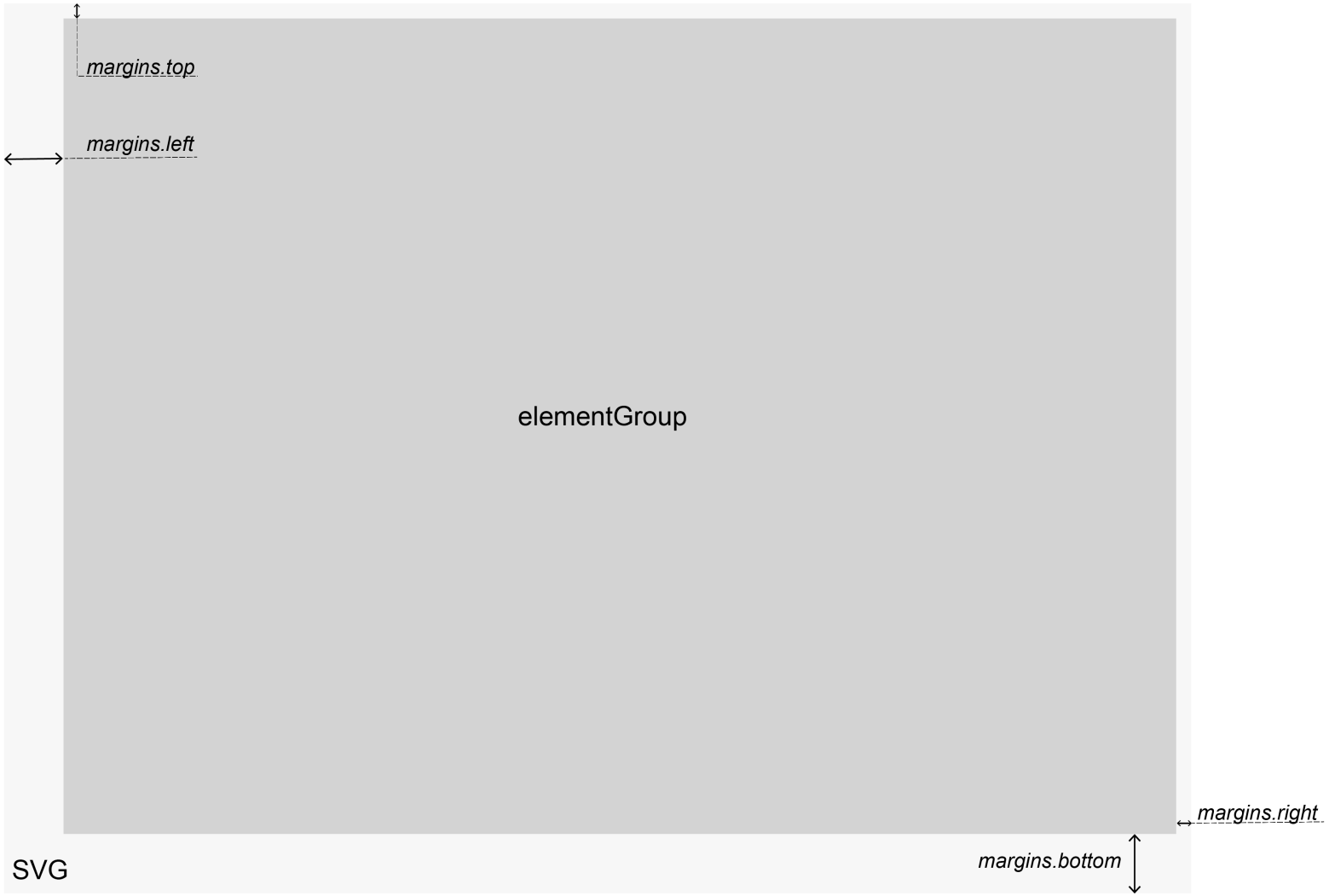
1. Escalas: definir una función de escala lineal que permita averiguar el tamaño a 1/50 de la altura de cualquier edificio. La función recibirá un dato en metros y lo hará 50 veces más pequeño.

EJERCICIO 8

Enunciado:
Datos.json contiene una serie de ciudades con la distancia desde cada una de ellas a Berlin, en km.

1. Escalas: representar en una línea horizontal la distancia en km que hay desde Berlin a las distintas ciudades. Da igual la geometría que usemos para representar dicha ciudad. Hay que incluir el nombre.

Márgenes y grupos



EJERCICIO 9

Enunciado:
Trastear con los márgenes ☺

Ejes

1. Declarar ejes y pasar las escalas
2. Trasladar los ejes
3. Llamar a los ejes

```
      const axisGroup = svg.append("g").attr("id", "axisGroup")

1
      let xAxis = d3.axisBottom().scale(x)
      let yAxis = d3.axisLeft().scale(y)

  let xAxisGroup = axisGroup.append("g").attr("id", "xAxisGroup")
2       .attr('transform', `translate(${margin.left}, ${height - margin.bottom})`)
  let yAxisGroup = axisGroup.append("g").attr("id", "yAxisGroup")
        .attr('transform', `translate(${margin.left}, ${margin.top})`)

      xAxisGroup.call(xAxis)
3
      yAxisGroup.call(yAxis)
```

EJERCICIO 10

Enunciado:
Usando los datos de la distancia entre ciudades, añadir el eje inferior, mostrando la distancia.

Animaciones e Interactividad

eventos

Son como sensores que enlazamos a elementos y que detectan cuando sucede algo concreto y a los que podemos enlazar una función. Hay de muchos tipos, pero principalmente vamos a usar eventos de ratón, que son aquellos que tienen que ver con la interacción del ratón con los elementos.

ejemplos

- onmouseover
- onmouseout
- click

Animaciones e Interactividad

transiciones

Las transiciones suavizan los cambios que sufren los elementos, son muy útiles para que el usuario detecte con facilidad lo que está ocurriendo. Casi cualquier atributo se puede animar. Típicamente, aquellos atributos que sean numéricos, pero también se pueden animar los colores. Lo que no se pueden animar son clases o ids.

EJERCICIO 11

Enunciado:

Part I:

En el ejercicio de las ciudades, añadir el nombre de la ciudad y animar su aparición.

Parte II:

Usando los datos de la distancia entre ciudades, cambiar de color la figura de la ciudad cuando se selecciona, y añadir el nombre de la misma cuando se posa el cursor.

EJERCICIO 12

Enunciado:
Crear una gráfica de globos su contenido.

Ayuda:
Hay que ejecutar el servidor de datos y llamar al endpoint que contiene los datos.

d3.interval(fn, time) permite hacer llamadas recurrentes en intervalos del tiempo especificado. En cada llamada el servidor retornará datos distintos.

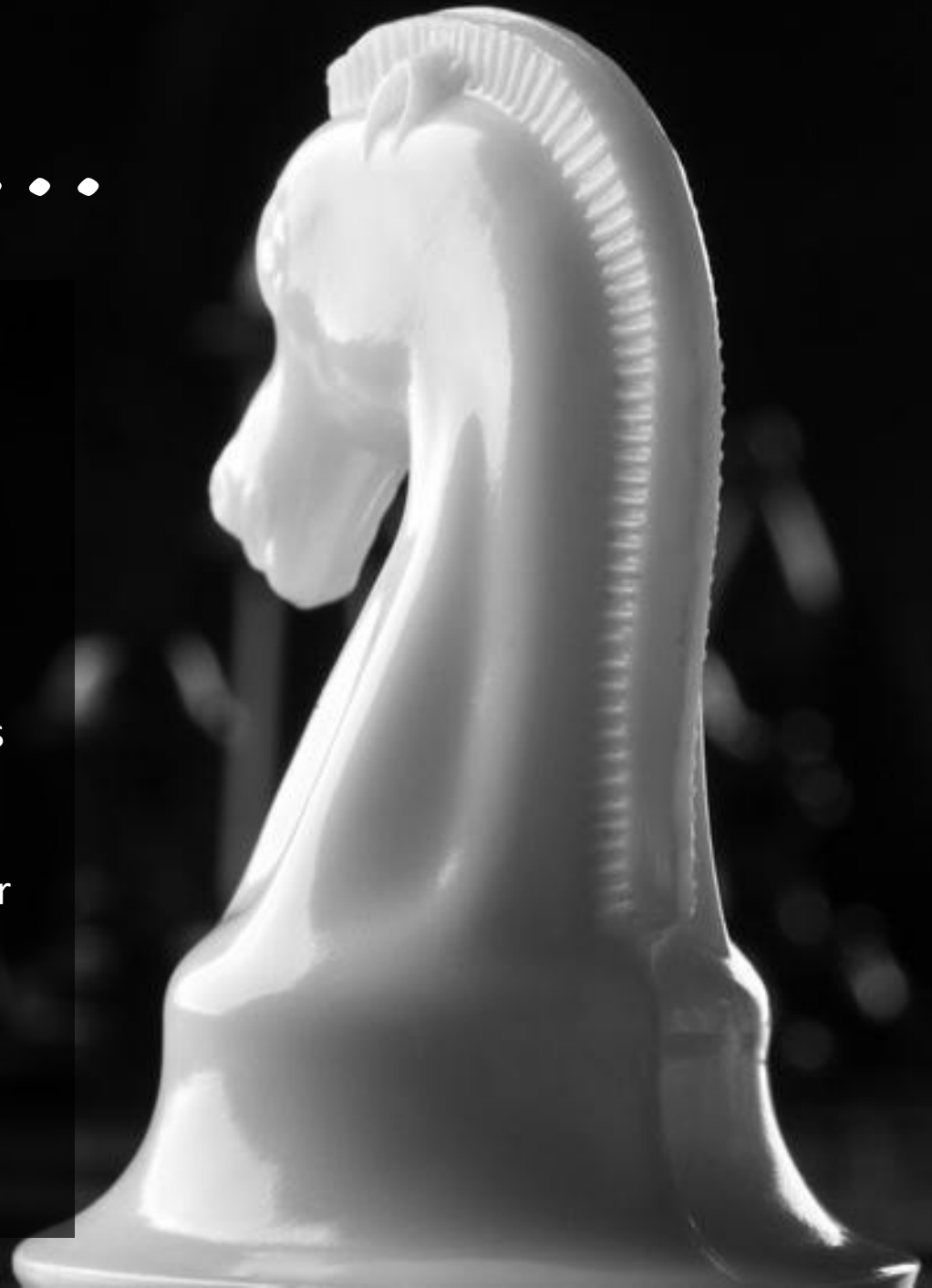
Hay que intentar que los cambios de datos estén animados.

EJERCICIO FINAL...

Enunciado:

1. Crear una gráfica de barras que represente cuántos mundiales tiene cada equipo.

Bonus: Conectar el slider para que la gráfica se modifique para ver quién tiene más mundiales hasta ese año



EJERCICIO FINAL...

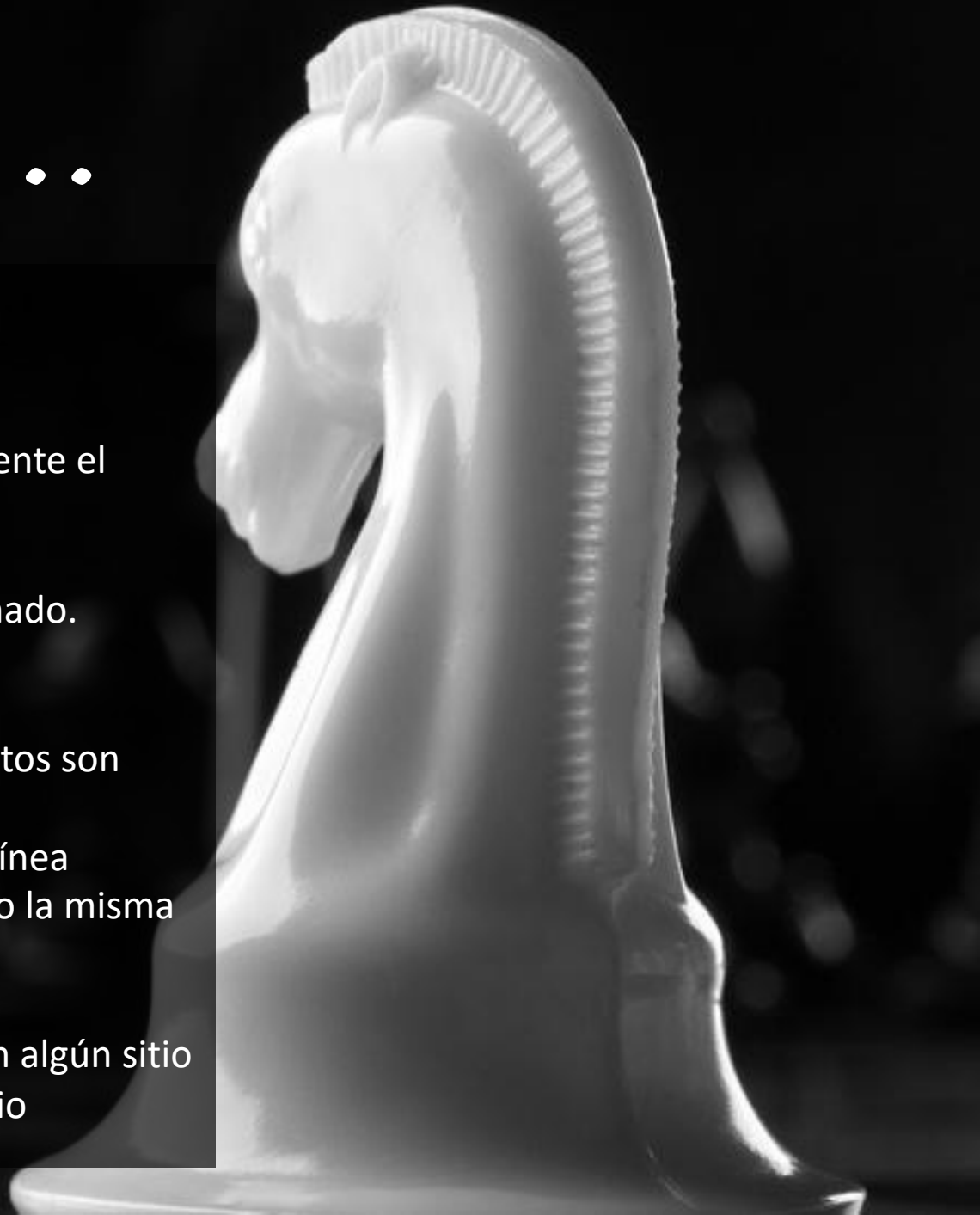
Enunciado:

Opción 1 (difícil)

1. Representar datos del IBEX en una gráfica de línea que represente el valor de cierre de mercado.
2. Representar los valores de volumen en la misma gráfica.
3. Crear un tooltip para ver el resto de valores en el día seleccionado.

Opción 2 (asequible)

1. En este caso también hay que combinar 2 gráficas, pero los datos son mucho más sencillos.
2. Representar la edad de Leonardo Di Caprio en una gráfica de línea
3. Representar la edad de sus ex en una gráfica de barras, usando la misma escala, para poder comparar ambos valores .
4. Cada chica debe tener un color distinto
5. Cuando se pase el cursor por encima, debemos representar en algún sitio el nombre de ella, su edad y la diferencia de edad con Di Caprio

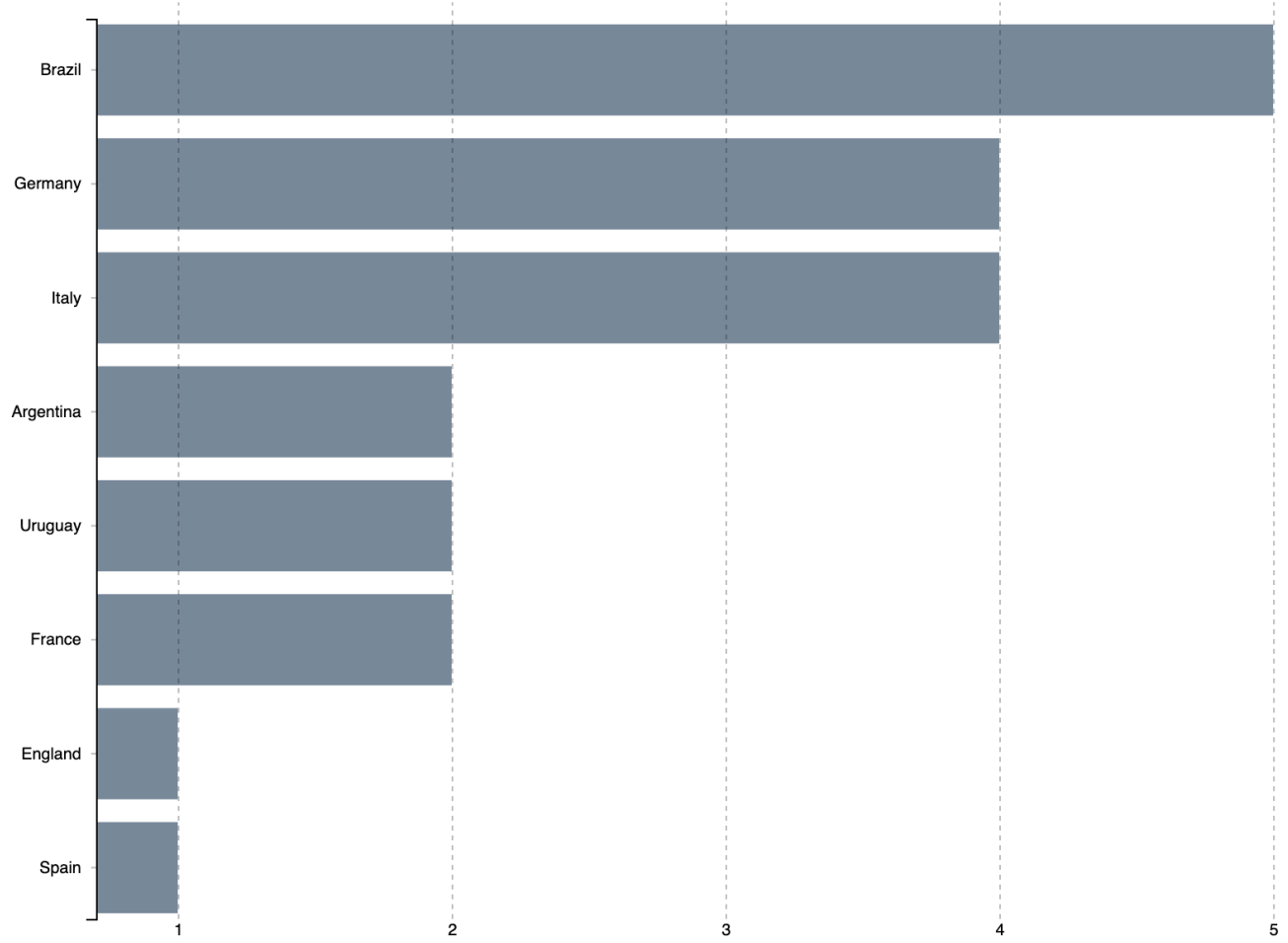


¿Ha quedado todo claro?

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones



Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
const width = 800
const height = 600
const margin = {
    top: 10,
    right: 10,
    bottom: 40,
    left: 70
}
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
const svg = d3.select("#chart")
    .append("svg")
    .attr("id", "svg")
    .attr("width", width)
    .attr("height", height)

const elementGroup = svg.append("g")
    .attr("id", "elementGroup")
    .attr('transform', `translate(${0}, ${margin.top})`)
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
let y = d3.scaleBand()
    .range([0, height - margin.top - margin.bottom])
    .paddingInner(0.2).paddingOuter(0.05)
let x = d3.scaleLinear()
       .range([50, width - margin.left - margin.right])
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
const axisGroup = svg.append("g")
        .attr("id", "axisGroup")

const xAxisGroup = axisGroup
    .append("g")
    .attr("id", "xAxisGroup")
    .attr('transform', `translate(
        ${margin.left},
        ${height - margin.bottom})`)

const yAxisGroup = axisGroup
    .append("g")
    .attr("id", "yAxisGroup")
    .attr('transform', `translate(
        ${margin.left},
        ${margin.top})`)

const xAxis = d3.axisBottom()
        .scale(x).ticks(5).tickSize(-height)
const yAxis = d3.axisLeft().scale(y)
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(data => {
    data = Object.values(data.worldCupWinners)
    data.map(d => d.titles = +d.titles)

    x.domain(d3.extent(data.map(d=>d.titles)))
    y.domain(data.map(d => d.country))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    xAxisGroup.select('.domain').remove()

    elementGroup.selectAll('.bar').data(data)
        .call(drawBar)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(data => {
    data = Object.values(data.worldCupWinners)
    data.map(d => d.titles = +d.titles)

    x.domain(d3.extent(data.map(d=>d.titles)))
    y.domain(data.map(d => d.country))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    xAxisGroup.select('.domain').remove()

    elementGroup.selectAll('.bar').data(data)
        .call(drawBar)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(data => {
    data = Object.values(data.worldCupWinners)
    data.map(d => d.titles = +d.titles)

    x.domain(d3.extent(data.map(d=>d.titles)))
    y.domain(data.map(d => d.country))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    xAxisGroup.select('.domain').remove()

    elementGroup.selectAll('.bar').data(data)
        .call(drawBar)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(data => {
    data = Object.values(data.worldCupWinners)
    data.map(d => d.titles = +d.titles)

    x.domain(d3.extent(data.map(d=>d.titles)))
    y.domain(data.map(d => d.country))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    xAxisGroup.select('.domain').remove()

    elementGroup.selectAll('.bar').data(data)
        .call(drawBar)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(data => {
    data = Object.values(data.worldCupWinners)
    data.map(d => d.titles = +d.titles)

    x.domain(d3.extent(data.map(d=>d.titles)))
    y.domain(data.map(d => d.country))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    xAxisGroup.select('.domain').remove()

    elementGroup.selectAll('.bar').data(data)
        .call(drawBar)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
function drawBar(group) {
    group.enter().append("rect")
        .attr('id', d => d.country)
        .attr('class', 'bar')
        .attr('height', y.bandwidth())
        .attr('width', d => x(d.titles))
        .attr('x', margin.left)
        .attr('y', d => y(d.country))
}
```

Bloque 1:

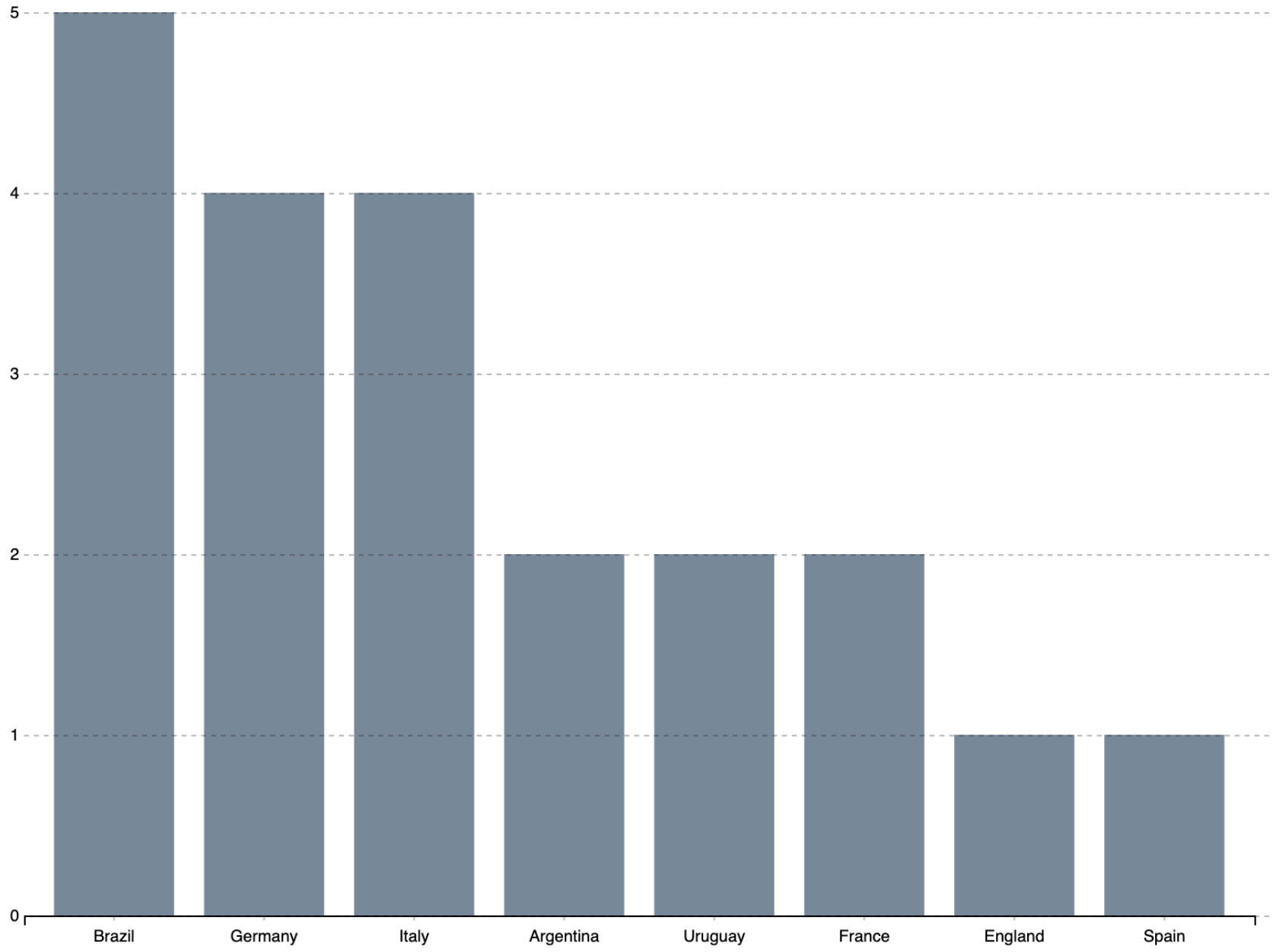
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call

Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:

Definición de funciones



Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
const width = 800
const height = 600

const margin = {
top: 10,
bottom: 40,
left: 40,
right: 10
}
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
const svg = d3.select("#chart").append("svg")
      .attr('id', "svg").attr("width", width)
      .attr("height", height)
const elementGroup = svg.append("g")
      .attr('id', "elementGroup")
      .attr("transform", `translate(
            ${margin.left}, ${margin.top})`)
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
let x = d3.scaleBand()
        .range([0, width - margin.left - margin.right])
        .padding(0.2)
let y = d3.scaleLinear()
        .range([height - margin.top - margin.bottom, 0])
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
const axisGroup = svg.append("g")
      .attr('id', "axisGroup")
const xAxisGroup = axisGroup.append("g")
      .attr("id", "xAxisGroup")
      .attr("transform", `translate(
            ${margin.left}, ${height - margin.bottom})`)
const yAxisGroup = axisGroup.append("g")
      .attr("id", "yAxisGroup")
      .attr("transform",
            `translate(${margin.left}, ${margin.top})`)

const xAxis = d3.axisBottom().scale(x)
const yAxis = d3.axisLeft().scale(y)
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json("data.json").then(data => {
    data = Object.values(data.worldCupWinners)
    console.log(data)

    x.domain(data.map(d => d.country))
    y.domain([0, d3.max(data.map(d => d.titles))])

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis.ticks(5).tickSize(-width))
    yAxisGroup.select('.domain').remove()

    let bars = elementGroup.selectAll('rect').data(data)
    bars.enter().append('rect')
        .attr('class', 'bar')
        .attr('x', d => x(d.country))
        .attr('y', d => y(d.titles))
        .attr('width', x.bandwidth())
        .attr('height', d =>
                height -
                y(d.titles) - margin.bottom - margin.top)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json("data.json").then(data => {
    data = Object.values(data.worldCupWinners)
    console.log(data)

    x.domain(data.map(d => d.country))
    y.domain([0, d3.max(data.map(d => d.titles))])

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis.ticks(5).tickSize(-width))
    yAxisGroup.select('.domain').remove()

    let bars = elementGroup.selectAll('rect').data(data)
    bars.enter().append('rect')
        .attr('class', 'bar')
        .attr('x', d => x(d.country))
        .attr('y', d => y(d.titles))
        .attr('width', x.bandwidth())
        .attr('height', d =>
                height -
                y(d.titles) - margin.bottom - margin.top)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json("data.json").then(data => {
    data = Object.values(data.worldCupWinners)
    console.log(data)

    x.domain(data.map(d => d.country))
    y.domain([0, d3.max(data.map(d => d.titles))])

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis.ticks(5).tickSize(-width))
    yAxisGroup.select('.domain').remove()

    let bars = elementGroup.selectAll('rect').data(data)
    bars.enter().append('rect')
        .attr('class', 'bar')
        .attr('x', d => x(d.country))
        .attr('y', d => y(d.titles))
        .attr('width', x.bandwidth())
        .attr('height', d =>
                height -
                y(d.titles) - margin.bottom - margin.top)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json("data.json").then(data => {
    data = Object.values(data.worldCupWinners)
    console.log(data)

    x.domain(data.map(d => d.country))
    y.domain([0, d3.max(data.map(d => d.titles))])

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis.ticks(5).tickSize(-width))
    yAxisGroup.select('.domain').remove()

    let bars = elementGroup.selectAll('rect').data(data)
    bars.enter().append('rect')
        .attr('class', 'bar')
        .attr('x', d => x(d.country))
        .attr('y', d => y(d.titles))
        .attr('width', x.bandwidth())
        .attr('height', d =>
                height -
                y(d.titles) - margin.bottom - margin.top)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json("data.json").then(data => {
    data = Object.values(data.worldCupWinners)
    console.log(data)

    x.domain(data.map(d => d.country))
    y.domain([0, d3.max(data.map(d => d.titles))])

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis.ticks(5).tickSize(-width))
    yAxisGroup.select('.domain').remove()

    let bars = elementGroup.selectAll('rect').data(data)
    bars.enter().append('rect')
        .attr('class', 'bar')
        .attr('x', d => x(d.country))
        .attr('y', d => y(d.titles))
        .attr('width', x.bandwidth())
        .attr('height', d =>
                height -
                y(d.titles) - margin.bottom - margin.top)
})
```

Bloque 1:

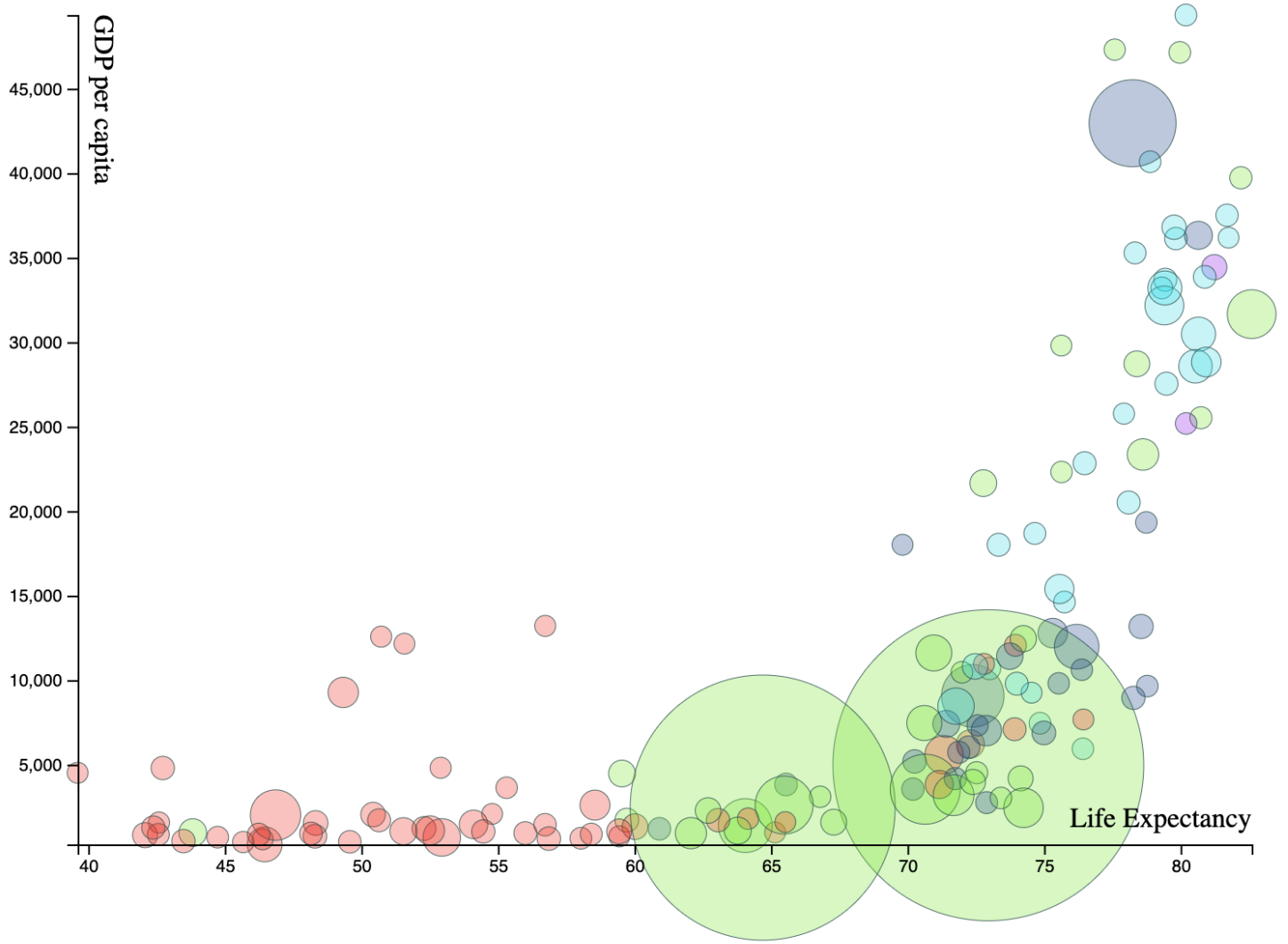
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call

Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:

Definición de funciones



Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// const
const width = 800
const height = 600
const margin = {
    top: 40,
    right: 40,
    bottom: 80,
    left: 80
}
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
// svg:
const svg = d3.select("#chart")
    .append('svg')
    .attr('id', "svg")
    .attr("width", width)
    .attr("height", height)

const elementGroup = svg
    .append("g")
    .attr('id', "elementGroup")
    .attr('transform', `translate(
        ${margin.left}, ${margin.top})`)

const axisGroup = svg
    .append("g")
    .attr("id", "axisGroup")
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
// axis & scales:
let x = d3.scaleLinear()
      .range([0, width - margin.left - margin.right])

let y = d3.scaleLinear()
      .range([height - margin.top - margin.bottom, 0])

let z = d3.scaleLinear()
      .range([6, 90])
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
let xAxisGroup = axisGroup
    .append("g")
    .attr("id", "xAxisGroup")
    .attr("transform", `translate(
        ${margin.left}, ${height - margin.bottom})`)
let yAxisGroup = axisGroup
    .append("g")
    .attr("id", "yAxisGroup")
    .attr("transform", `translate(
        ${margin.left}, ${margin.top})`)

let xAxis = d3.axisBottom().scale(x)
let yAxis = d3.axisLeft().scale(y)

let xAxisLabel = axisGroup
    .append('text')
    .text("Life Expectancy")
    .attr('transform', `translate(
        ${width - margin.right}, ${height - margin.bottom -10})`)
    .attr('text-anchor', 'end')
let yAxisLabel = axisGroup
    .append('text')
    .text("GDP per capita")
    .attr('transform', `translate(
        ${margin.left + 10}, ${margin.top}) rotate(90)`)
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// data
d3.csv('data.csv').then(data => {
    data.map(d => {
        d.GDPpc = +d.GDPpc
        d.lifeExpectancy = +d.lifeExpectancy
        d.population = +d.population
    })

    x.domain(d3.extent(data.map(d=>d.lifeExpectancy)))
    y.domain(d3.extent(data.map(d=>d.GDPpc)))
    z.domain(d3.extent(data.map(d=>d.population)))

    xAxisGroup.call(xAxis)
    yAxisGroup.call(yAxis)

    // data binding:
    let elements =
    elementGroup.selectAll('.country').data(data)
    elements.enter().call(drawCircles)

})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
function drawCircles(group){
    group.append('circle')
        .attr('class', d=> `country ${d.continent}`)
        .attr("cx", d=> x(d.lifeExpectancy))
        .attr("cy", d=> y(d.GDPpc))
        .attr("r", d=> z(d.population))
}
```

Bloque 1:

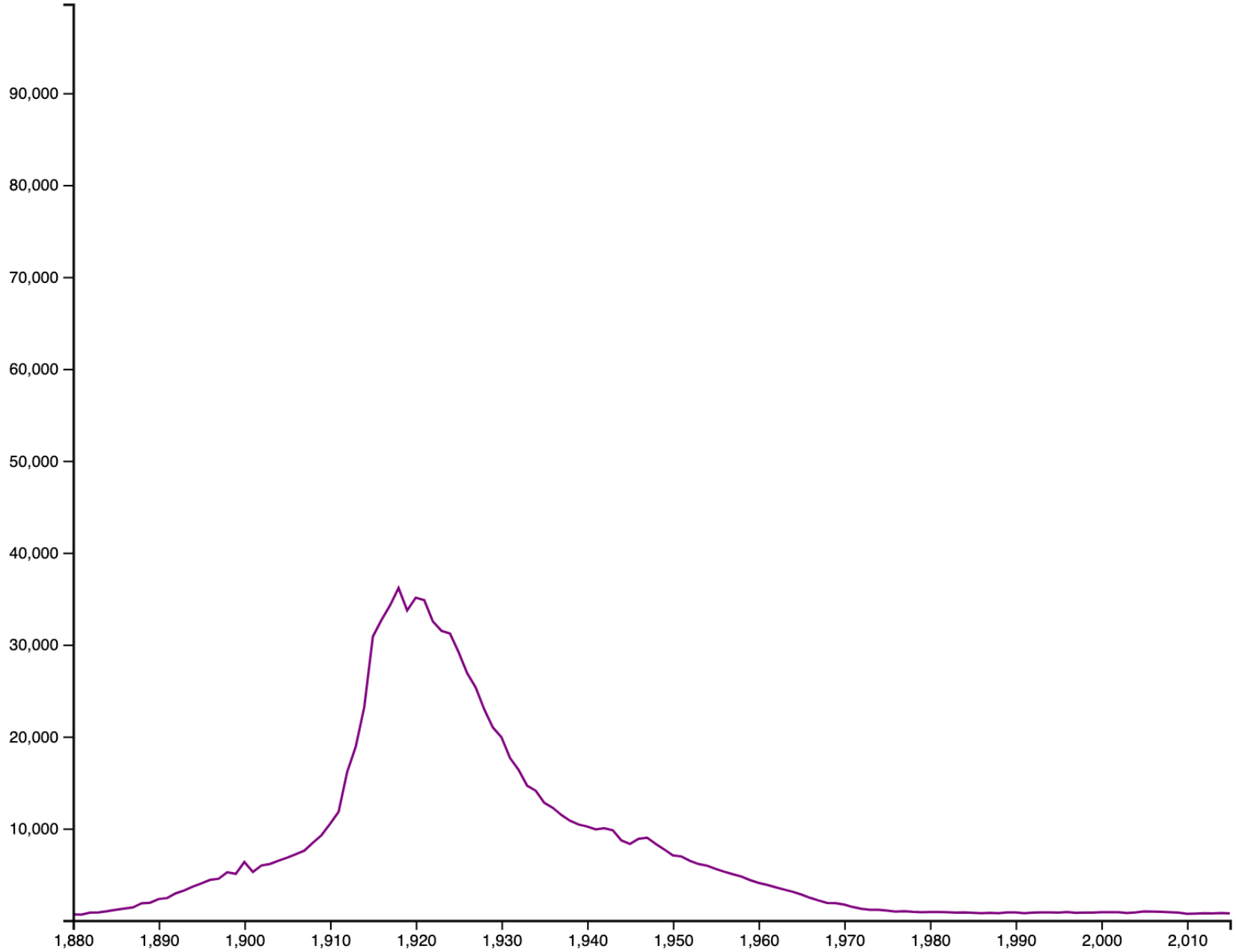
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call

Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:

Definición de funciones



Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// const
const width = 800
const height = 600
const margin = {
top: 10,
bottom: 20,
left: 40,
right: 40
}
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
// svg
const svg = d3.select("#chart")
        .append('svg')
            .attr("id", "svg")
            .attr("width", width)
            .attr("height", height)

let elementGroup = svg
        .append("g").attr("id", "elementGroup")
        .attr('transform', `translate(
                ${margin.left},
                ${margin.top})`)
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
let xAxis = d3.axisBottom().scale(x)
let yAxis = d3.axisLeft().scale(y)
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
let axisGroup = svg.append("g")
        .attr("id", "axisGroup")
let xAxisGroup = axisGroup
        .append("g").attr("id", "xAxisGroup")
        .attr('transform', `translate(
                ${margin.left},
                ${height - margin.bottom})`)
let yAxisGroup = axisGroup
        .append("g")
                .attr("id", "yAxisGroup")
                .attr('transform', `translate(
                        ${margin.left},
                        ${margin.top})`)
```

```
let xAxis = d3.axisBottom().scale(x)
let yAxis = d3.axisLeft().scale(y)
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.csv("data.csv").then(data => {
    data.map(d => {
        d.year = +d.year
        d.prop = +d.prop
        d.n = +d.n
    })
// scale domain:
x.domain(d3.extent(data.map(d => d.year)))
y.domain(d3.extent(data.map(d => d.n)))

// call axes
xAxisGroup.call(xAxis)
yAxisGroup.call(yAxis)
let Helen = data.filter(d => d.name == "Helen")

// datum!
elementGroup.datum(Helen).append('path')
    .attr("id", "Helen")
    .attr("d", d3.line()
    .x(d => x(d.year))
    .y(d => y(d.n))
    )
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.csv("data.csv").then(data => {
    data.map(d => {
        d.year = +d.year
        d.prop = +d.prop
        d.n = +d.n
    })
// scale domain:
x.domain(d3.extent(data.map(d => d.year)))
y.domain(d3.extent(data.map(d => d.n)))

// call axes
xAxisGroup.call(xAxis)
yAxisGroup.call(yAxis)
let Helen = data.filter(d => d.name == "Helen")

// datum!
elementGroup.datum(Helen).append('path')
    .attr("id", "Helen")
    .attr("d", d3.line()
    .x(d => x(d.year))
    .y(d => y(d.n))
    )
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.csv("data.csv").then(data => {
    data.map(d => {
        d.year = +d.year
        d.prop = +d.prop
        d.n = +d.n
    })
// scale domain:
x.domain(d3.extent(data.map(d => d.year)))
y.domain(d3.extent(data.map(d => d.n)))

// call axes
xAxisGroup.call(xAxis)
yAxisGroup.call(yAxis)
let Helen = data.filter(d => d.name == "Helen")

// datum!
elementGroup.datum(Helen).append('path')
    .attr("id", "Helen")
    .attr("d", d3.line()
    .x(d => x(d.year))
    .y(d => y(d.n))
    )
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.csv("data.csv").then(data => {
    data.map(d => {
        d.year = +d.year
        d.prop = +d.prop
        d.n = +d.n
    })
// scale domain:
x.domain(d3.extent(data.map(d => d.year)))
y.domain(d3.extent(data.map(d => d.n)))

// call axes
xAxisGroup.call(xAxis)
yAxisGroup.call(yAxis)
let Helen = data.filter(d => d.name == "Helen")

// datum!
elementGroup.datum(Helen).append('path')
    .attr("id", "Helen")
    .attr("d", d3.line()
    .x(d => x(d.year))
    .y(d => y(d.n))
    )
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

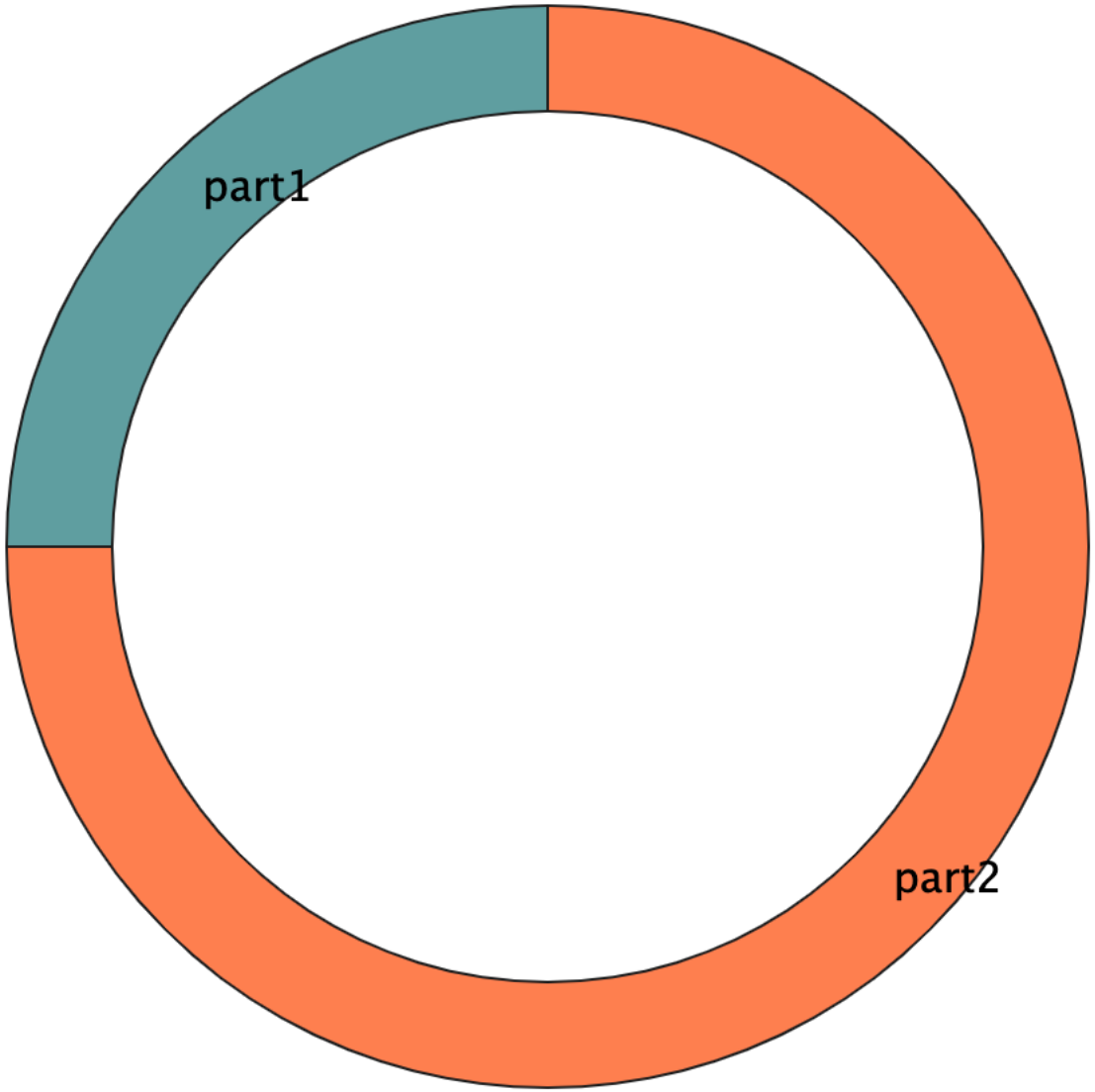
Bloque 3:
Definición de funciones

```
d3.csv("data.csv").then(data => {
    data.map(d => {
        d.year = +d.year
        d.prop = +d.prop
        d.n = +d.n
    })
// scale domain:
x.domain(d3.extent(data.map(d => d.year)))
y.domain(d3.extent(data.map(d => d.n)))

// call axes
xAxisGroup.call(xAxis)
yAxisGroup.call(yAxis)
let Helen = data.filter(d => d.name == "Helen")

// datum!
elementGroup.datum(Helen).append('path')
    .attr("id", "Helen")
    .attr("d", d3.line()
    .x(d => x(d.year))
    .y(d => y(d.n))
    )
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones



Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
// constants:
const width = 450
const height = 450
const margin = 20
const radius = (width - (2 * margin)) / 2
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
// define SVG
const svg = d3.select("#chart")
        .append('svg’)
        .attr('id', 'svg’)
        .attr("width", width)
        .attr("height", height)

const elementGroup = svg
        .append('g’)
                .attr('id', "elementGroup")
                .attr('transform', `translate(
                        ${width/2}, ${height/2})`)
```

Bloque 1:

Definir Constantes

Declarar el svg

Definir escala

Definir ejes

Tooltip

Bloque 2: data call

Formato de datos

Añadir el dominio a la escala

Llamar a los ejes

Data binding

Patrón de actualización

Bloque 3:

Definición de funciones

```
// pie scales:
let pie = d3.pie().value(function(d) {
    return d.value
})

let arc = d3.arc()
    .innerRadius(radius - 40)
    .outerRadius(radius)
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(_data => {
// data transform
let data = pie(d3.entries(_data))

// data binding:
let sector =
elementGroup.selectAll('.sector').data(data)

sector.enter()
.append('g').call(drawSlices)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(_data => {
// data transform
let data = pie(d3.entries(_data))

// data binding:
let sector =
elementGroup.selectAll('.sector').data(data)

sector.enter()
.append('g').call(drawSlices)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(_data => {
// data transform
let data = pie(d3.entries(_data))

// data binding:
let sector =
elementGroup.selectAll('.sector').data(data)

sector.enter()
.append('g').call(drawSlices)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
d3.json('data.json').then(_data => {
// data transform
let data = pie(d3.entries(_data))

// data binding:
let sector =
elementGroup.selectAll('.sector').data(dat
a)

sector.enter()
.append('g').call(drawSlices)
})
```

Bloque 1:
Definir Constantes
Declarar el svg
Definir escala
Definir ejes
Tooltip

Bloque 2: data call
Formato de datos
Añadir el dominio a la escala
Llamar a los ejes
Data binding
Patrón de actualización

Bloque 3:
Definición de funciones

```
function drawSlices(group) {
    group.attr('class', 'sector')
        .append('path')
            .attr('class', d => d.data.key)
            .attr('d', arc)
    group.append('text')
        .attr('transform',
                d =>`translate(${arc.centroid(d)})`)
        .text(d => d.data.key)
}
```

Constantes

```
const width = 800
const height = 400
const margin = {
    top: 50,
    right: 50,
    left: 50,
    bottom: 50
}
```

Declaro svg

```
const svg = d3.select('#map').append('svg').attr("width", width).attr("height", height)
const elementGroup = svg.append("g").attr("id", "elementGroup")
```

Declaro la proyección

```
const projection = d3.geoMercator().translate([width / 2, height / 2]).scale(100)
let path = d3.geoPath().projection(projection)
```

Data

```
d3.json("world.topojson").then(data => {

    let countries = topojson.feature(data, data.objects.countries)

    let map = elementGroup.selectAll("path.country").data(countries.features)
    map.call(drawMap)
})
```

Función que dibuja el mapa

```
function drawMap(selection) {
    selection.enter()
    .append('path')
        .attr('class', 'country')
        .attr("d", path)
        .on("mouseover", function() {
            d3.selectAll("path.country").classed("hover", false)
            d3.select(this).classed("hover", true)
        })
}
```