Herramienta para la exploración de tendencias y detección de patrones epidemiológicos en Argentina

Morales Arturo Leonardo, Figueroa Marcelo , Delrieux Claudio, Ramallo Virginia, Dipierri José Edgardo

CONICET
IPCSH
Instituto Patagónico de Ciencias Sociales y Humanas
"Dra. María Florencia del Castillo Bernal"

GIBEH

LCI
LABORATORIO DE CIENCIAS DE LAS IMAGENES

Motivación

El estudio de las causas de los fallecimientos constituye una fuente de información fundamental para la planificación y elaboración de políticas públicas de salud de un país.

Motivación

Saber por qué (a causa de qué) mueren las personas, constituye un insumo fundamental para entender cómo viven las personas.

Objetivos

Analizar el comportamiento temporal y espacial de los fallecimientos por causas específicas en la Argentina.

Objetivos

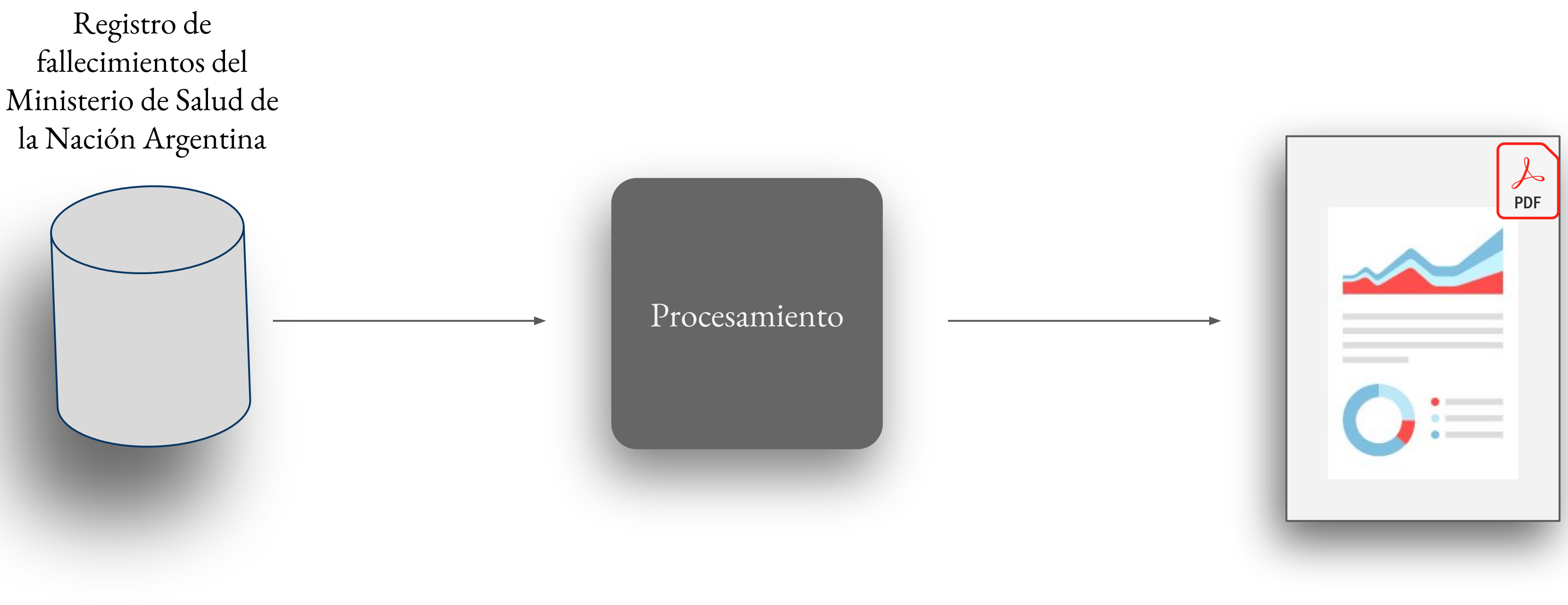
Analizar el comportamiento temporal y espacial de fallecimientos por causas específicas en la Argentina.

Repetir el análisis de forma sencilla, con diferentes conjuntos de causas específicas de fallecimientos, respetando los sistemas de codificación CIE-9 o CIE-10.

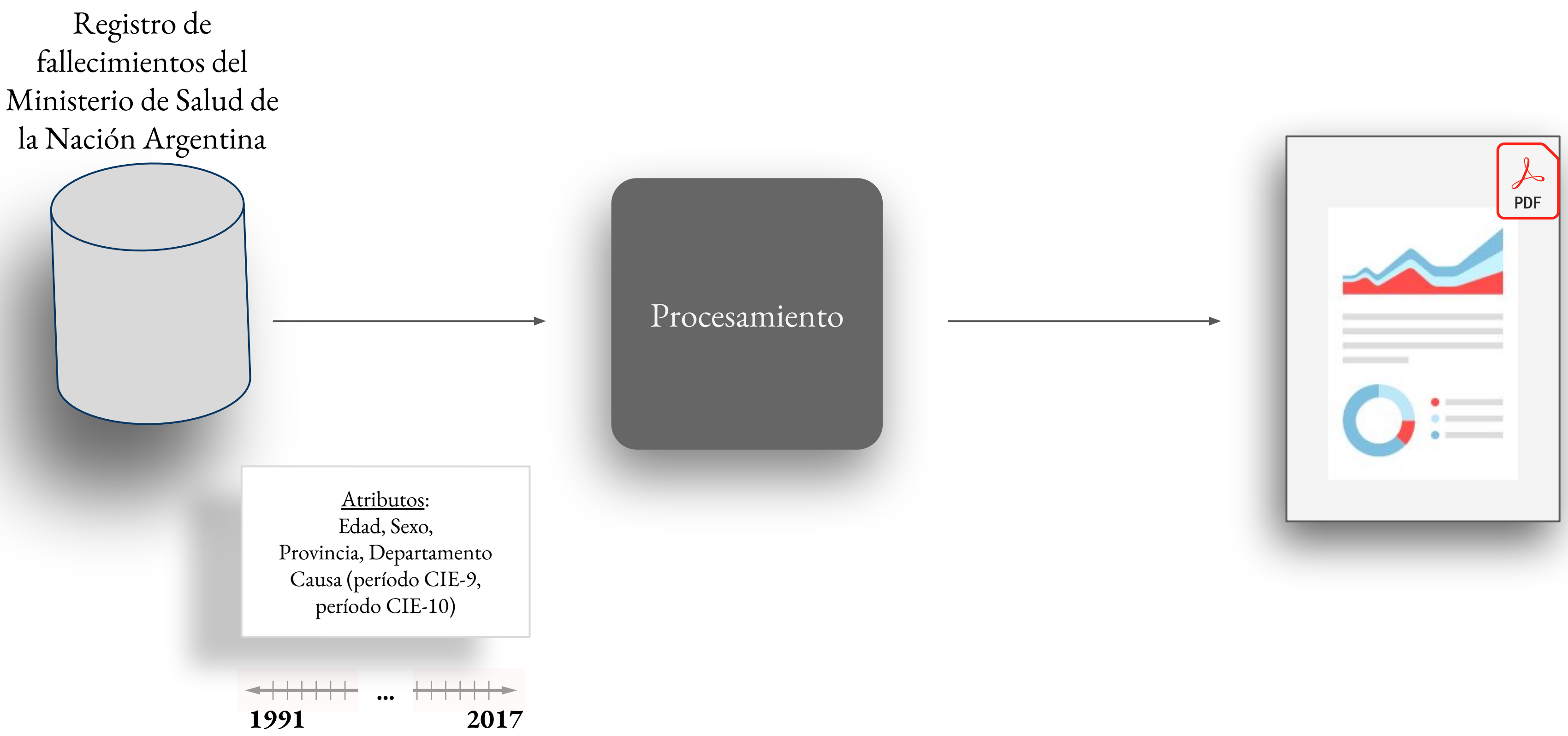
Herramienta

Pipeline, entradas, procesamiento y salidas

Pipeline, entradas, procesamiento y salidas

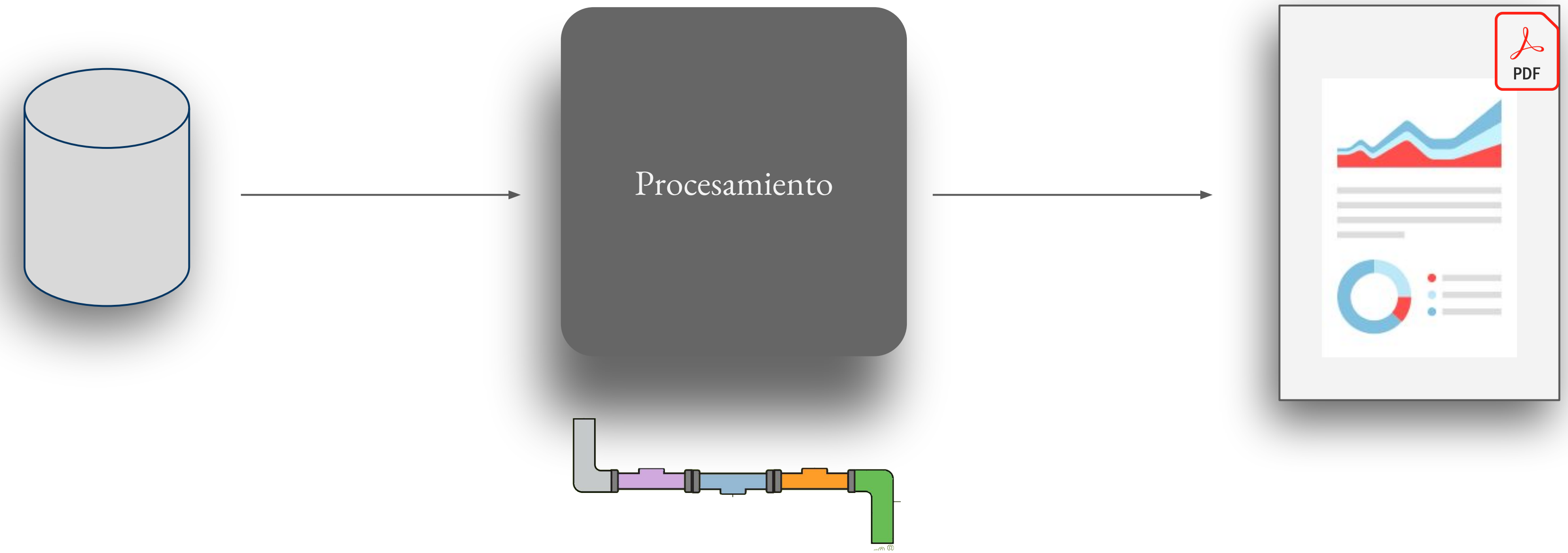


Pipeline, entradas, procesamiento y salidas

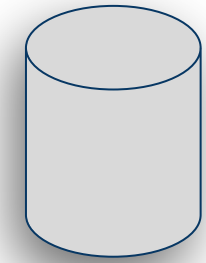


Pipeline, entradas, procesamiento y salidas

Pipeline, entradas, procesamiento y salidas

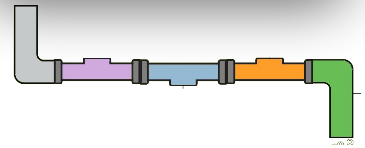


Pipeline, entradas, procesamiento y salidas

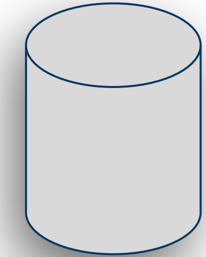


Procesamiento

Perspectiva:
Nacional,
Regional,
Provincial,
Departamental

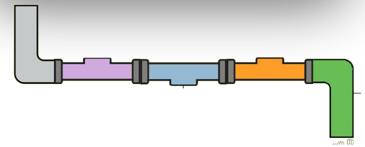


Pipeline, entradas, procesamiento y salidas



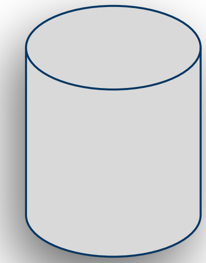
Procesamiento

Perspectiva:
Nacional,
Regional,
Provincial,
Departamental



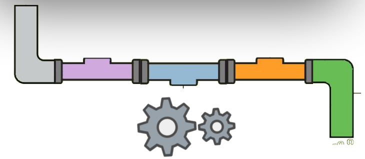
Análisis:
Sexo,
Grupo etario,
Año,
Período,
etc

Pipeline, entradas, procesamiento y salidas



Perspectiva:
Nacional,
Regional,
Provincial,
Departamental

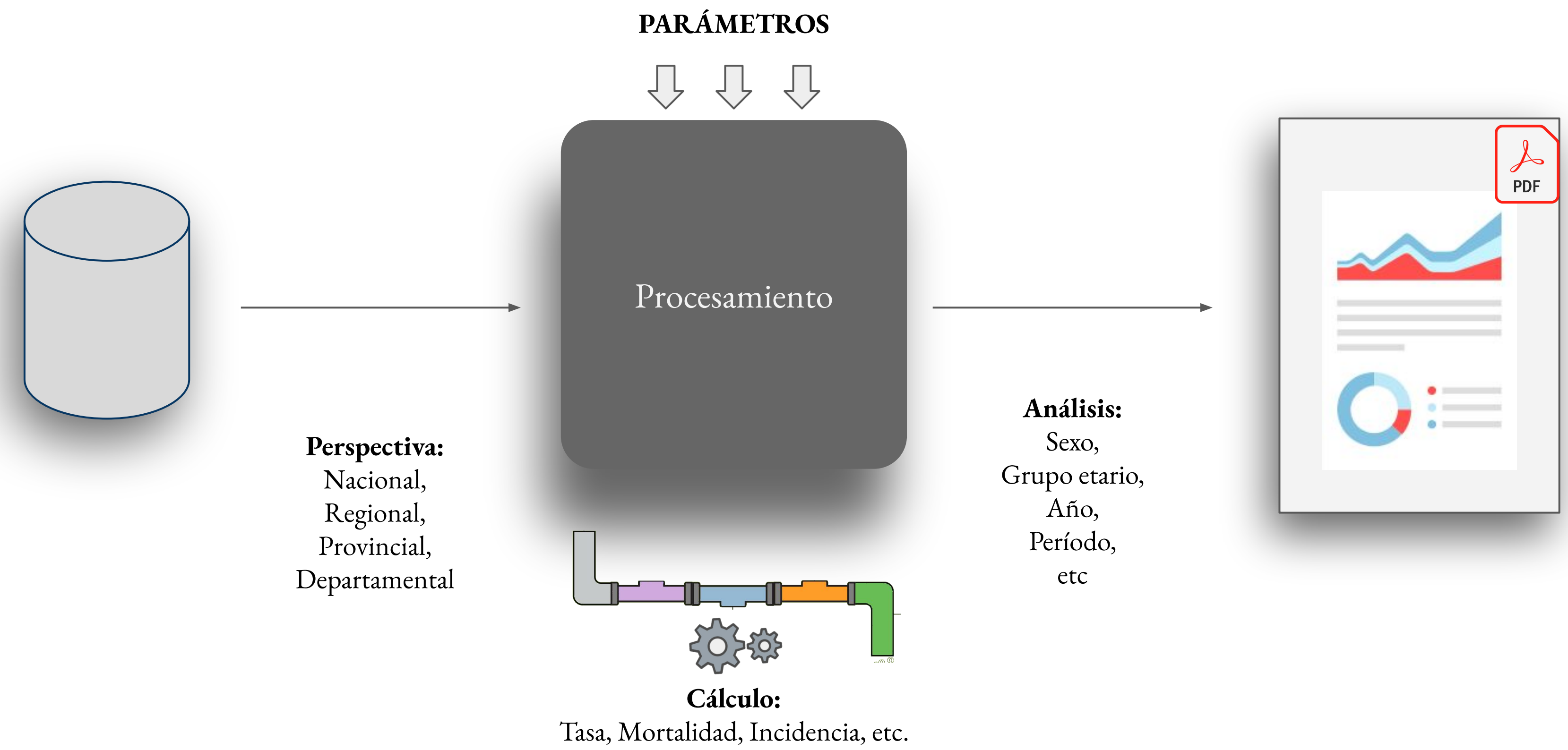
Procesamiento



Cálculo:
Tasa, Mortalidad, Prevalencia, etc.

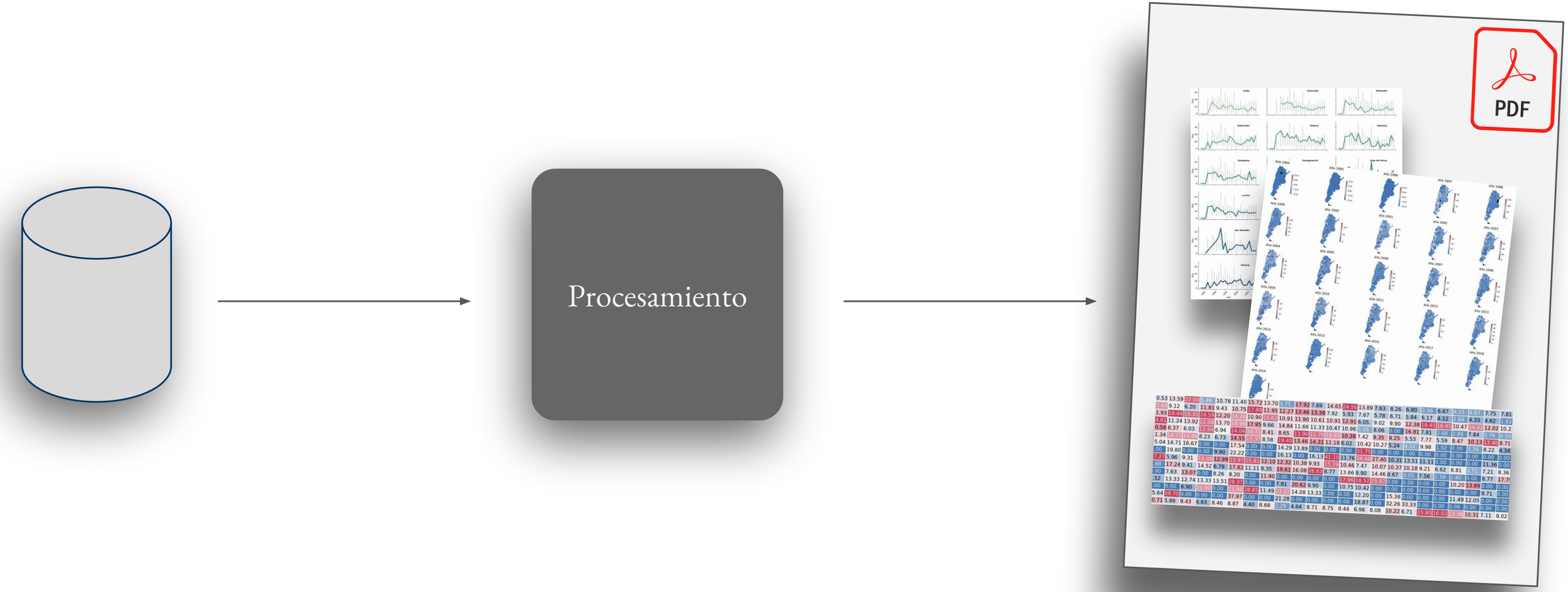
Análisis:
Sexo,
Grupo etario,
Año,
Período,
etc

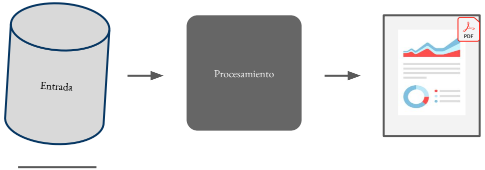
Pipeline, entradas, procesamiento y salidas



Pipeline, entradas, procesamiento y salidas

Pipeline, entradas, procesamiento y salidas

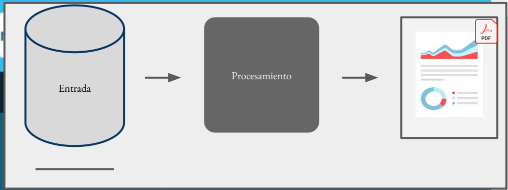




Información de entrada

Argentina.gob.ar

Buscar trámites, servicios o áreas



Dirección de Estadísticas e Información de la Salud

Producimos, difundimos y analizamos estadísticas relacionadas con condiciones de vida y problemas de salud.

Indicadores básicos

Publicaciones

Datos

Guías COVID-19

Enfermedad por COVID-19: Guía para la certificación médica de las causas de muerte

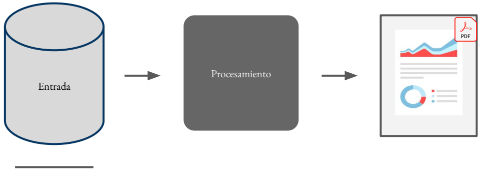
Descargar

Enfermedad por COVID-19: Guía para la codificación de las causas de muerte

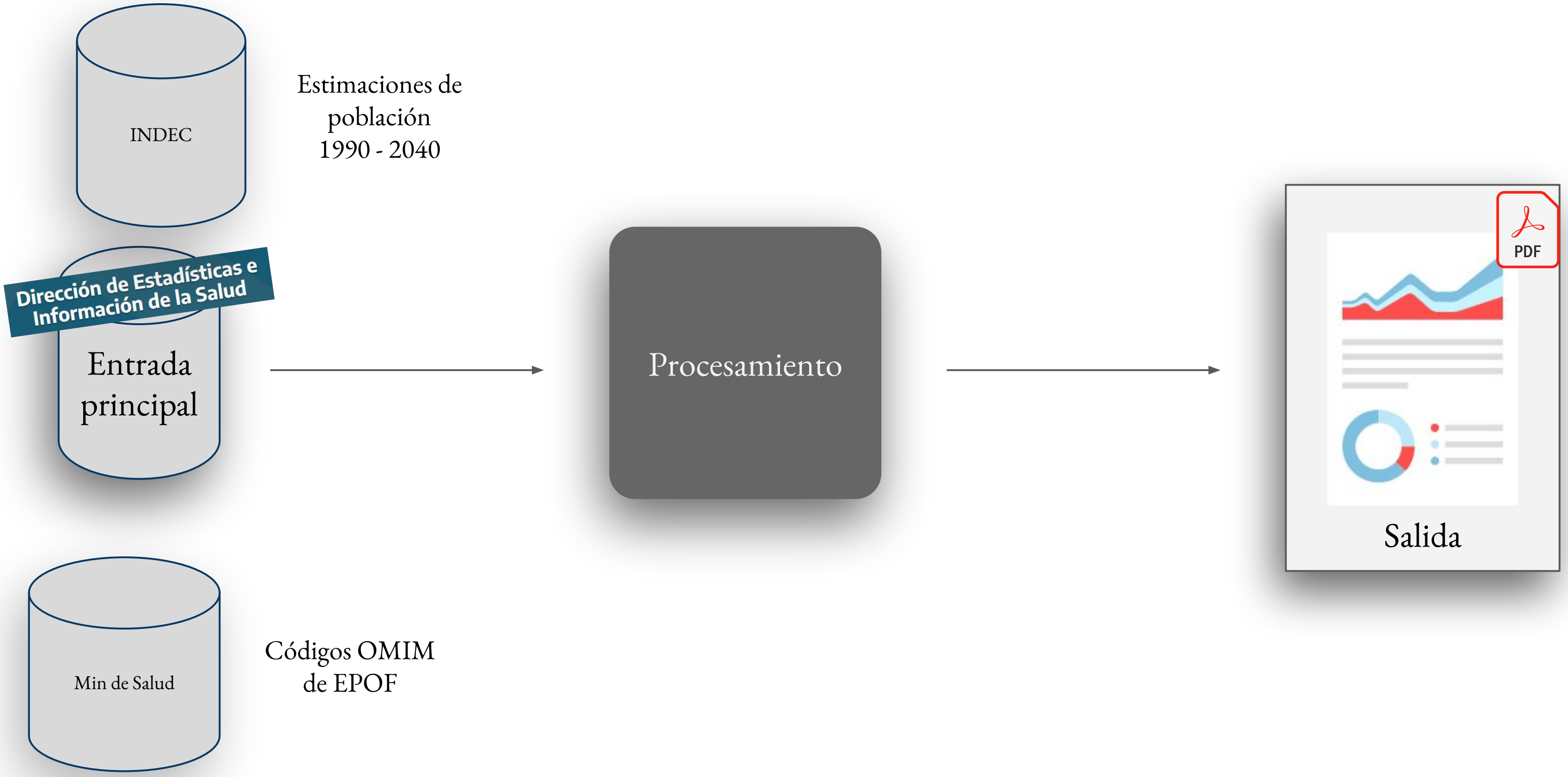
Descargar

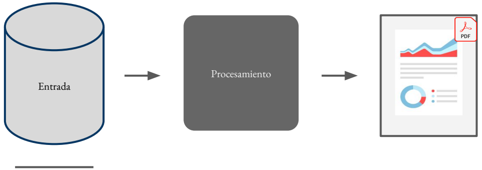
¿Cómo podemos ayudarte?

Información de entrada

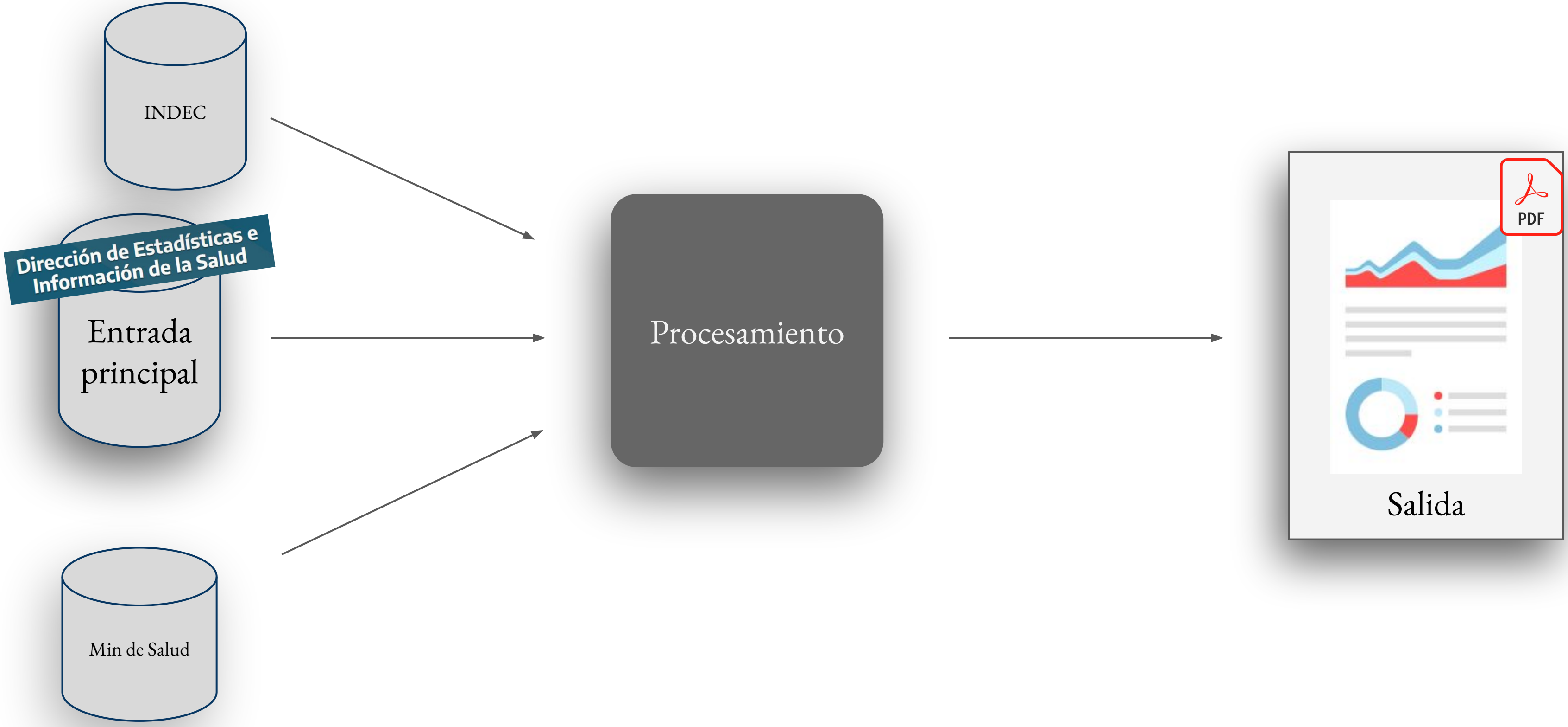


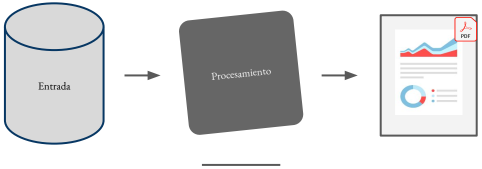
Información de entrada



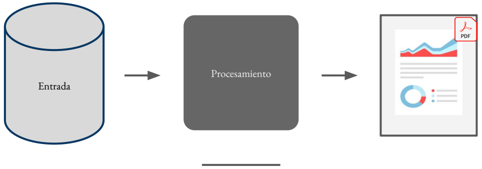


Información de entrada





Procesamiento

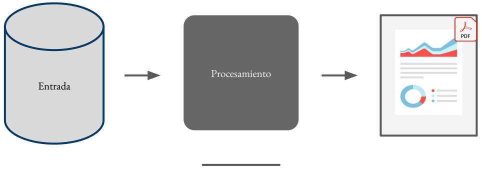


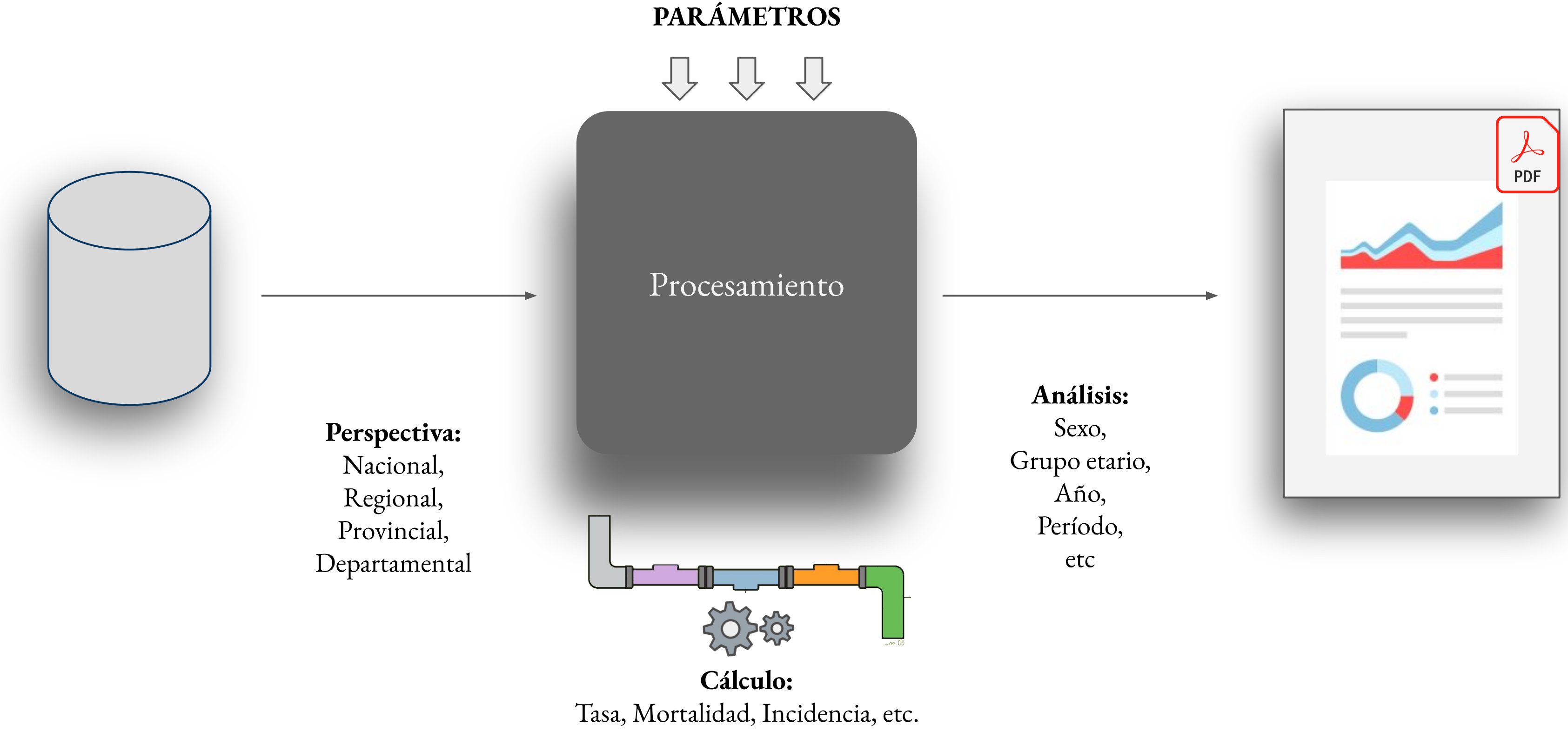
Procesamiento

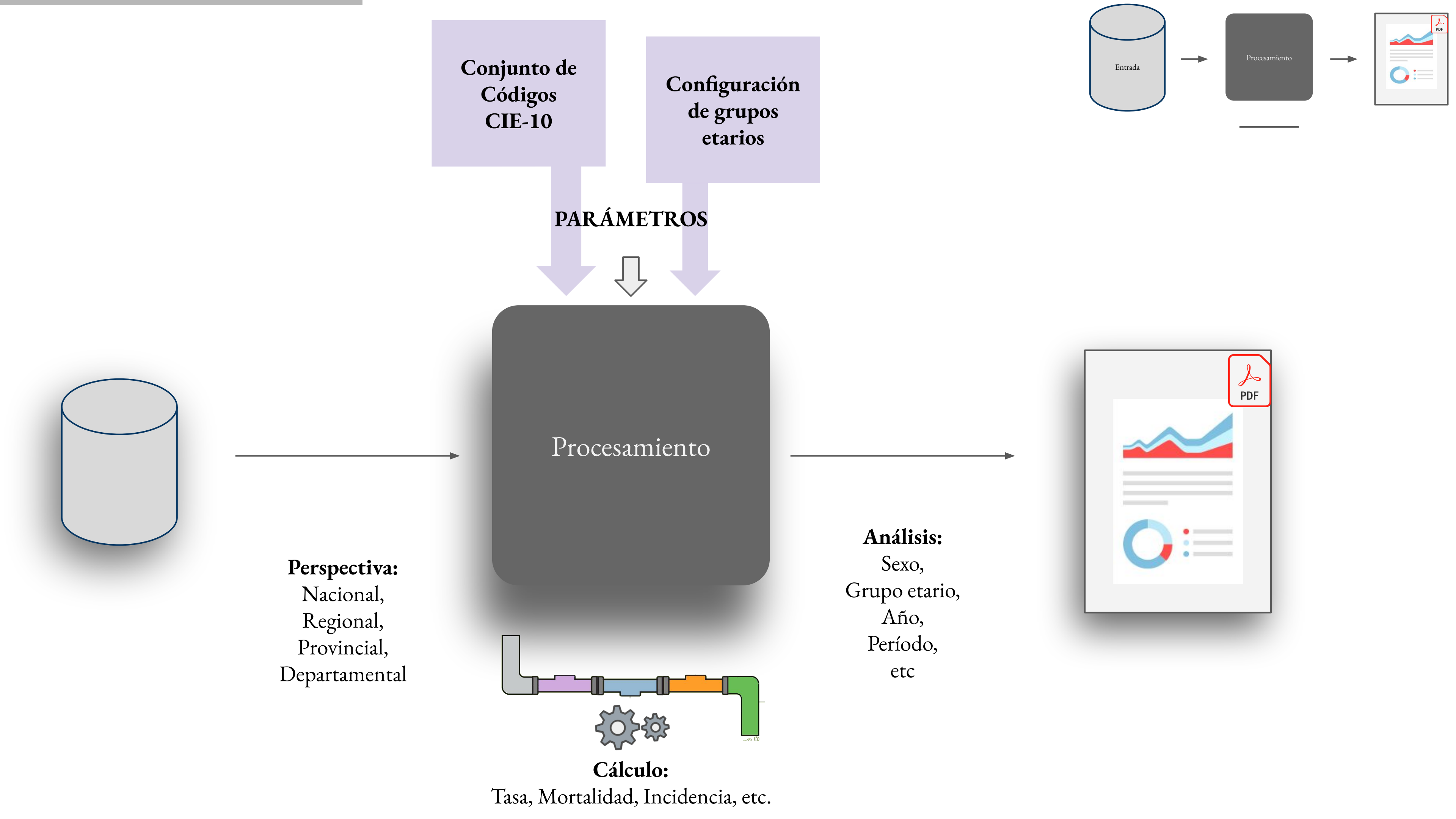
Tasa de Mortalidad por Causas Específicas (CSMR): Número de fallecimientos por una causa específica / Muertes totales en una población específica, en un tiempo específico.

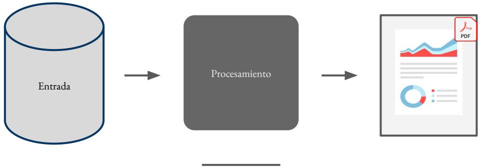
CSMR detalladas: Agrupando, filtrando por grupo etario, sexo, categoría de código de deceso.

Prevalencia: Número de fallecimientos por una causa específica / Población específica, en un punto específico del tiempo.


Entrada
Procesamiento
PDF


PARÁMETROS
Procesamiento
PDF
Perspectiva:
Nacional,
Regional,
Provincial,
Departamental
Análisis:
Sexo,
Grupo etario,
Año,
Período,
etc
Cálculo:
Tasa, Mortalidad, Incidencia, etc.

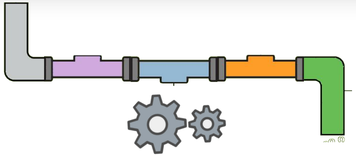

Entrada
Procesamiento
PDF
Conjunto de Códigos CIE-10
Configuración de grupos etarios
PARÁMETROS
Procesamiento
PDF
Perspectiva:
Nacional,
Regional,
Provincial,
Departamental
Análisis:
Sexo,
Grupo etario,
Año,
Período,
etc
Cálculo:
Tasa, Mortalidad, Incidencia, etc.

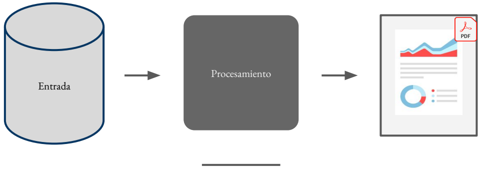


Procesamiento

```
df.head()
```

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN





	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

```
df_regional.head()
```

	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Region A	1997	4114
4	Region A	1998	4146

```
df_provincial.tail()
```

	provincia_id	año	fallecimientos
667	10	2015	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	233

```
df_departamental.head()
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	88456	1998	6

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

```
df_regional.head()
```

	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Regi[illegible]	[illegible]	[illegible]
4	Regi[illegible]	[illegible]	[illegible]

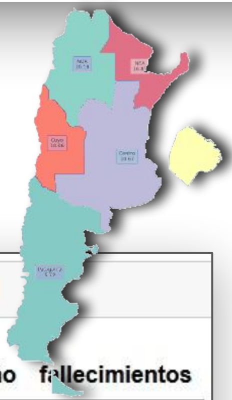
```
df_provincial.tail()
```

	provincia_id	año	fallecimientos
667	10	2015	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

```
df_departamental.head()
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



```
df_regional.head()
```

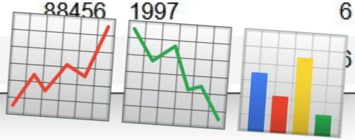
	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Regi[illegible]	[illegible]	[illegible]
4	Regi[illegible]	[illegible]	[illegible]

```
df_provincial.tail()
```

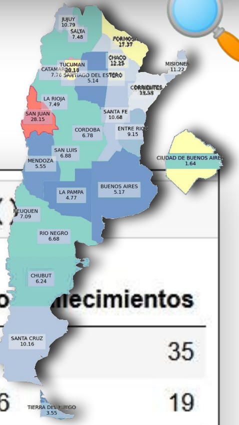
	provincia_id	año	fallecimientos
667	10	2015	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

```
df_departamental.head()
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]



	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



```
df_regional.head()
```

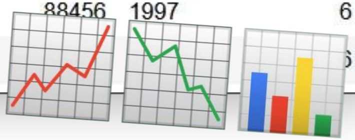
	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Regi	[illegible]	[illegible]
4	Regi	[illegible]	[illegible]

```
df_provincial.tail()
```

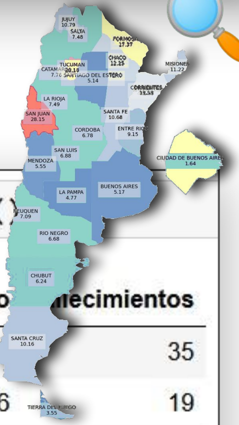
	provincia_id	año	fallecimientos
667	10	201	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

```
df_departamental.head()
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]



	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN




```
df_regional.head()
```

	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Regi[illegible]	[illegible]	[illegible]
4	Regi[illegible]	[illegible]	[illegible]

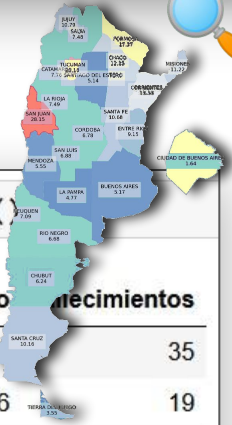
```
df_provincial.tail()
```

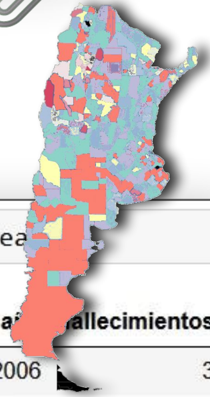
	provincia_id	año	fallecimientos
667	10	201[illegible]	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

```
df_departamental.head()
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN





```
df_regional.head()
```

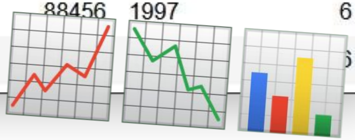
	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	[illegible]	[illegible]	[illegible]
4	[illegible]	[illegible]	[illegible]

```
df_provincial.tail()
```

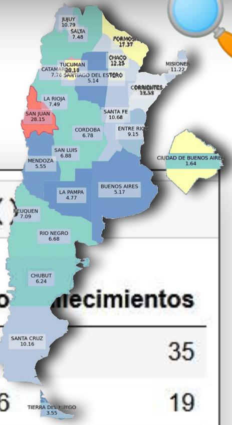
	provincia_id	año	[illegible]
667	10	[illegible]	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

```
df_departamental.hea[illegible]
```

	departamento_id	[illegible]	[illegible]
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]

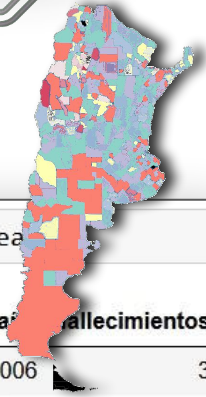


	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



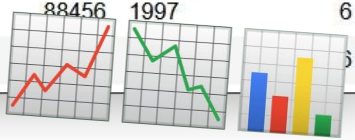
```
df_provincial.tail()
```

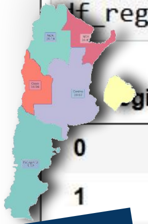
	provincia_id	año	fallecimientos
667	10	201[illegible]	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]



```
df_departamental.hea
```

	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	[illegible]	[illegible]	[illegible]



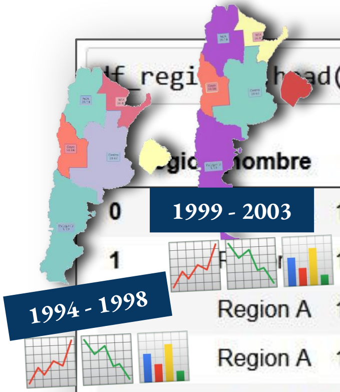


```
df_regional.head()
```

	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
	Region A	1996	4425
	Region A	1997	4114
4	Region A	1998	4146

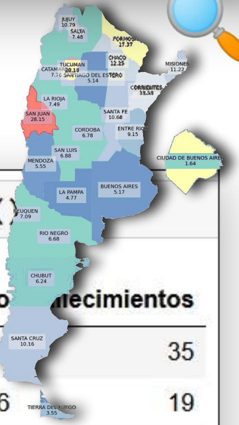
1994 - 1998

	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



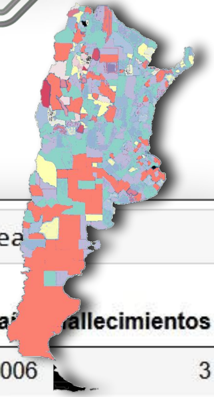

```
df_regi[illegible]head()
```

	[illegible]ombre	año	fallecimientos
0		1994	4603
1		1995	4365
	Region A	1996	4425
	Region A	1997	4114
4	Region A	1998	4146



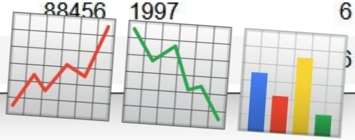
```
df_provincial.tail()
```

	provincia_id	año	[illegible]lecimientos
667	10	201[illegible]	35
668	10	2016	19
669	10	2017	21
670	10	2018	226
671	10	2019	[illegible]

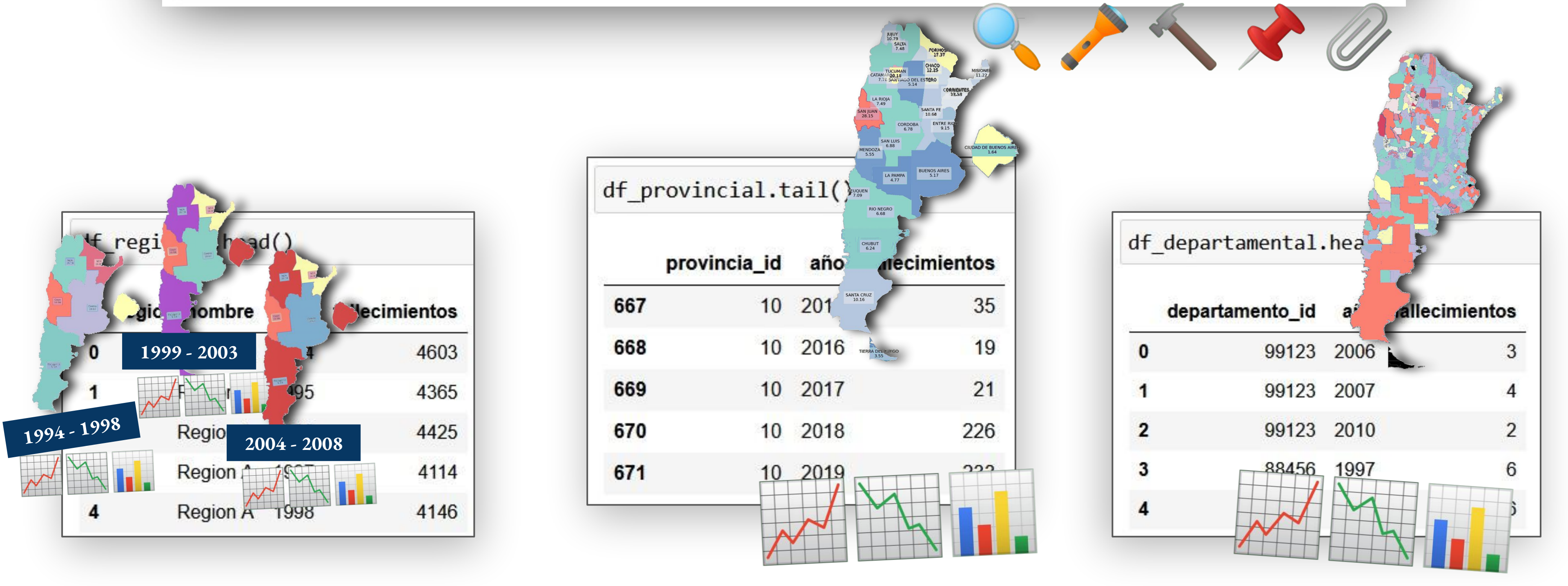


```
df_departamental.hea[illegible]
```

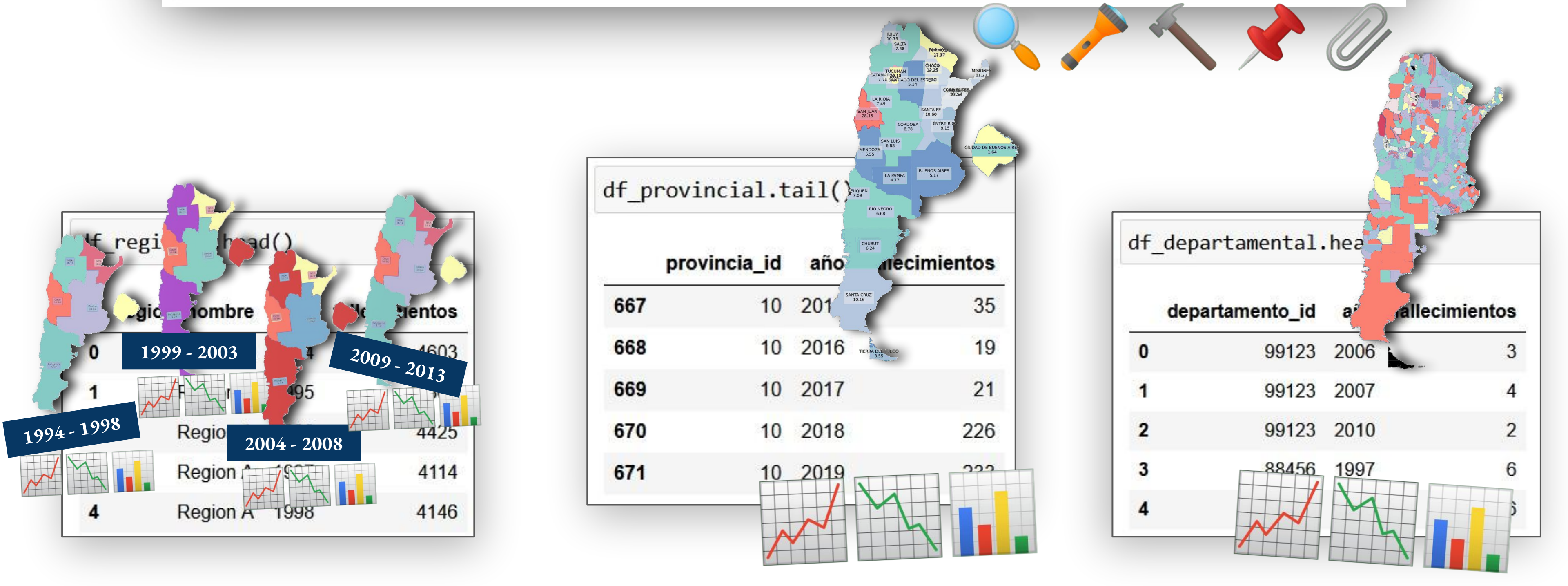
	departamento_id	a[illegible]	[illegible]allecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4			[illegible]



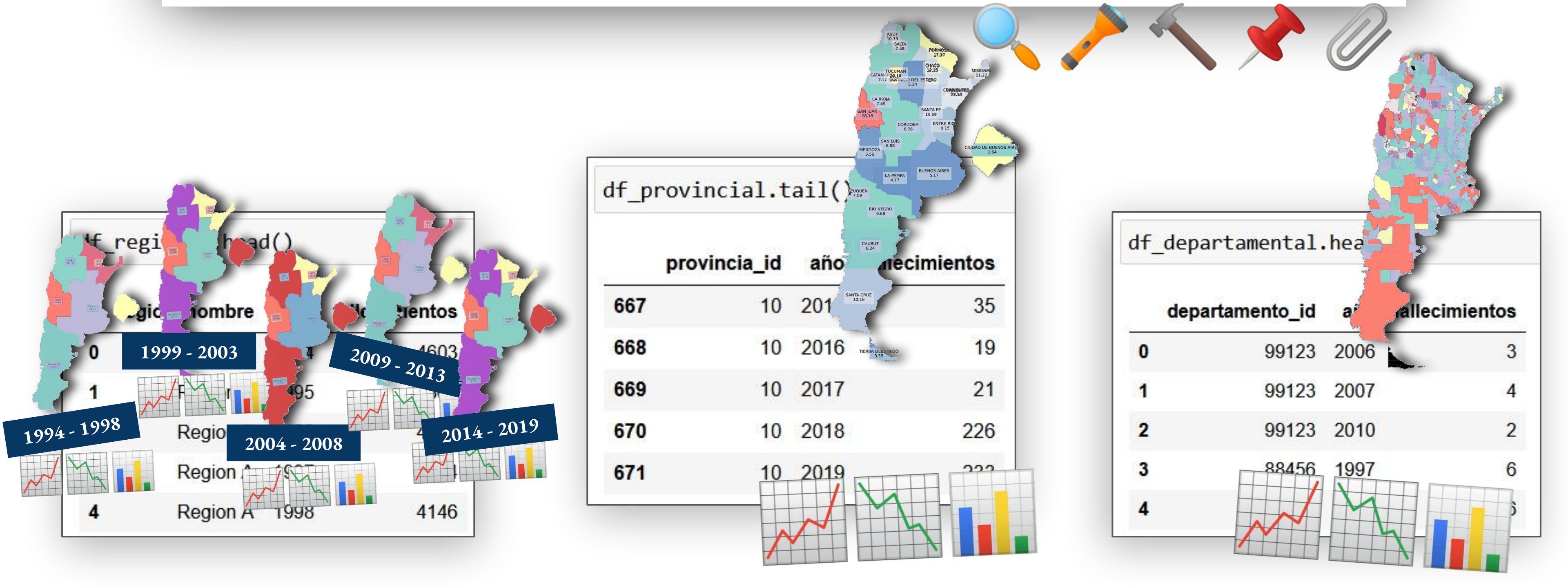
	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



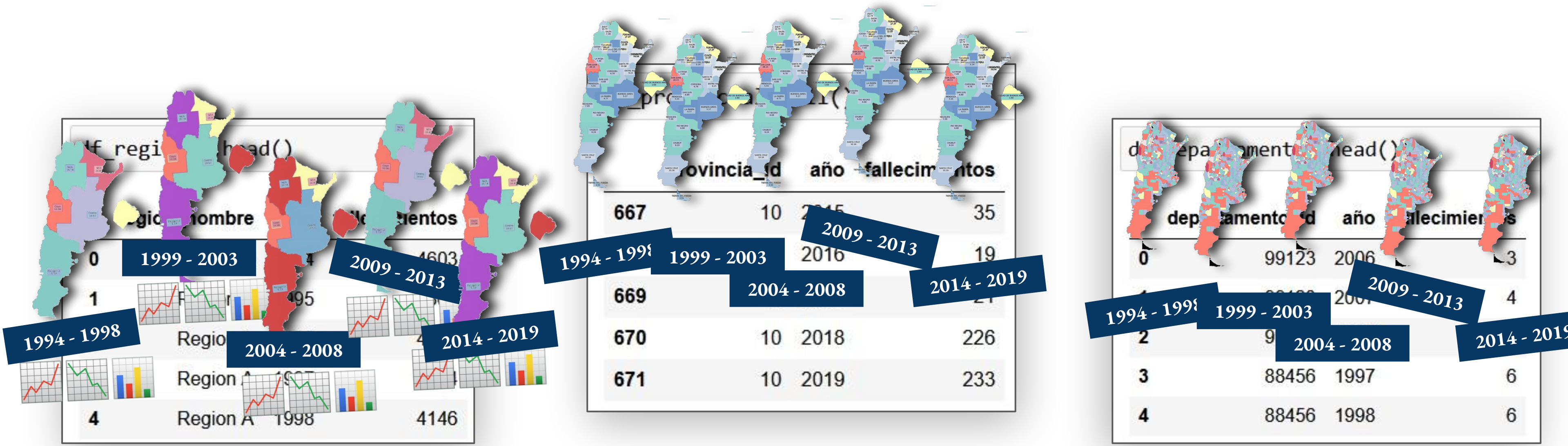
	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

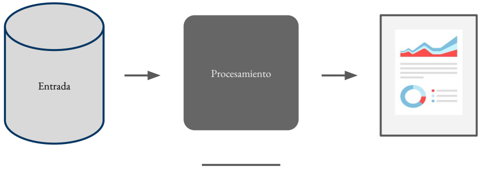


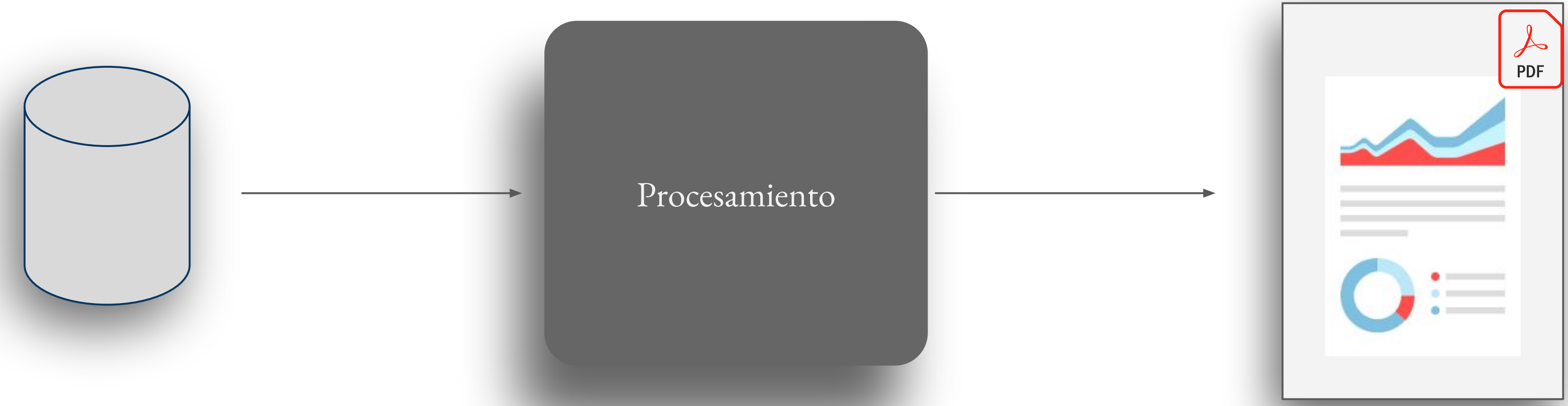
	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

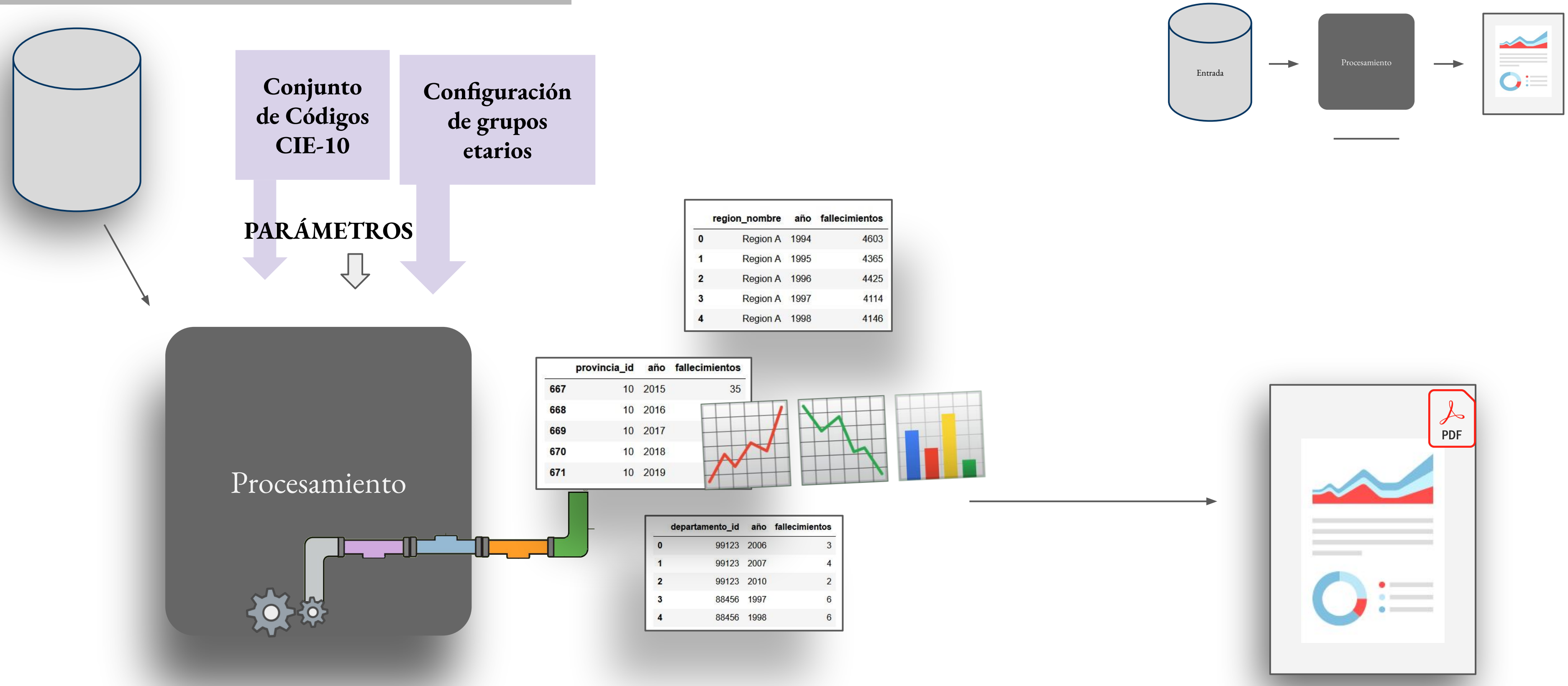


	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN




Entrada
Procesamiento

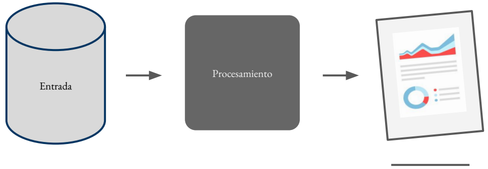

Procesamiento
PDF



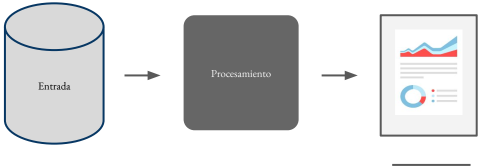
	region_nombre	año	fallecimientos
0	Region A	1994	4603
1	Region A	1995	4365
2	Region A	1996	4425
3	Region A	1997	4114
4	Region A	1998	4146

	provincia_id	año	fallecimientos
667	10	2015	35
668	10	2016	
669	10	2017	
670	10	2018	
671	10	2019	

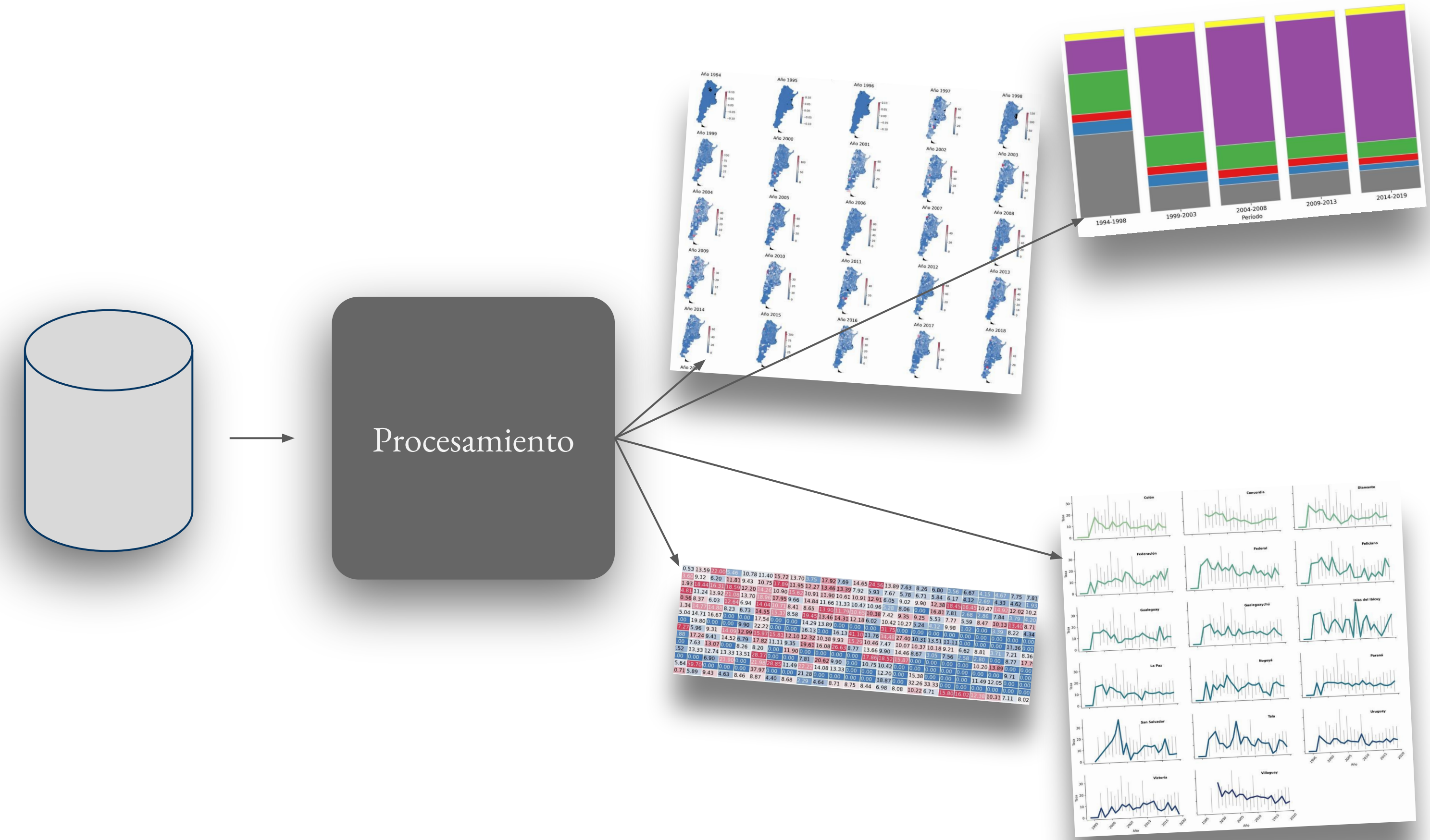
	departamento_id	año	fallecimientos
0	99123	2006	3
1	99123	2007	4
2	99123	2010	2
3	88456	1997	6
4	88456	1998	6

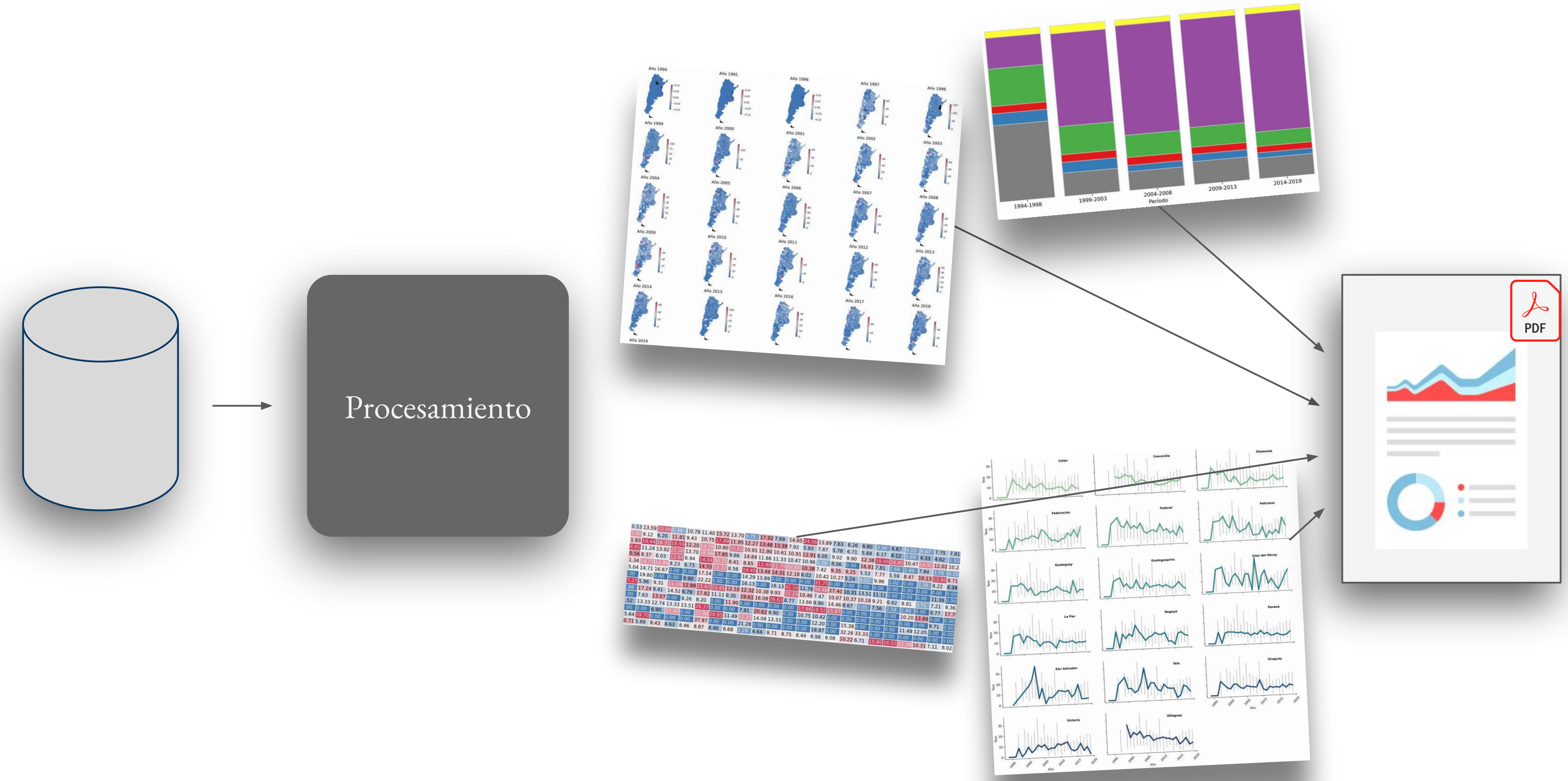


Salida


Entrada
Procesamiento

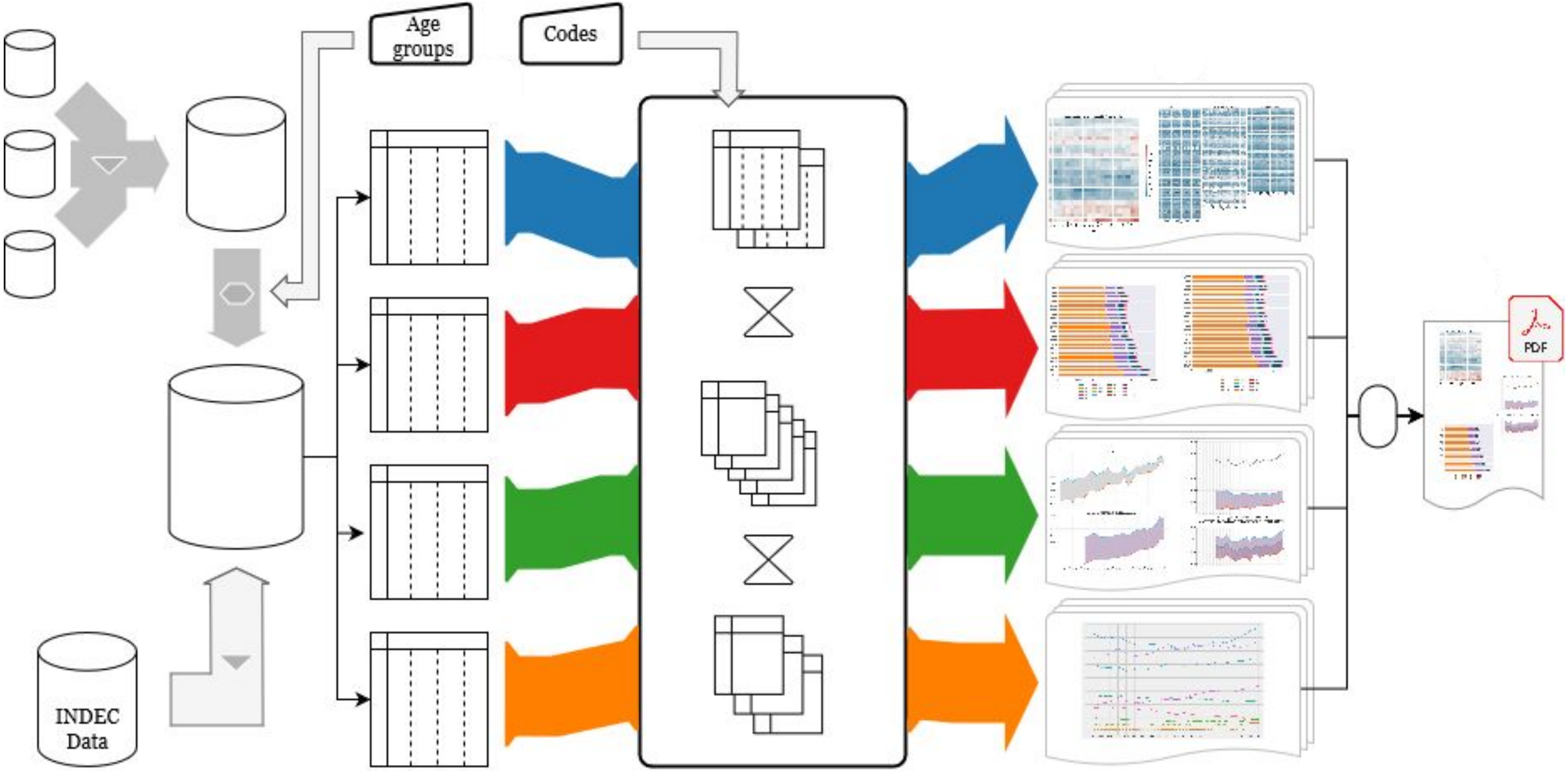

Procesamiento
PDF


Procesamiento
1994-1998
1999-2003
2004-2008
Período
2009-2013
2014-2019


Procesamiento
1994-1998
1999-2003
2004-2008
Período
2009-2013
2014-2019
PDF

Pipeline


Entrada
Procesami
ento


Age groups
Codes
INDEC Data
PDF

Implementación

jupyter

File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Python 3 (ipykernel)

Run   Markdown   nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:
```
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:
```
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:
```
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:
```
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:
```
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:

```python
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:

```python
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:

```python
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:

```python
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:

```python
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:

```python
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:

```python
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:

```python
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

Dataset general - Parte I

In [ ]:

```python
#departments_shape[['provincia_id', 'provincia_nombre', 'region_name']].drop_duplicates(),
```

In [241]:

```python
birth_dataset = (
    pandas.merge(
        birth_data_departments,
        departments_shape.drop(columns='geometry'),
        on='departamento_id',
        how='left'
    )
)

print(birth_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15027 entries, 0 to 15026
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   departamento_id    15027 non-null  object
 1   año                15027 non-null  int64 
 2   nacimientos        15027 non-null  int64 
 3   departamento_nombre 13201 non-null  object
 4   provincia_id       13201 non-null  object
 5   provincia_nombre   13201 non-null  object
 6   region_indec       13201 non-null  object
dtypes: int64(2), object(5)
memory usage: 939.2+ KB
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```python
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```python
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```python
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:
```python
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:
```python
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:
```python
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:
```python
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:
```python
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

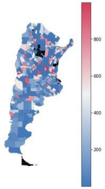
Dataset general - Parte I

In [ ]:
```python
#departments_shape[['provincia_id', 'provincia_nombre', 'region_name']].drop_duplicates(),
```

In [241]:
```python
birth_dataset = (
    pandas.merge(
        birth_data_departments,
        departments_shape.drop(columns='geometry'),
        on='departamento_id',
        how='left'
    )
)

print(birth_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15027 entries, 0 to 15026
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   departamento_id    15027 non-null  object
 1   año                15027 non-null  int64
 2   nacimientos        15027 non-null  int64
 3   departamento_nombre  13201 non-null  object
 4   provincia_id       13201 non-null  object
 5   provincia_nombre   13201 non-null  object
 6   region_indec       13201 non-null  object
dtypes: int64(2), object(5)
memory usage: 939.2+ KB
```

In [251]:
```python
f, ax = plt.subplots(figsize=(4, 8))

departments_shape.plot(
    ax=ax,
    color='black'
)

cmap = seaborn.diverging_palette(250, 5, as_cmap=True)
pandas.merge(
    departments_shape,
    birth_dataset[birth_dataset['año'] == 1994],
    on='departamento_id'
).plot(
    ax=ax,
    column='nacimientos',
    cmap=cmap,
    edgecolor='none',
    linewidth=0.02,
    legend=True
)
ax.axis('off')
plt.show();
```



File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Python 3 (ipykernel)

Run   Markdown   nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:
```
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:
```
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:
```
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:
```
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:
```
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

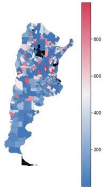
Dataset general - Parte I

In [ ]:
```
#departments_shape[['provincia_id', 'provincia_nombre', 'region_name']].drop_duplicates(),
```

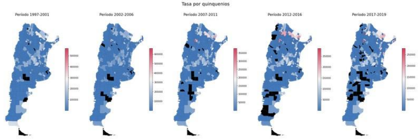
In [241]:
```
birth_dataset = (
    pandas.merge(
        birth_data_departments,
        departments_shape.drop(columns='geometry'),
        on='departamento_id',
        how='left'
    )
)

print(birth_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15027 entries, 0 to 15026
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   departamento_id    15027 non-null  object
 1   año                15027 non-null  int64 
 2   nacimientos        15027 non-null  int64 
 3   departamento_nombre 13201 non-null  object
 4   provincia_id       13201 non-null  object
 5   provincia_nombre   13201 non-null  object
 6   region_indec       13201 non-null  object
dtypes: int64(2), object(5)
memory usage: 939.2+ KB
```

In [251]:
```
f, ax = plt.subplots(figsize=(4, 8))

departments_shape.plot(
    ax=ax,
    color='black'
)

cmap = seaborn.diverging_palette(250, 5, as_cmap=True)
pandas.merge(
    departments_shape,
    birth_dataset[birth_dataset['año'] == 1994],
    on='departamento_id'
).plot(
    ax=ax,
    column='nacimientos',
    cmap=cmap,
    edgecolor='none',
    linewidth=0.02,
    legend=True
)
ax.axis('off')
plt.show();
```



In [344]:
```
f, ax = plt.subplots(nrows=1, ncols=5, figsize=(26, 8), constrained_layout=True)
axs = ax.flatten()

for i, period in enumerate(quinq_data):
    # == Map: ==
    ax = axs[i]

    departments_shape.plot(
        ax=ax,
        color='black'
    )

    pandas.merge(
        departments_shape,
        quinq_data[period],
        on='departamento_id'
    ).plot(
        ax=ax,
        column='tasa',
        cmap=cmap,
        edgecolor='none',
        linewidth=0.02,
        legend=True,
        legend_kwds={'shrink': 0.5},
    )
    ax.axis('off')
    ax.set_title(f"Período {period}", fontsize=16)

f.suptitle("Tasa por quinquenios", fontsize=20, y=1.08)
plt.show();
```



In [340]:
```
TASA_MAX = numpy.max([quinq_data[key]['tasa'].max() for key in quinq_data])
TASA_MIN = numpy.min([quinq_data[key]['tasa'].min() for key in quinq_data])

print(TASA_MAX, TASA_MIN)
```

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Markdown | nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:
```
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:
```
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:
```
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:
```
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:
```
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

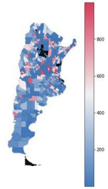
Dataset general - Parte I

In [ ]:
```
#departments_shape[['provincia_id', 'provincia_nombre', 'region_name']].drop_duplicates(),
```

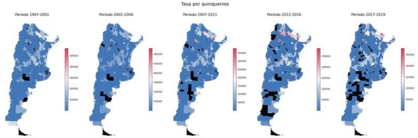
In [241]:
```
birth_dataset = (
    pandas.merge(
        birth_data_departments,
        departments_shape.drop(columns='geometry'),
        on='departamento_id',
        how='left'
    )
)

print(birth_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15027 entries, 0 to 15026
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   departamento_id    15027 non-null  object
 1   año                15027 non-null  int64 
 2   nacimientos        15027 non-null  int64 
 3   departamento_nombre  13201 non-null  object
 4   provincia_id       13201 non-null  object
 5   provincia_nombre   13201 non-null  object
 6   region_indec       13201 non-null  object
dtypes: int64(2), object(5)
memory usage: 939.2+ KB
```

In [251]:
```
f, ax = plt.subplots(figsize=(4, 8))

departments_shape.plot(
    ax=ax,
    color='black'
)

cmap = seaborn.diverging_palette(250, 5, as_cmap=True)
pandas.merge(
    departments_shape,
    birth_dataset[birth_dataset['año'] == 1994],
    on='departamento_id'
).plot(
    ax=ax,
    column='nacimientos',
    cmap=cmap,
    edgecolor='none',
    linewidth=0.02,
    legend=True
)
ax.axis('off')
plt.show();
```

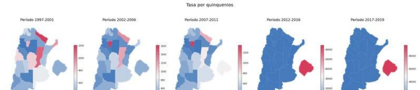


In [344]:
```
f, ax = plt.subplots(nrows=1, ncols=5, figsize=(26, 8), constrained_layout=True)
axs = ax.flatten()

for i, period in enumerate(quinq_data):
    # == Map: ==
    ax = axs[i]

    departments_shape.plot(
        ax=ax,
        color='black'
    )

    pandas.merge(
        departments_shape,
        quinq_data[period],
        on='departamento_id'
    ).plot(
        ax=ax,
        column='tasa',
        cmap=cmap,
        edgecolor='none',
        linewidth=0.02,
        legend=True,
        legend_kwds={'shrink': 0.5},
    )
    ax.axis('off')
    ax.set_title(f"Período {period}", fontsize=16)

f.suptitle("Tasa por quinquenios", fontsize=20, y=1.08)
plt.show();
```



In [340]:
```
TASA_MAX = numpy.max([quinq_data[key]['tasa'].max() for key in quinq_data])
TASA_MIN = numpy.min([quinq_data[key]['tasa'].min() for key in quinq_data])

print(TASA_MAX, TASA_MIN)
```

In [461]:
```
f, ax = plt.subplots(nrows=1, ncols=5, figsize=(26, 8), constrained_layout=True)
axs = ax.flatten()

for i, period in enumerate(provincial_quinq_data):
    # == Map: ==
    ax = axs[i]

    pandas.merge(
        provinces_shp,
        provincial_quinq_data[period],
        on='provincia_id'
    ).plot(
        ax=ax,
        column='tasa',
        cmap=cmap,
        edgecolor='none',
        linewidth=0.02,
        legend=True,
        legend_kwds={'shrink': 0.5},
    )

    ax.axis('off')
    ax.set_title(f"Período {period}", fontsize=16)

f.suptitle("Tasa por quinquenios", fontsize=20, y=1.08)
plt.show();
```



File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown nbdiff

Objetivos:

1) Calcular las tasas quinquenales a nivel regional 2) Generar 5 mapas (1 por cada quinquenio) 3) Dejar variable para poder cambiar las causas

Es necesario entonces trabajr con dos bases de datos:

- Nacidos vivos (<**años**>)
- Muertes fetales (<**años**>)

Generar un dataset general con granularidad id departamento, id de provincia, nombre de region + nacimientos + muertes fetales.

Al dataset anterior se debe agregar una columna TASA, que se calcula según la siguiente fórmula:

- (muerte fetal / nacidos vivos) * 10000

In [127]:
```
import pandas
from pprint import pprint
import geopandas
import matplotlib.pyplot as plt
import seaborn
import numpy
```

In [480]:
```
IMG_PREFIX = 'YP1'
CSV_PREFIX = 'YP1'
```

Capa

Capa del mapa de Argentina. Util para obtener la información de departamento, provincia y region estandarizada.

In [168]:
```
from functools import partial

departments_shape = geopandas.read_file(
    '/home/lmorales/work/docbook/shapes/departamentos-argentina.geojson')

regiones = {
    'Cuyo':[
        'Mendoza',
        'San Juan',
        'San Luis',
    ],
    'NOA': [
        'Salta',
        'Jujuy',
        'Santiago del Estero',
        'Tucumán',
        'La Rioja',
        'Catamarca',
    ],
    'NEA':[
        'Chaco',
        'Corrientes',
        'Formosa',
        'Misiones',
    ],
```

In [150]:
```
datos_nacidos_vivos = pandas.read_csv(
    '../datasets/Copia para Leo_de Nacidos vivos 1994-2019.xlsx - Nacidos vivos 2005-2018.csv',
    dtype={'PROVRES': object, 'DEPRES': object,}# 'CUENTA': int, 'AÑO': int}
)
```

In [151]:
```
datos_nacidos_vivos.head()
```

Out[151]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1.380
1	1994	2	2	1.321
2	1994	2	3	881.000
3	1994	2	4	933.000
4	1994	2	5	1.951

In [152]:
```
datos_nacidos_vivos['CUENTA'] = datos_nacidos_vivos['CUENTA'].astype('int')
```

In [153]:
```
datos_nacidos_vivos.head()
```

Out[153]:

	AÑO	PROVRES	DEPRES	CUENTA
0	1994	2	1	1
1	1994	2	2	1
2	1994	2	3	881
3	1994	2	4	933
4	1994	2	5	1

In [154]:
```
def clean_department_code(code):
    pad = 3 - len(str(code))
    return '0'*pad + str(code)

# funcion para limpiar el código de provincia
def clean_province_code(code):
    return '0'+str(code) if (len(str(code)) == 1) else str(code)


print(4, clean_department_code(4))
print(41, clean_department_code(41))
print(411, clean_department_code(411))
```

```
4 004
41 041
411 411
```

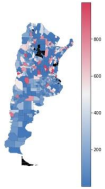
Dataset general - Parte I

In [ ]:
```
#departments_shape[['provincia_id', 'provincia_nombre', 'region_name']].drop_duplicates(),
```

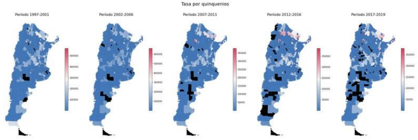
In [241]:
```
birth_dataset = (
    pandas.merge(
        birth_data_departments,
        departments_shape.drop(columns='geometry'),
        on='departamento_id',
        how='left'
    )
)

print(birth_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15027 entries, 0 to 15026
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   departamento_id     15027 non-null  object
 1   año                 15027 non-null  int64
 2   nacimientos         15027 non-null  int64
 3   departamento_nombre 13201 non-null  object
 4   provincia_id        13201 non-null  object
 5   provincia_nombre    13201 non-null  object
 6   region_indec        13201 non-null  object
dtypes: int64(2), object(5)
memory usage: 939.2+ KB
```

In [291]:
```
f, ax = plt.subplots(figsize=(4, 8))

departments_shape.plot(
    ax=ax,
    color='black'
)

cmap = seaborn.diverging_palette(250, 5, as_cmap=True)
pandas.merge(
    departments_shape,
    birth_dataset[birth_dataset['año'] == 1994],
    on='departamento_id'
).plot(
    ax=ax,
    column='nacimientos',
    cmap=cmap,
    edgecolor='none',
    linewidth=0.02,
    legend=True
)
ax.axis('off')
plt.show();
```

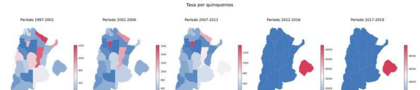


In [344]:
```
f, ax = plt.subplots(nrows=1, ncols=5, figsize=(26, 8), constrained_layout=True)
axs = ax.flatten()

for i, period in enumerate(quinq_data):
    # == Map: ==
    ax = axs[i]

    departments_shape.plot(
        ax=ax,
        color='black'
    )

    pandas.merge(
        departments_shape,
        quinq_data[period],
        on='departamento_id'
    ).plot(
        ax=ax,
        column='tasa',
        cmap=cmap,
        edgecolor='none',
        linewidth=0.02,
        legend=True,
        legend_kwds={'shrink': 0.5},
    )
    ax.axis('off')
    ax.set_title(f"Período {period}", fontsize=16)

f.suptitle("Tasa por quinquenios", fontsize=20, y=1.08)
plt.show();
```



In [340]:
```
TASA_MAX = numpy.max([quinq_data[key]['tasa'].max() for key in quinq_data])
TASA_MIN = numpy.min([quinq_data[key]['tasa'].min() for key in quinq_data])

print(TASA_MAX, TASA_MIN)
```

In [461]:
```
f, ax = plt.subplots(nrows=1, ncols=5, figsize=(26, 8), constrained_layout=True)
axs = ax.flatten()

for i, period in enumerate(provincial_quinq_data):
    # == Map: ==
    ax = axs[i]

    pandas.merge(
        provinces_shp,
        provincial_quinq_data[period],
        on='provincia_id'
    ).plot(
        ax=ax,
        column='tasa',
        cmap=cmap,
        edgecolor='none',
        linewidth=0.02,
        legend=True,
        legend_kwds={'shrink': 0.5},
    )

    ax.axis('off')
    ax.set_title(f"Período {period}", fontsize=16)

f.suptitle("Tasa por quinquenios", fontsize=20, y=1.08)
plt.show();
```

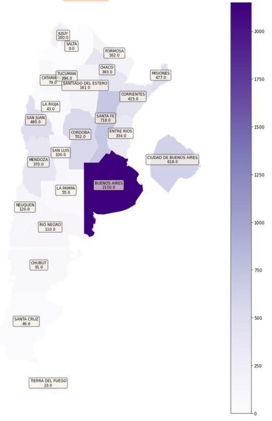


In [157]:
```
deceases_data_1994.head()
```

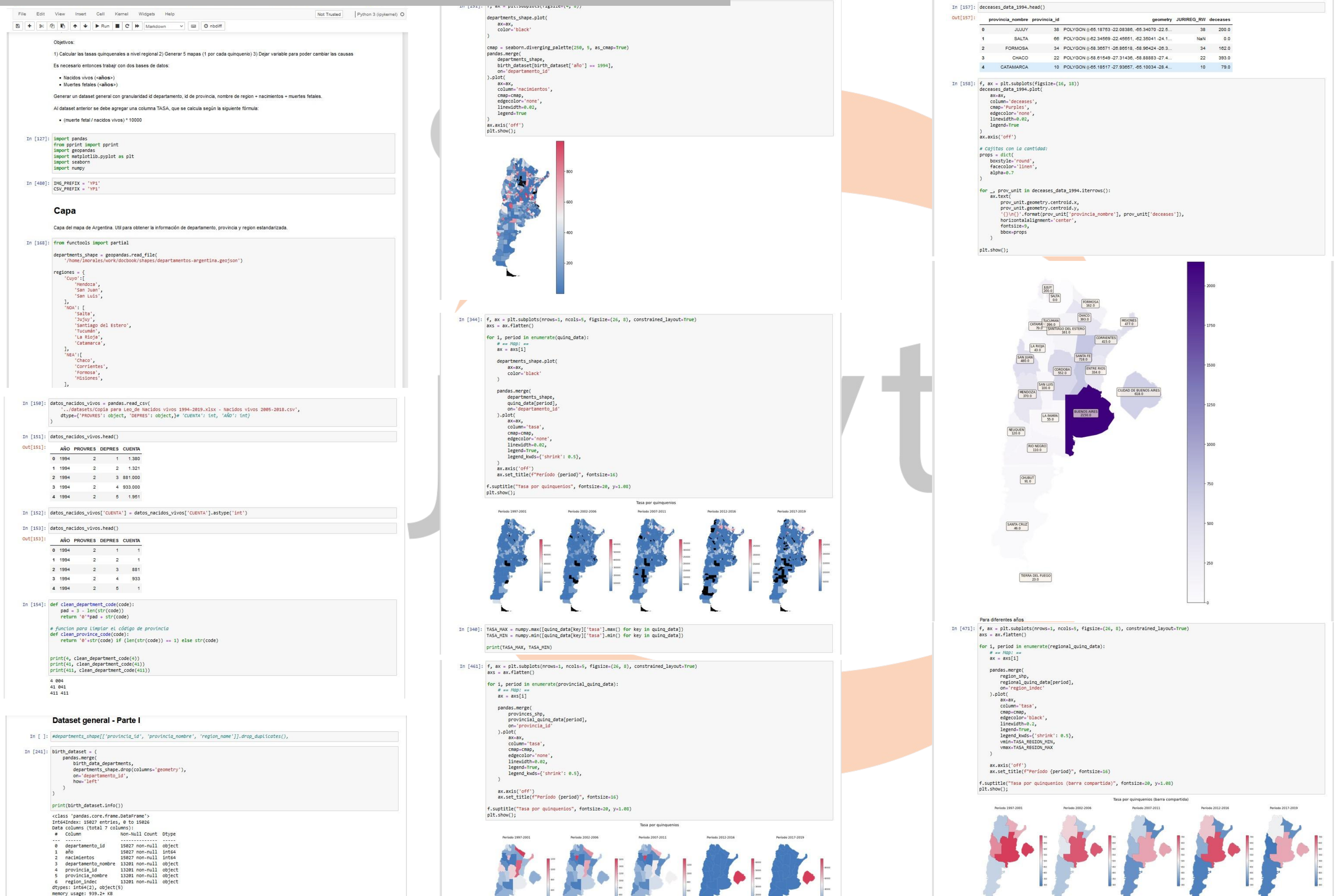
Out[157]:

	provincia_nombre	provincia_id	geometry	JURIREG_RW	deceases
0	JUJUY	38	POLYGON ((-65.18753 -22.08386, -65.34070 -22.5...	38	200.0
1	SALTA	66	POLYGON ((-62.34569 -22.45651, -62.35041 -24.1...	NaN	0.0
2	FORMOSA	34	POLYGON ((-58.36571 -26.86518, -58.96424 -26.3...	34	162.0
3	CHACO	22	POLYGON ((-58.61549 -27.31436, -58.88883 -27.4...	22	393.0
4	CATAMARCA	10	POLYGON ((-65.18517 -27.93657, -65.10034 -28.4...	10	79.0

In [158]:
```
f, ax = plt.subplots(figsize=(16, 18))
deceases_data_1994.plot(
    ax=ax,
    column='deceases',
    cmap='Purples',
    edgecolor='none',
    linewidth=0.02,
    legend=True
)
ax.axis('off')

# Cajitas con la cantidad:
props = dict(
    boxstyle='round',
    facecolor='linen',
    alpha=0.7
)

for _, prov_unit in deceases_data_1994.iterrows():
    ax.text(
        prov_unit.geometry.centroid.x,
        prov_unit.geometry.centroid.y,
        '{}\n{}'.format(prov_unit['provincia_nombre'], prov_unit['deceases']),
        horizontalalignment='center',
        fontsize=9,
        bbox=props
    )

plt.show();
```



Para diferentes años



Notebooks de difícil mantenimiento

PLOOMBER

TIPS AND TRICKS

On writing clean Jupyter notebooks

10 recommendations for writing readable and maintainable notebooks

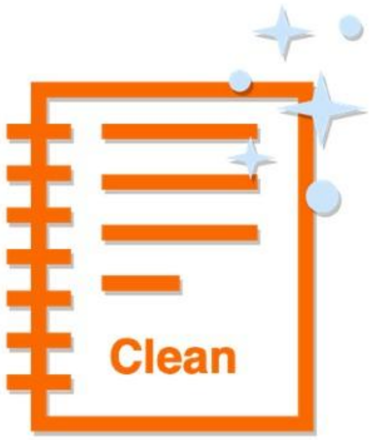


Image by author

Notebooks are a magnificent tool to explore data, but such a powerful tool can become hard to manage quickly. Ironically, the ability to interact with our data rapidly (modify code cells, run, and repeat) is the exact reason why a notebook may become an obscure entanglement of variables that are hard to understand, even to the notebook's author. But it doesn't have to be that way. This post summarizes my learnings over the past few years on writing clean notebooks.

TIPS AND TRICKS

On writing clean Jupyter notebooks

10 recommendations for writing readable and maintainable notebooks

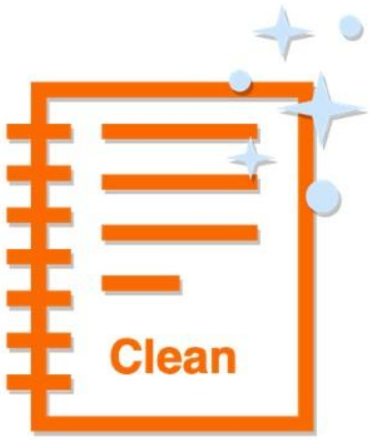
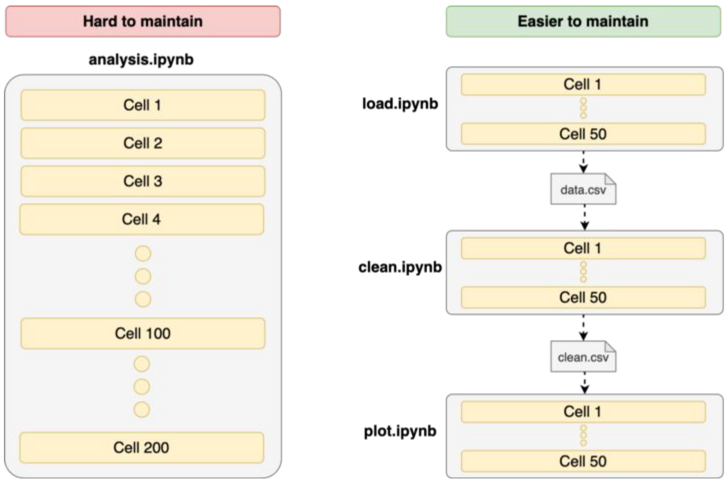


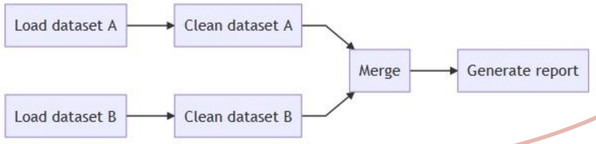
Image by author



Notebooks are a magnificent tool to explore data, but such a powerful tool can become hard to manage quickly. Ironically, the ability to interact with our data rapidly (modify code cells, run, and repeat) is the exact reason why a notebook may become an obscure entanglement of variables that are hard to understand, even to the notebook's author. But it doesn't have to be that way. This post summarizes my learnings over the past few years on writing clean notebooks.

Analytics

Ploomber is a fantastic tool for data manipulation and generating analytical reports.



▸ Example: BigQuery and Cloud Storage pipeline
▾ Example: Exploratory data analysis pipeline

Terminal (shell) Click to copy

```
$ pip install ploomber
$ ploomber examples -n templates/exploratory-analysis -o exploratory-analysis
```

Modularize your project

Instead of coding everything in a single notebook (which is difficult to maintain and collaborate), you can quickly break down your analysis into multiple parts.

Faster iterations

Finding data insights is an iterative process, with Ploomber's incremental builds you can rapidly iterate on your data since the framework skips redundant computations and only executes tasks whose source code has changed since the last execution.

Automated report generation

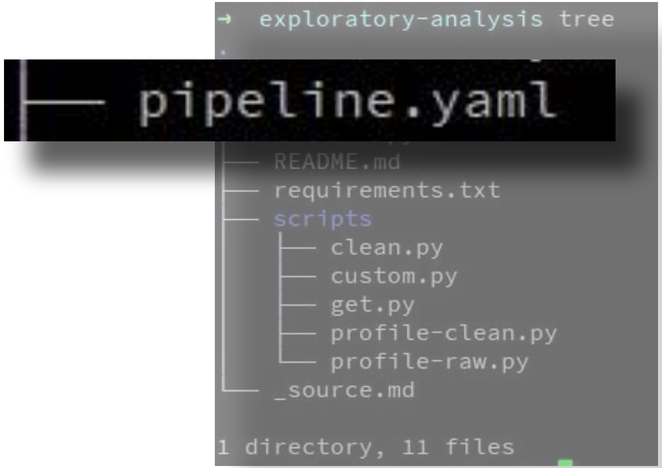
Once your pipeline is ready, you can easily create HTML reports from your scripts/notebooks. Just change the extension of the task, and Ploomber will automatically convert the output for you.

Text editor

```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```
➜  exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```yaml
tasks:
  # get raw data
  - source: scripts/get.py
    product:
      nb: products/get.html
      data: products/raw.csv

  # clean raw data
  - source: scripts/clean.py
    product:
      nb: products/clean.html
      # clean data
      data: products/clean.csv

  # quick clean data profiling
  - source: scripts/profile-clean.py
    # html report
    product: products/report-clean.html


  # custom plots
  - source: scripts/custom.py
    product: products/custom.html
```



```yaml
tasks:
  # get raw data
  - source: scripts/get.py
    product:
      nb: products/get.html
      data: products/raw.csv

  # clean raw data
  - source: scripts/clean.py
    product:
      nb: products/clean.html
      # clean data
      data: products/clean.csv

  # quick clean data profiling
  - source: scripts/profile-clean.py
    # html report
    product: products/report-clean.html


  # custom plots
  - source: scripts/custom.py
    product: products/custom.html
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# ---
# jupyter:
#   jupytext:
#     text_representation:
#       extension: .py
#       format_name: percent
#       format_version: '1.3'
#       jupytext_version: 1.13.6
#   kernelspec:
#     display_name: Python 3 (ipykernel)
#     language: python
#     name: python3
# ---

# %% [markdown]
# Get penguins data.


# %%
import seaborn as sns

# %% tags=["parameters"]
upstream = None
product = None


# %%
df = sns.load_dataset('penguins')

# %%
df.to_csv(product['data'], index=False)
```

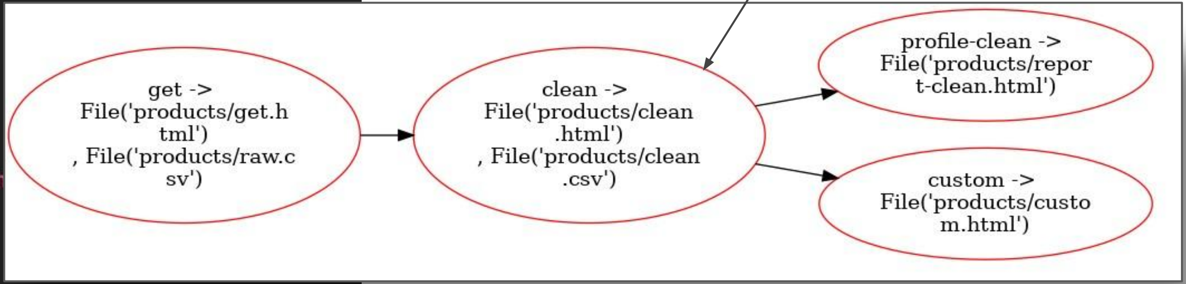
```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Clean penguins data.


# %%
import pandas as pd

# %% tags=["parameters"]
upstream = ['get']
product = None

# %%
df = pd.read_csv(upstream['get']['data'])

# %%
print(f'Raw data has {len(df)} rows...')

# %%
df.head()

# %%
# really simple cleaning, remove if the row has nas...
clean = df[df.isna().sum(axis=1) == 0]

# %%
print(f'Clean data has {len(clean)} rows...')

# %%
clean.to_csv(product['data'], index=False)
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Clean penguins data.


# %%
import pandas as pd

# %% tags=["parameters"]
upstream = ['get']
product = None

# %%
df = pd.read_csv(upstream['get']['data'])

# %%
print(f'Raw data has {len(df)} rows...')

# %%
df.head()

# %%
# really simple cleaning, remove if the row has n
clean = df[df.isna().sum(axis=1) == 0]

# %%
print(f'Clean data has {len(clean)} rows...')

# %%
clean.to_csv(product['data'], index=False)
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```



```python
# %% [markdown]
# Clean penguins data.


# %%
import pandas as pd

# %% tags=["parameters"]
upstream = ['get']
product = None

# %%
df = pd.read_csv(upstream['get']['data'])

# %%
print(f'Raw data has {len(df)} rows...')

# %%
df.head()

# %%
# really simple cleaning, remove if the row has n
clean = df[df.isna().sum(axis=1) == 0]

# %%
print(f'Clean data has {len(clean)} rows...')

# %%
clean.to_csv(product['data'], index=False)
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Quick clean data profiling.


# %%
import pandas as pd
from pandas_profiling import ProfileReport

# %% tags=["parameters"]
upstream = ['clean']
product = None


# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
ProfileReport(df, title="Clean Data Profiling Report")

# %%
```

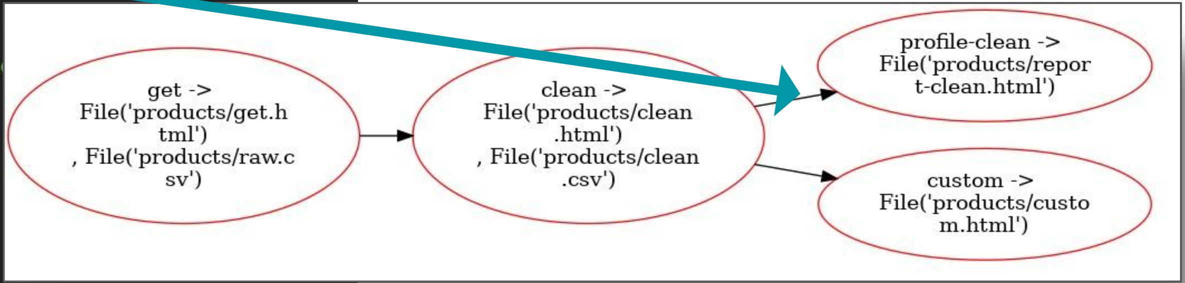
```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Quick clean data profiling.


# %%
import pandas as pd
from pandas_profiling import ProfileReport

# %% tags=["parameters"]
upstream = ['clean']
product = None


# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
ProfileReport(df, title="Clean Data Profiling Rep

# %%
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Quick clean data profiling.


# %%
import pandas as pd
from pandas_profiling import ProfileReport

# %% tags=["parameters"]
upstream = ['clean']
product = None


# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
ProfileReport(df, title="Clean Data Profiling Rep

# %%
```

```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```



```python
# %% [markdown]
# Create custom plots.

# %%
import seaborn as sns
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import matplotlib

# %%
# prettier plots
plt.style.use('ggplot')
# larger plots
matplotlib.rc('figure', figsize=(15, 10))
# larger fonts
sns.set_context('notebook', font_scale=1.5)

# %% tags=["parameters"]
upstream = ['clean']
product = None

# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
df.head()

# %%
_ = df['species'].value_counts().plot(kind='bar')

# %%
fig = px.histogram(df, x="bill_length_mm")
fig.show()

# %%
sns.pairplot(df, hue="species", height=3,diag_kind="hist")

# %%
```

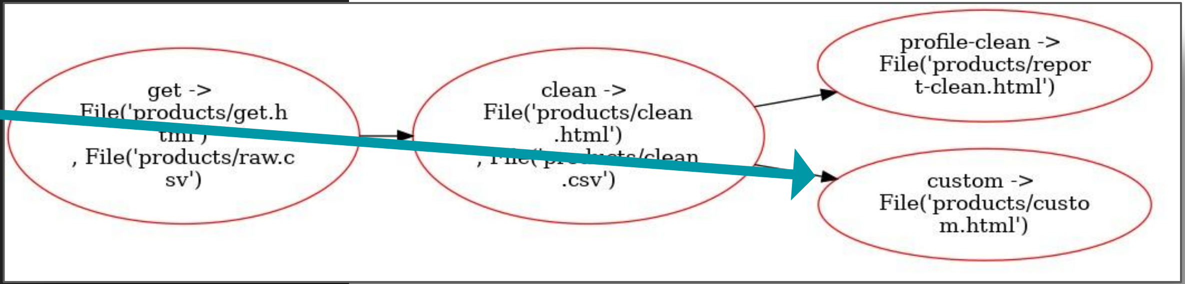
```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Create custom plots.

# %%
import seaborn as sns
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import matplotlib

# %%
# prettier plots
plt.style.use('ggplot')
# larger plots
matplotlib.rc('figure', figsize=(15, 10))
# larger fonts
sns.set_context('notebook', font_scale=1.5)

# %% tags=["parameters"]
upstream = ['clean']
product = None

# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
df.head()

# %%
_ = df['species'].value_counts().plot(kind='bar')

# %%
fig = px.histogram(df, x="bill_length_mm")
fig.show()

# %%
sns.pairplot(df, hue="species", height=3,diag_kind="hist")

# %%
```

```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── custom.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```

```python
# %% [markdown]
# Create custom plots.

# %%
import seaborn as sns
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import matplotlib

# %%
# prettier plots
plt.style.use('ggplot')
# larger plots
matplotlib.rc('figure', figsize=(15, 10))
# larger fonts
sns.set_context('notebook', font_scale=1.5)

# %% tags=["parameters"]
upstream = ['clean']
product = None

# %%
df = pd.read_csv(upstream['clean']['data'])

# %%
df.head()

# %%
_ = df['species'].value_counts().plot(kind='bar')

# %%
fig = px.histogram(df, x="bill_length_mm")
fig.show()

# %%
sns.pairplot(df, hue="species", height=3,diag_kind="hist")

# %%
```

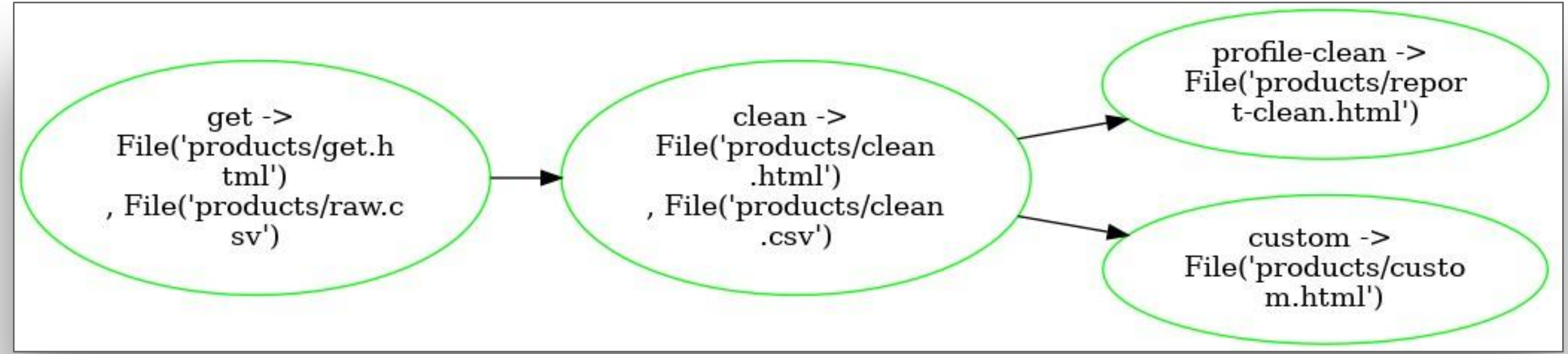
```
→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.yaml
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

1 directory, 11 files
```



```
$ ploomber build
```

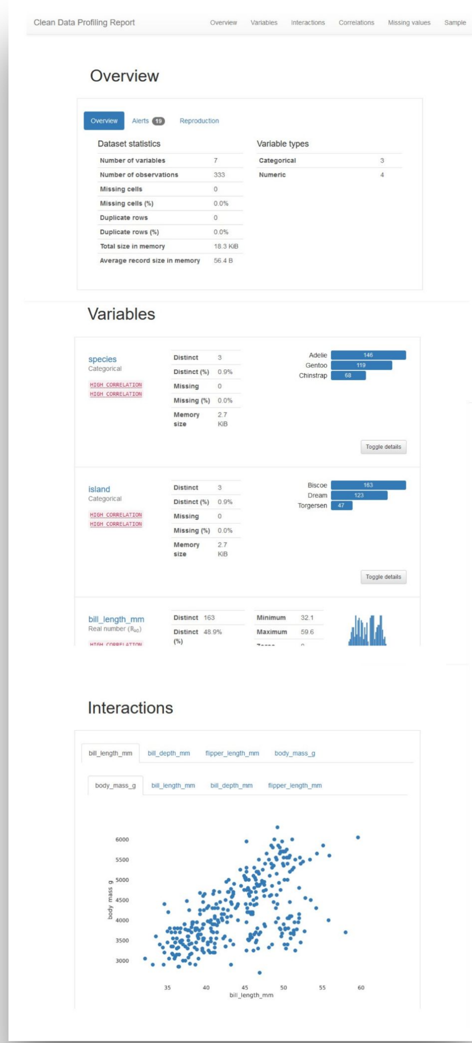
name	Ran?	Elapsed (s)	Percentage
get	True	1.80789	12.0805
clean	True	1.10755	7.40075
profile-clean	True	7.13636	47.6859
custom	True	4.91354	32.8328

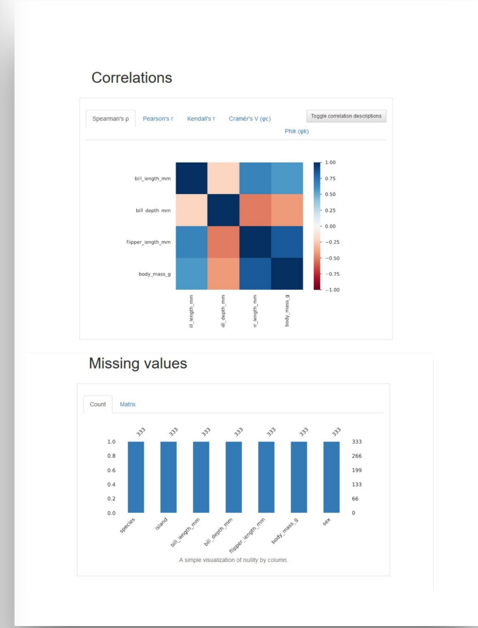
name	Ran?	Elapsed (s)	Percentage
get	True	1.80789	12.0805
clean	True	1.10755	7.40075
profile-clean	True	7.13636	47.6859
custom	True	4.91354	32.8328



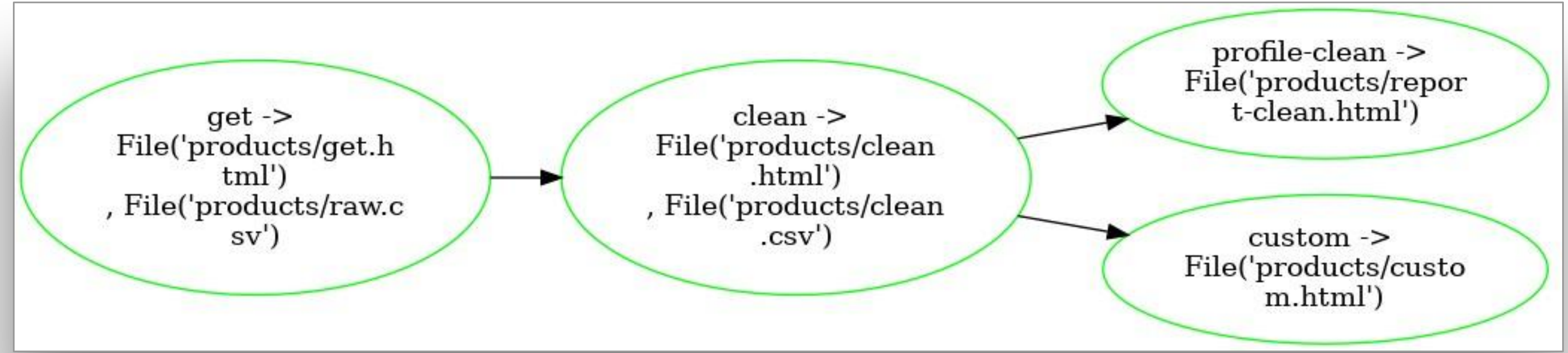
```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.png
├── pipeline.yaml
├── products
│   ├── clean.csv
│   ├── clean.html
│   ├── custom.html
│   ├── get.html
│   ├── raw.csv
│   └── report-clean.html
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

2 directories, 18 files
```


→ exploratory-analysis tree
.
├── environment.yml
├── pipeline.png
├── pipeline.yaml
├── products
│   ├── clean.csv
│   ├── clean.html
│   ├── custom.html
│   ├── get.html
│   ├── raw.csv
│   └── report-clean.html
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md
2 directories, 18 files

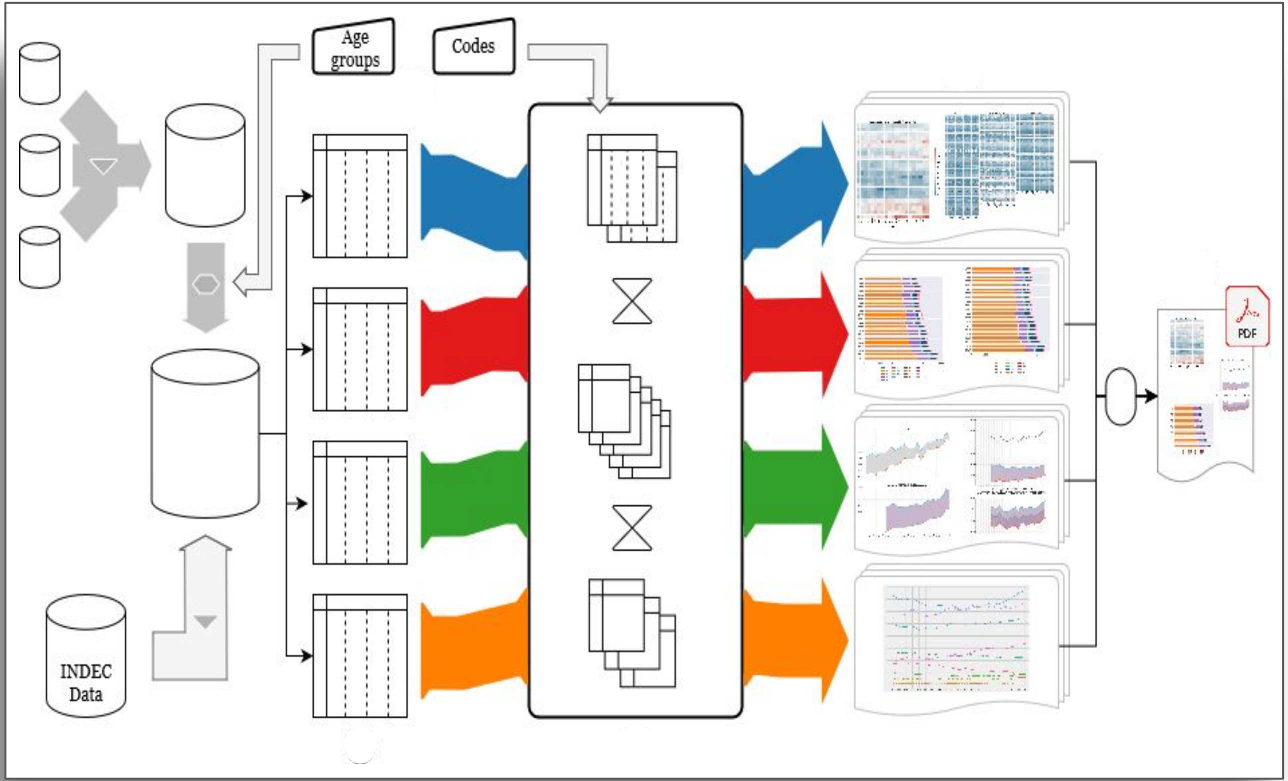

Clean Data Profiling Report
Overview
Variables
Interactions
Correlations
Missing values
Sample
Overview
Alerts 19
Reproduction
Dataset statistics
Number of variables 7
Number of observations 333
Missing cells 0
Missing cells (%) 0.0%
Duplicate rows 0
Duplicate rows (%) 0.0%
Total size in memory 18.3 KiB
Average record size in memory 56.4 B
Variable types
Categorical 3
Numeric 4
Variables
species
Categorical
island
Categorical
bill_length_mm
Adelie 146
Gentoo 119
Chinstrap 68
Biscoe 163
Dream 123
Torgersen 47
Toggle details
Interactions
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g


Correlations
Spearman's ρ
Pearson's r
Kendall's τ
Cramér's V (φc)
Phik (φk)
Toggle correlation descriptions
Missing values
Count
Matrix
A simple visualization of nullity by column.

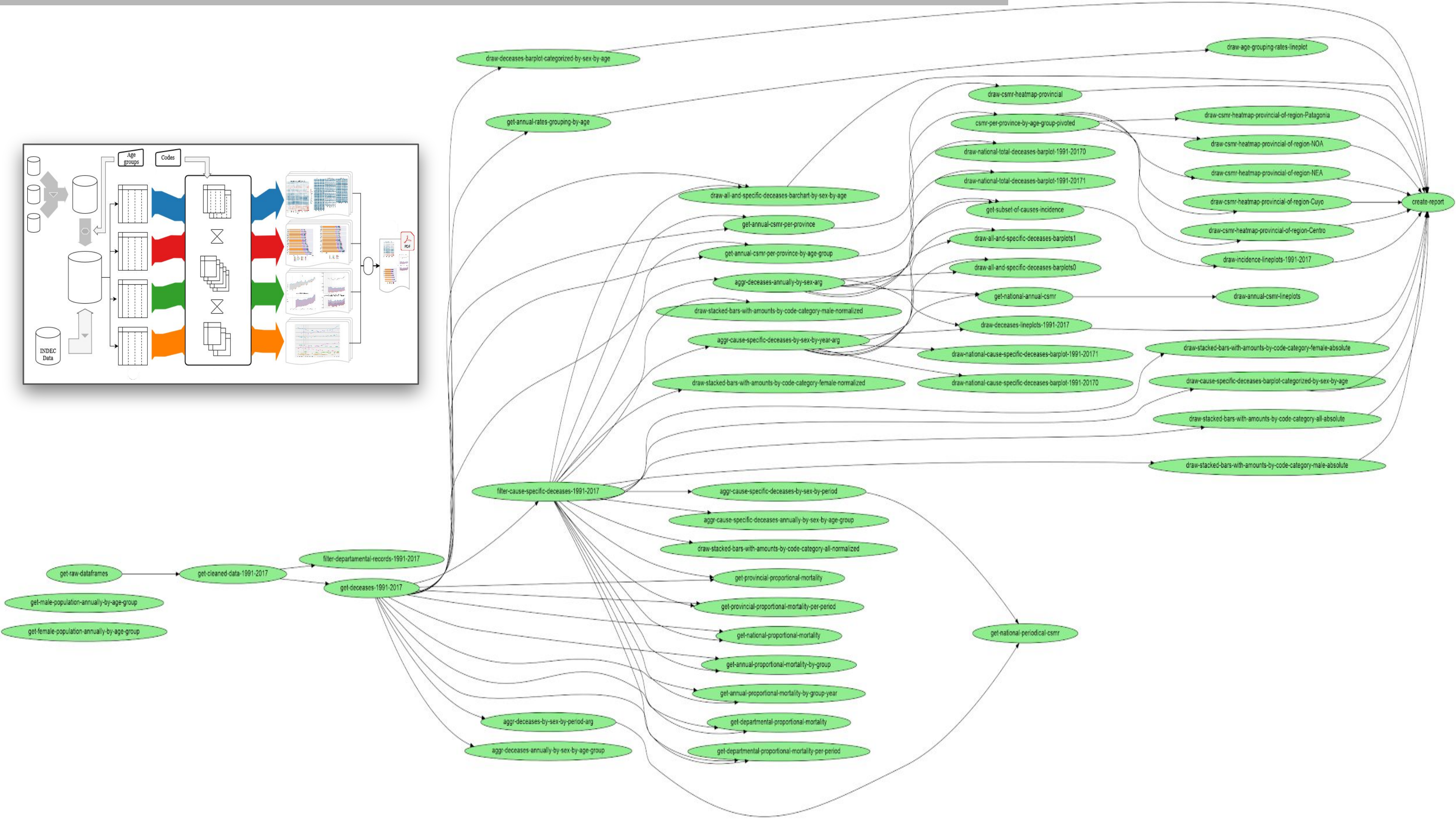
```
→  exploratory-analysis tree
.
├── environment.yml
├── pipeline.png
├── pipeline.yaml
├── products
│   ├── clean.csv
│   ├── clean.html
│   ├── custom.html
│   ├── get.html
│   ├── raw.csv
│   └── report-clean.html
├── README.ipynb
├── README.md
├── requirements.txt
├── scripts
│   ├── clean.py
│   ├── custom.py
│   ├── get.py
│   ├── profile-clean.py
│   └── profile-raw.py
└── _source.md

2 directories, 18 files
```

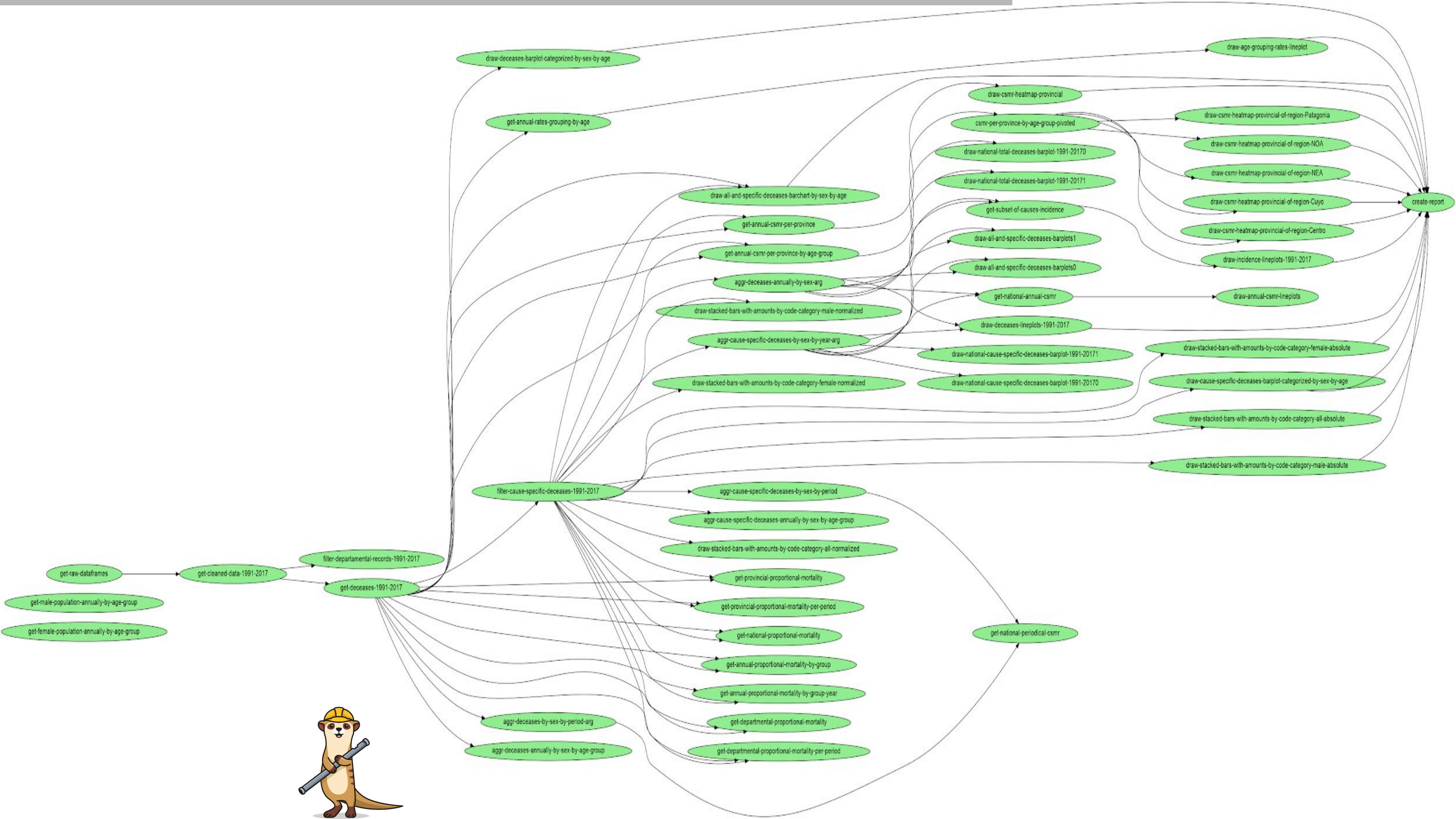


get ->
File('products/get.h
tml')
, File('products/raw.c
sv')
clean ->
File('products/clean
.html')
, File('products/clean
.csv')
profile-clean ->
File('products/repor
t-clean.html')
custom ->
File('products/custo
m.html')



Age groups
Codes
INDEC Data
PDF


Age groups
Codes
INDEC Data
PDF
draw-deceases-barplot-categorized-by-sex-by-age
draw-age-grouping-rates-lineplot
draw-csmr-heatmap-provincial
get-annual-rates-grouping-by-age
csmr-per-province-by-age-group-pivoted
draw-csmr-heatmap-provincial-of-region-Patagonia
draw-csmr-heatmap-provincial-of-region-NOA
draw-national-total-deceases-barplot-1991-20170
draw-national-total-deceases-barplot-1991-20171
draw-csmr-heatmap-provincial-of-region-NEA
draw-all-and-specific-deceases-barchart-by-sex-by-age
draw-csmr-heatmap-provincial-of-region-Cuyo
create-report
get-subset-of-causes-incidence
get-annual-csmr-per-province
draw-csmr-heatmap-provincial-of-region-Centro
draw-all-and-specific-deceases-barplots1
get-annual-csmr-per-province-by-age-group
draw-incidence-lineplots-1991-2017
draw-all-and-specific-deceases-barplots0
aggr-deceases-annually-by-sex-arg
get-national-annual-csmr
draw-annual-csmr-lineplots
draw-stacked-bars-with-amounts-by-code-category-male-normalized
draw-deceases-lineplots-1991-2017
aggr-cause-specific-deceases-by-sex-by-year-arg
draw-stacked-bars-with-amounts-by-code-category-female-absolute
draw-national-cause-specific-deceases-barplot-1991-20171
draw-stacked-bars-with-amounts-by-code-category-female-normalized
draw-national-cause-specific-deceases-barplot-1991-20170
draw-cause-specific-deceases-barplot-categorized-by-sex-by-age
draw-stacked-bars-with-amounts-by-code-category-all-absolute
draw-stacked-bars-with-amounts-by-code-category-male-absolute
filter-cause-specific-deceases-1991-2017
aggr-cause-specific-deceases-by-sex-by-period
aggr-cause-specific-deceases-annually-by-sex-by-age-group
draw-stacked-bars-with-amounts-by-code-category-all-normalized
filter-departamental-records-1991-2017
get-raw-dataframes
get-cleaned-data-1991-2017
get-provincial-proportional-mortality
get-deceases-1991-2017
get-male-population-annually-by-age-group
get-provincial-proportional-mortality-per-period
get-female-population-annually-by-age-group
get-national-proportional-mortality
get-national-periodical-csmr
get-annual-proportional-mortality-by-group
get-annual-proportional-mortality-by-group-year
aggr-deceases-by-sex-by-period-arg
get-departmental-proportional-mortality
aggr-deceases-annually-by-sex-by-age-group
get-departmental-proportional-mortality-per-period


draw-deceases-barplot-categorized-by-sex-by-age
draw-age-grouping-rates-lineplot
draw-csmr-heatmap-provincial
get-annual-rates-grouping-by-age
csmr-per-province-by-age-group-pivoted
draw-csmr-heatmap-provincial-of-region-Patagonia
draw-csmr-heatmap-provincial-of-region-NOA
draw-national-total-deceases-barplot-1991-20170
draw-csmr-heatmap-provincial-of-region-NEA
draw-national-total-deceases-barplot-1991-20171
draw-all-and-specific-deceases-barchart-by-sex-by-age
draw-csmr-heatmap-provincial-of-region-Cuyo
create-report
get-subset-of-causes-incidence
get-annual-csmr-per-province
draw-csmr-heatmap-provincial-of-region-Centro
draw-all-and-specific-deceases-barplots1
get-annual-csmr-per-province-by-age-group
draw-incidence-lineplots-1991-2017
draw-all-and-specific-deceases-barplots0
aggr-deceases-annually-by-sex-arg
get-national-annual-csmr
draw-annual-csmr-lineplots
draw-stacked-bars-with-amounts-by-code-category-male-normalized
draw-deceases-lineplots-1991-2017
aggr-cause-specific-deceases-by-sex-by-year-arg
draw-stacked-bars-with-amounts-by-code-category-female-absolute
draw-national-cause-specific-deceases-barplot-1991-20171
draw-stacked-bars-with-amounts-by-code-category-female-normalized
draw-national-cause-specific-deceases-barplot-1991-20170
draw-cause-specific-deceases-barplot-categorized-by-sex-by-age
draw-stacked-bars-with-amounts-by-code-category-all-absolute
draw-stacked-bars-with-amounts-by-code-category-male-absolute
filter-cause-specific-deceases-1991-2017
aggr-cause-specific-deceases-by-sex-by-period
aggr-cause-specific-deceases-annually-by-sex-by-age-group
draw-stacked-bars-with-amounts-by-code-category-all-normalized
filter-departamental-records-1991-2017
get-raw-dataframes
get-cleaned-data-1991-2017
get-deceases-1991-2017
get-provincial-proportional-mortality
get-male-population-annually-by-age-group
get-provincial-proportional-mortality-per-period
get-female-population-annually-by-age-group
get-national-proportional-mortality
get-national-periodical-csmr
get-annual-proportional-mortality-by-group
get-annual-proportional-mortality-by-group-year
aggr-deceases-by-sex-by-period-arg
get-departmental-proportional-mortality
aggr-deceases-annually-by-sex-by-age-group
get-departmental-proportional-mortality-per-period

Conclusiones

1) Modularización → mantenimiento

Modularización → mantenimiento

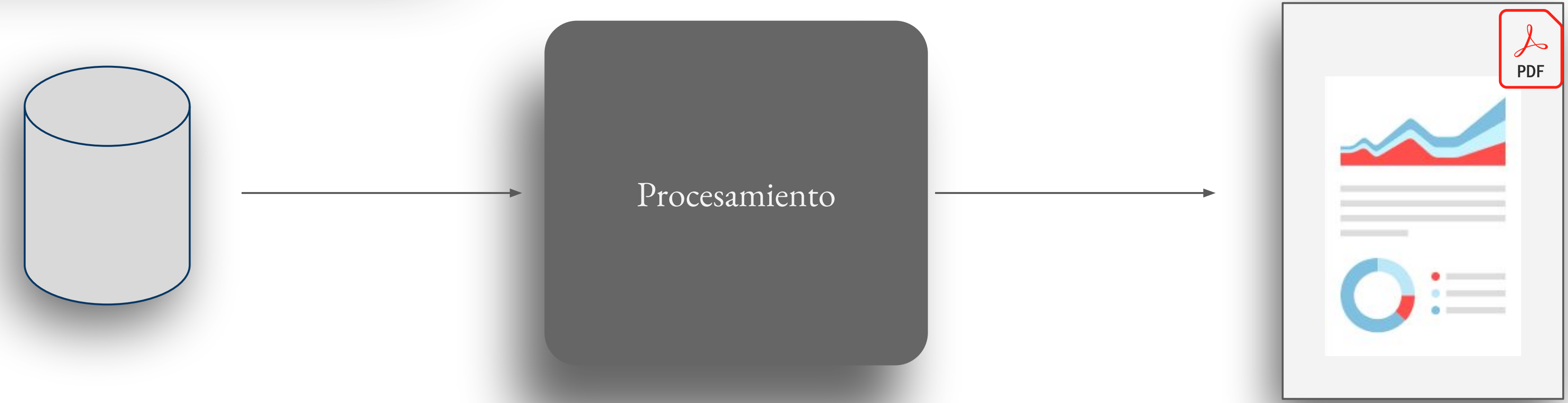
	AÑO	PROVRES	DEPRES	CUENTA
0	1994.0	2.0	1.0	1380.0
1	1994.0	2.0	2.0	1321.0
2	1994.0	2.0	3.0	881.0
3	1994.0	2.0	4.0	933.0
4	1994.0	2.0	5.0	1951.0

Modularización → mantenimiento

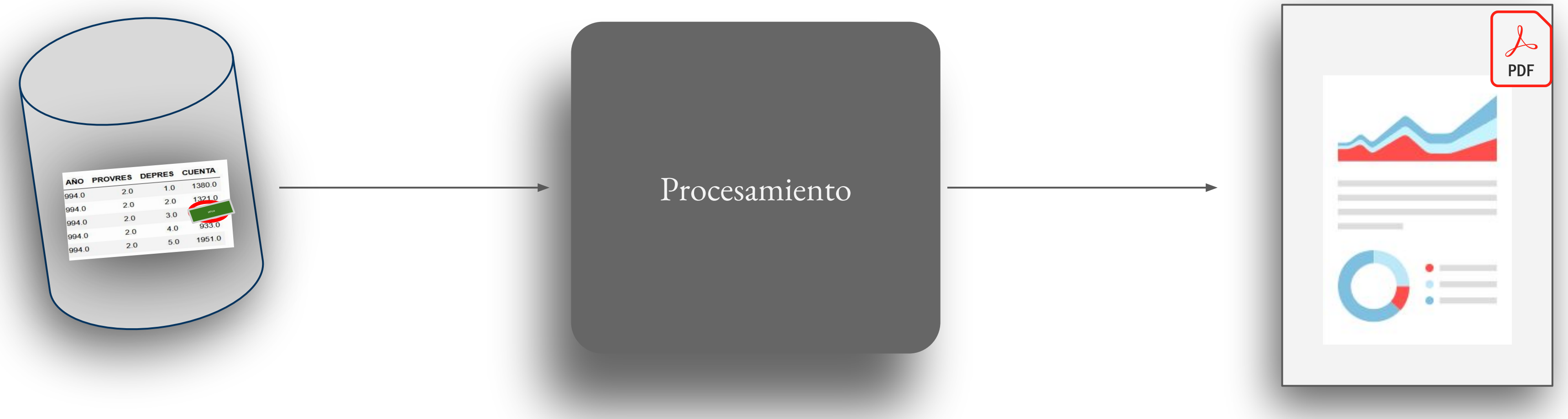
	AÑO	PROVRES	DEPRES	CUENTA
0	1994.0	2.0	1.0	1380.0
1	1994.0	2.0	2.0	1321.0
2	1994.0	2.0	3.0	1832.0
3	1994.0	2.0	4.0	933.0
4	1994.0	2.0	5.0	1951.0

Modularización → mantenimiento

	AÑO	PROVRES	DEPRES	CUENTA
0	1994.0	2.0	1.0	1380.0
1	1994.0	2.0	2.0	1321.0
2	1994.0	2.0	3.0	1832.0
3	1994.0	2.0	4.0	933.0
4	1994.0	2.0	5.0	1951.0

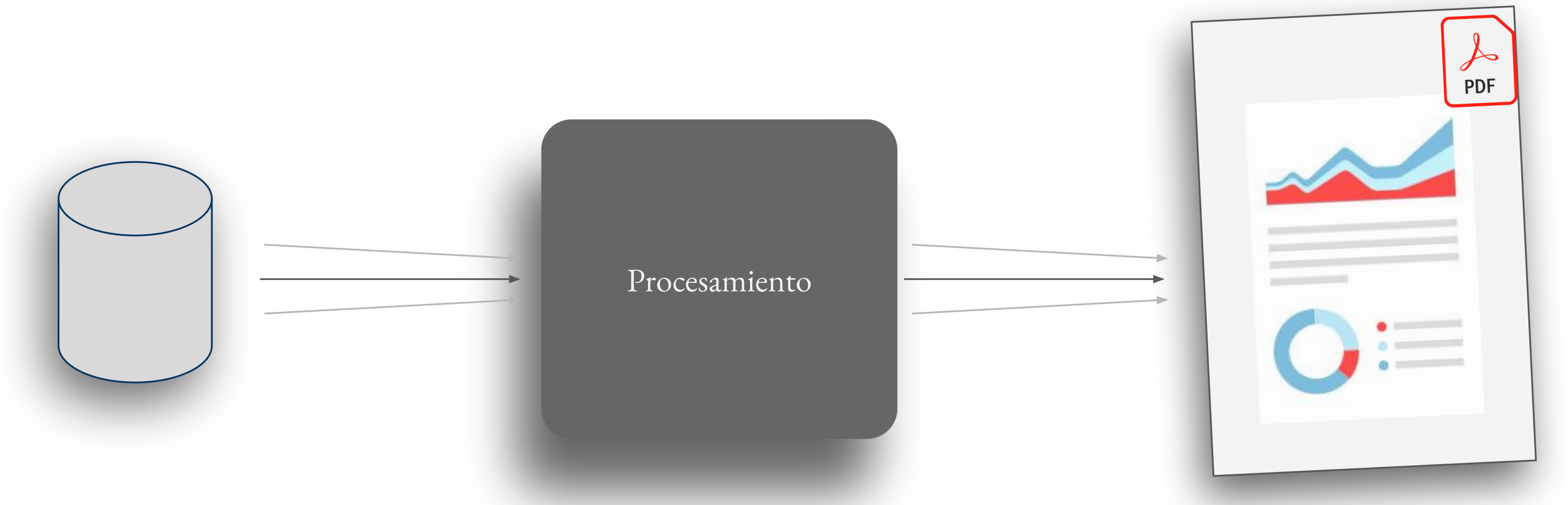


Modularización → mantenimiento



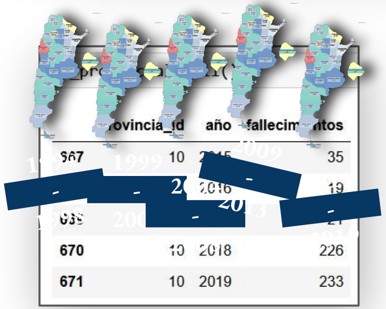
```
$ ploomber build
```

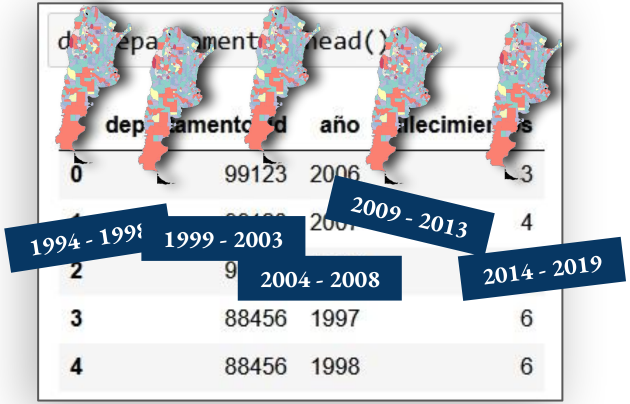
Modularización → mantenimiento

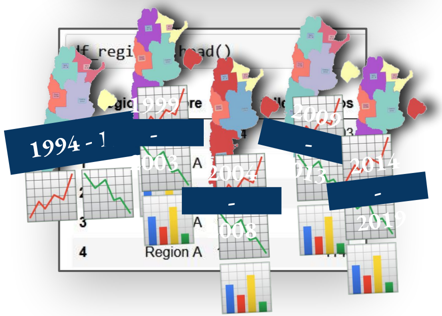


2) Modularización → evolución

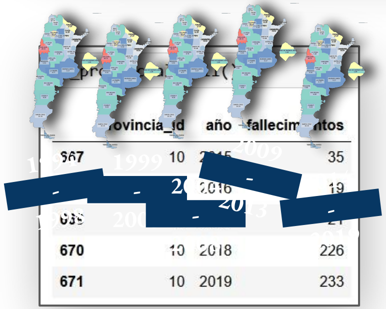
	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN

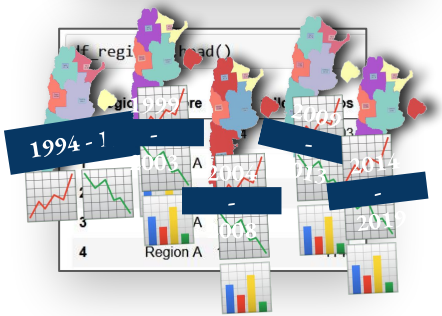


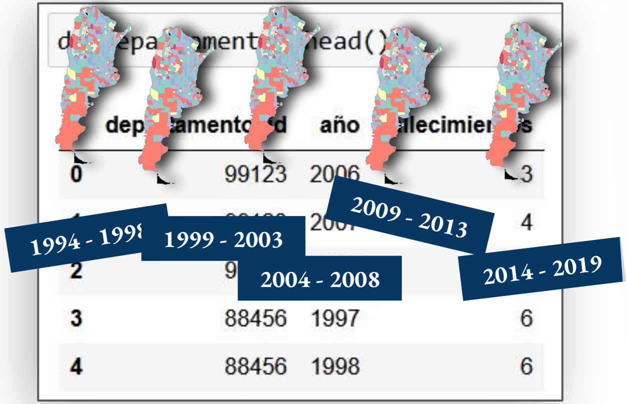


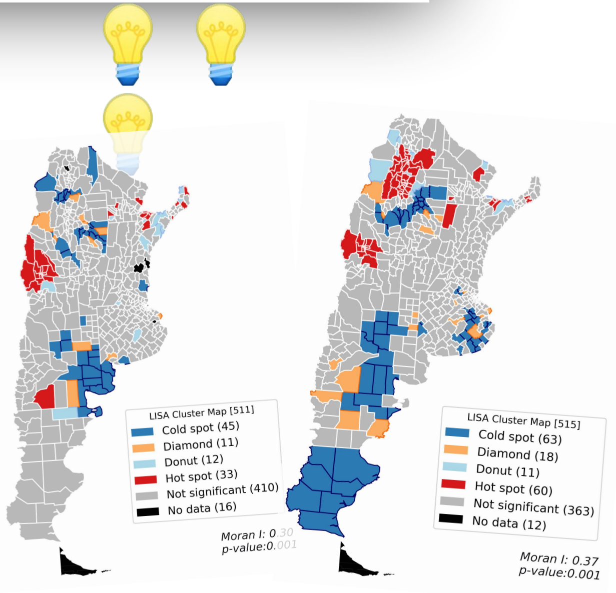


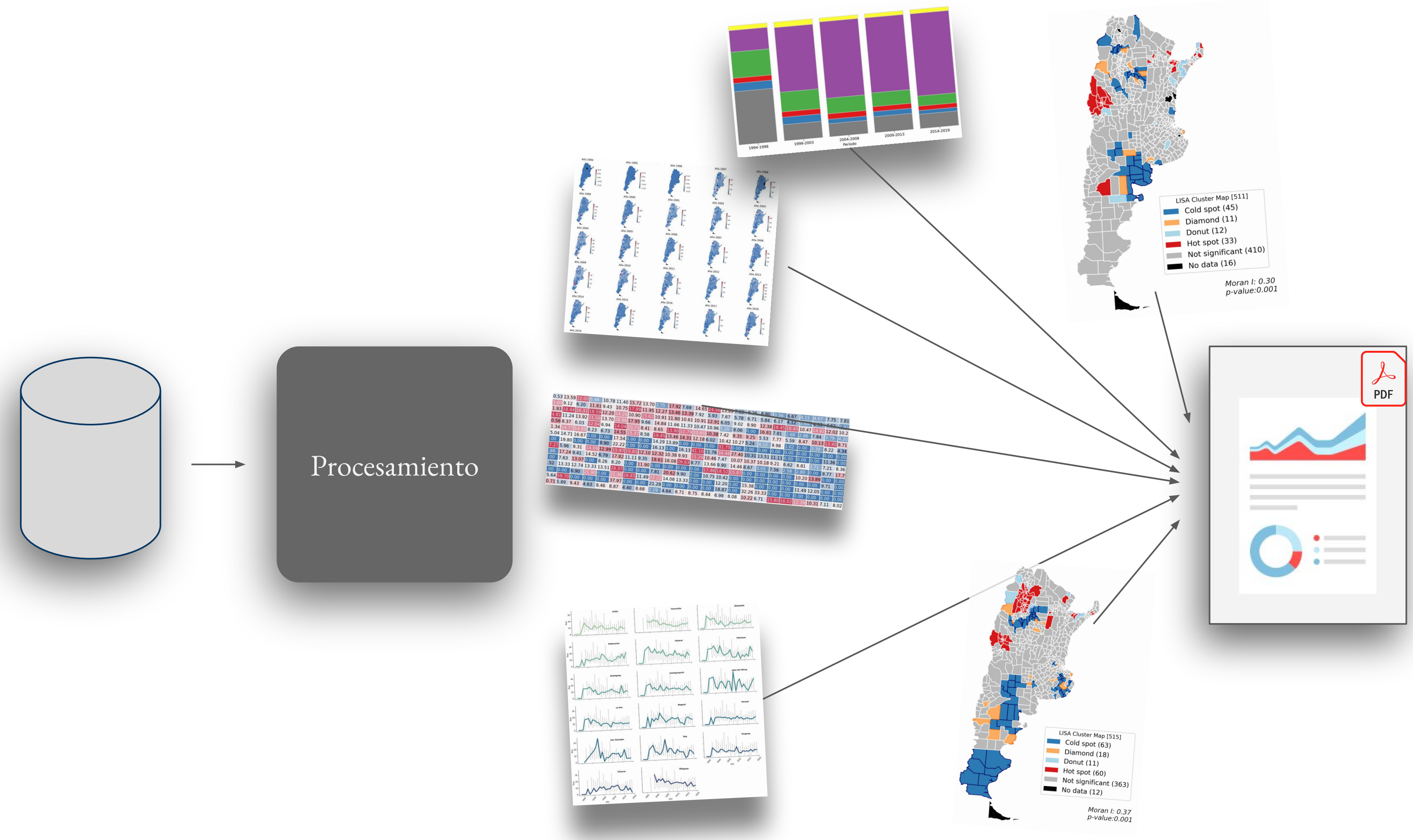
	AÑO	JURIREG	PROVRES	DEPRES	CAUSAMUERCIE10	TIEMGEST	PESOFETO	Unnamed: 7	Unnamed: 8
0	1994.0	62.0	62.0	NaN	A41	20.0	300.0	NaN	NaN
1	1994.0	62.0	62.0	NaN	A41	35.0	2600.0	NaN	NaN
2	1994.0	6.0	6.0	NaN	A50	34.0	3050.0	NaN	NaN
3	1994.0	6.0	6.0	NaN	A50	33.0	2000.0	NaN	NaN
4	1994.0	6.0	6.0	NaN	A50	37.0	3070.0	NaN	NaN



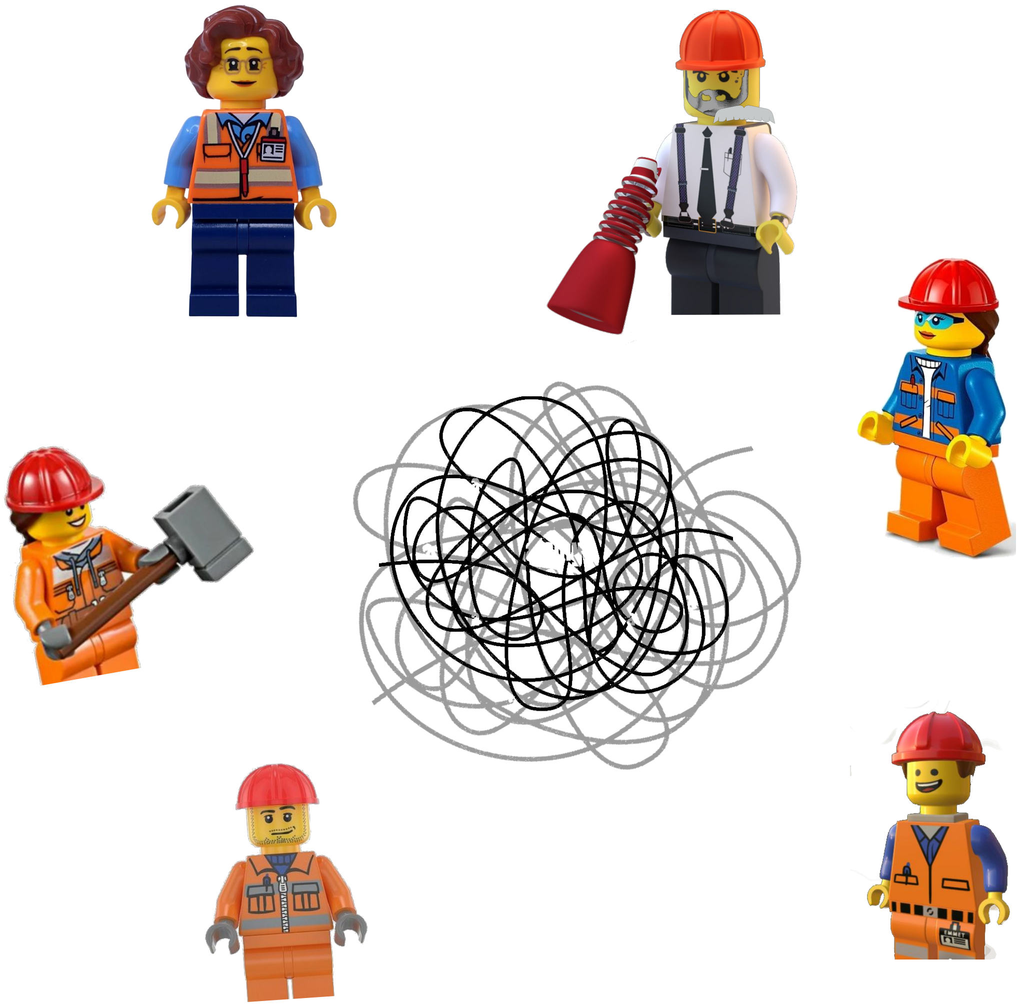






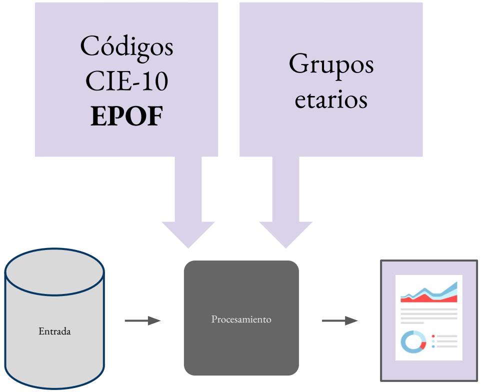

Procesamiento
1994-1998
1999-2003
2004-2008
Periodo
2009-2013
2014-2019
LISA Cluster Map [511]
Cold spot (45)
Diamond (11)
Donut (12)
Hot spot (33)
Not significant (410)
No data (16)
Moran I: 0.30
p-value:0.001
PDF
LISA Cluster Map [515]
Cold spot (63)
Diamond (18)
Donut (11)
Hot spot (60)
Not significant (363)
No data (12)
Moran I: 0.37
p-value:0.001

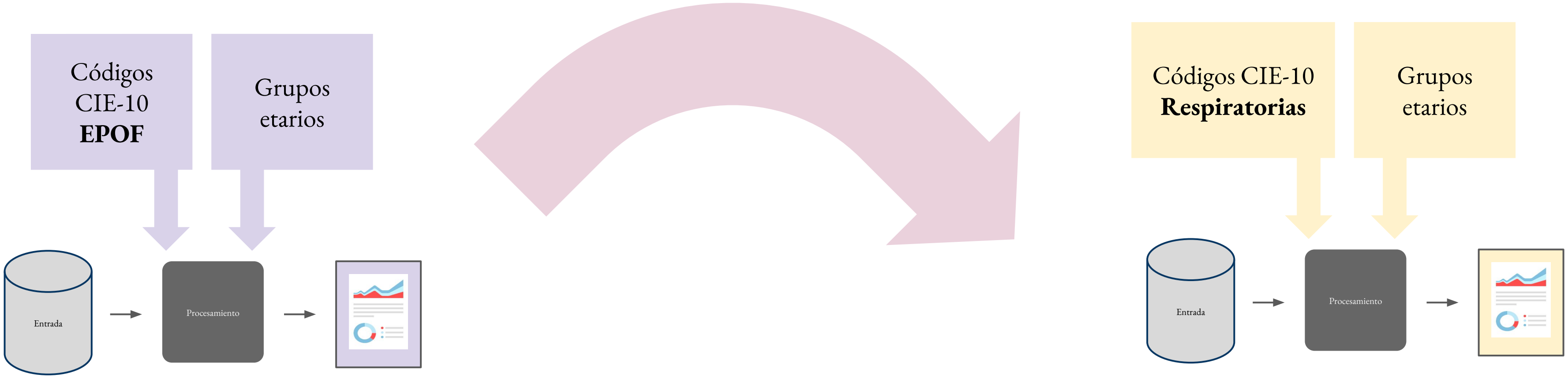
3) Mejora la interacción del equipo

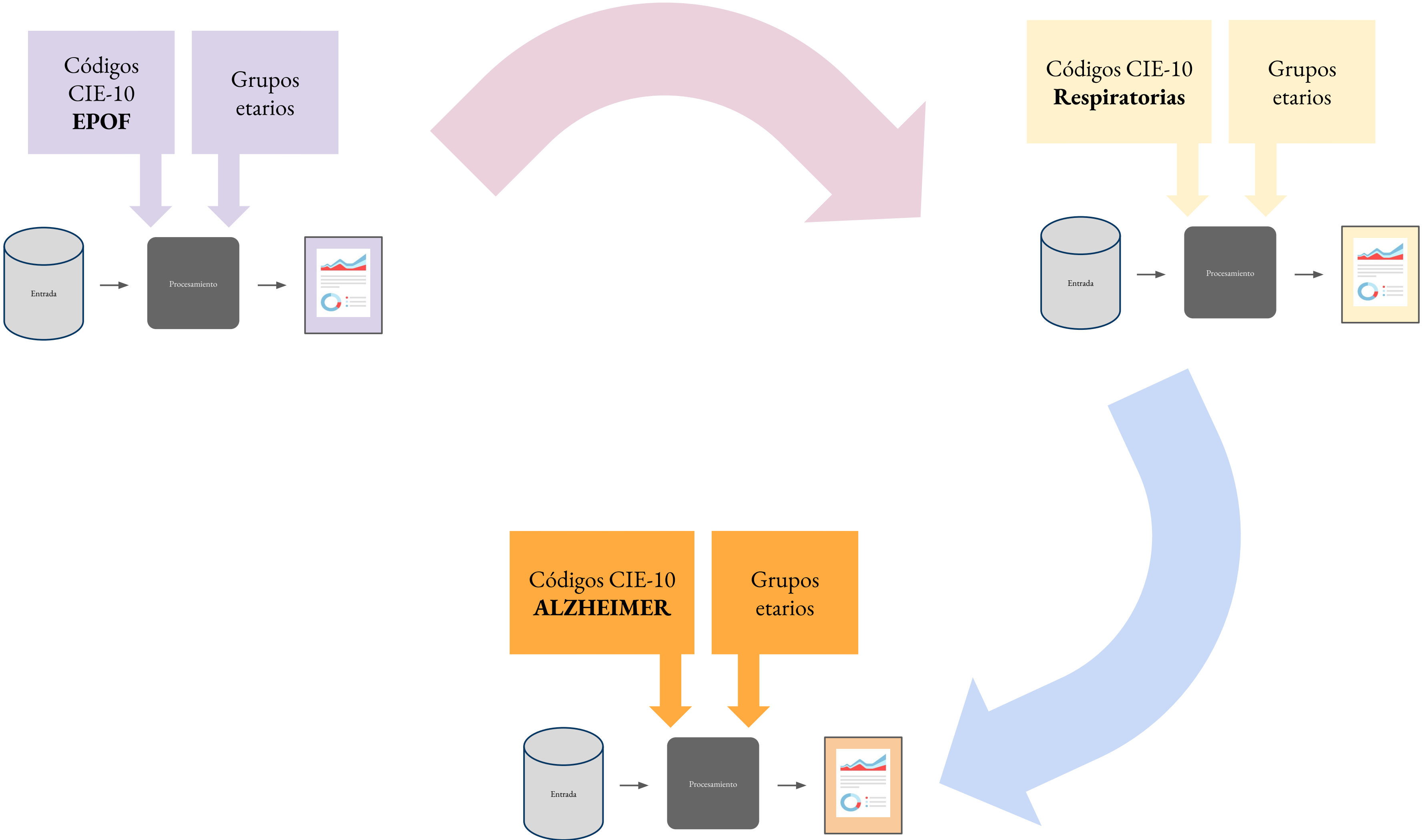


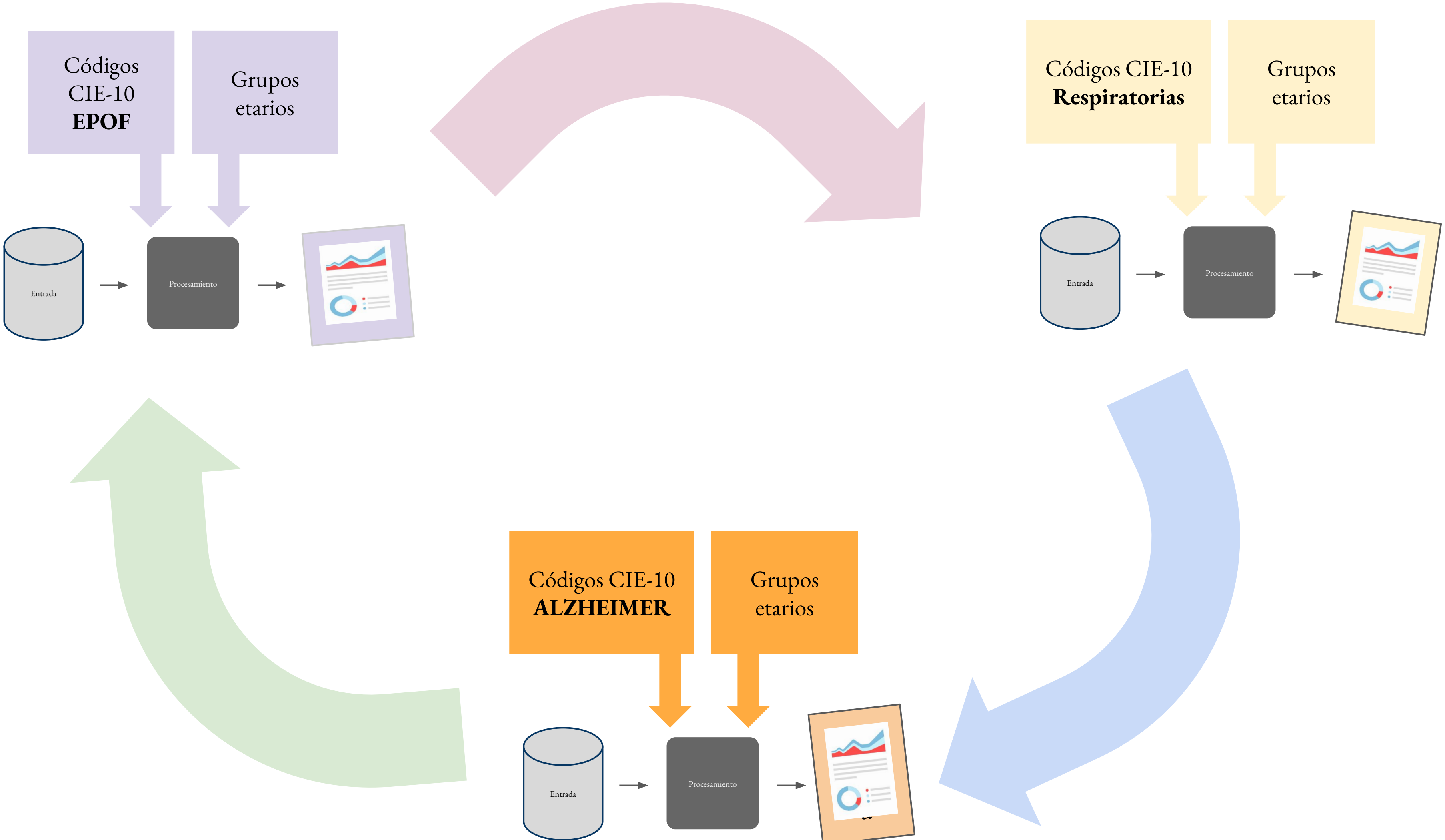

Mortinatos
Análisis regional anual
Parámetros:
Mapas por año
Tabla de calor
Lineplot de tasas anuales por regiones

4) Parametrización → Reutilización

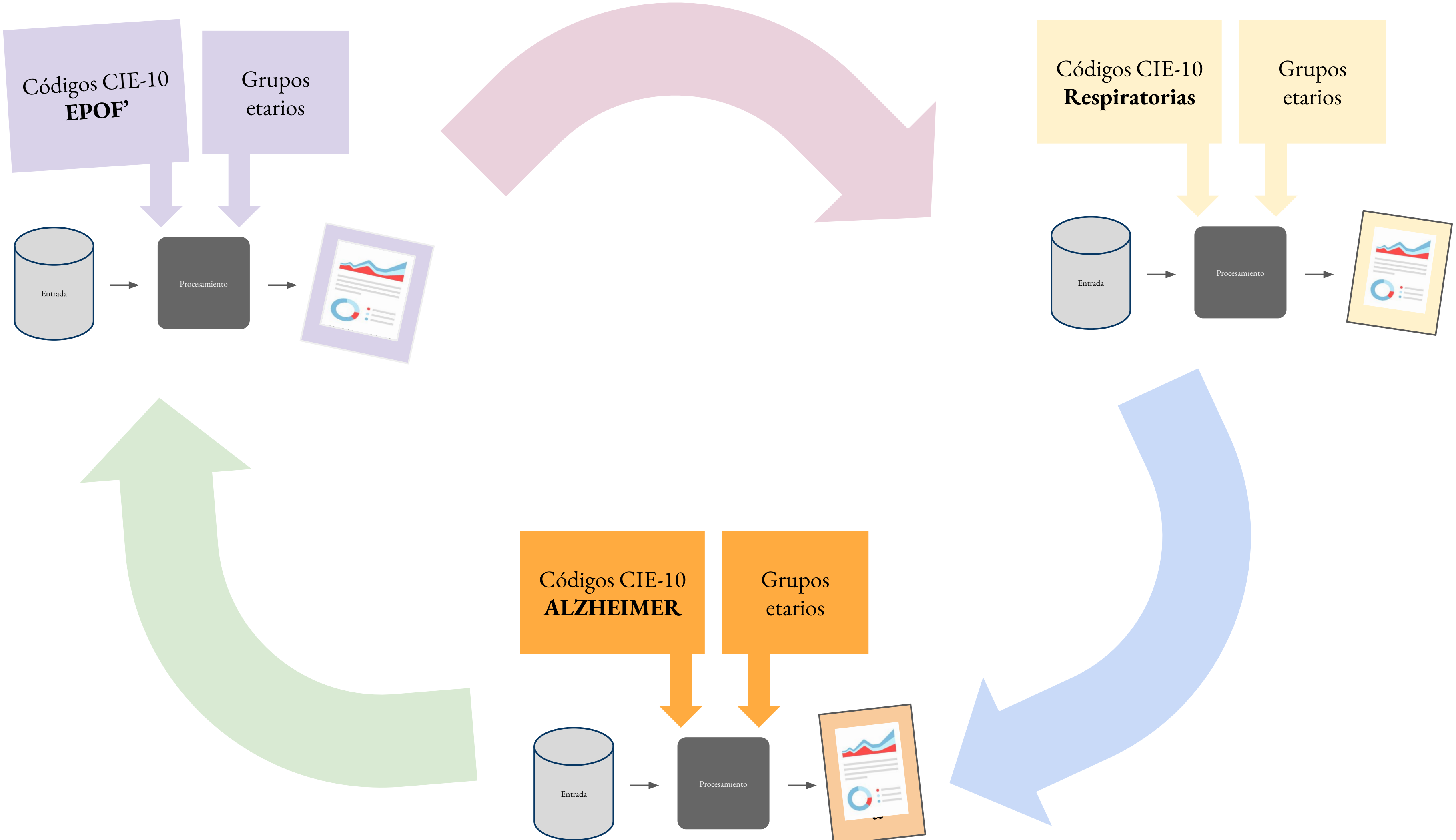

Códigos
CIE-10
EPOF
Grupos
etarios
Entrada
Procesamiento


Códigos CIE-10 EPOF
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 Respiratorias
Grupos etarios
Entrada
Procesamiento

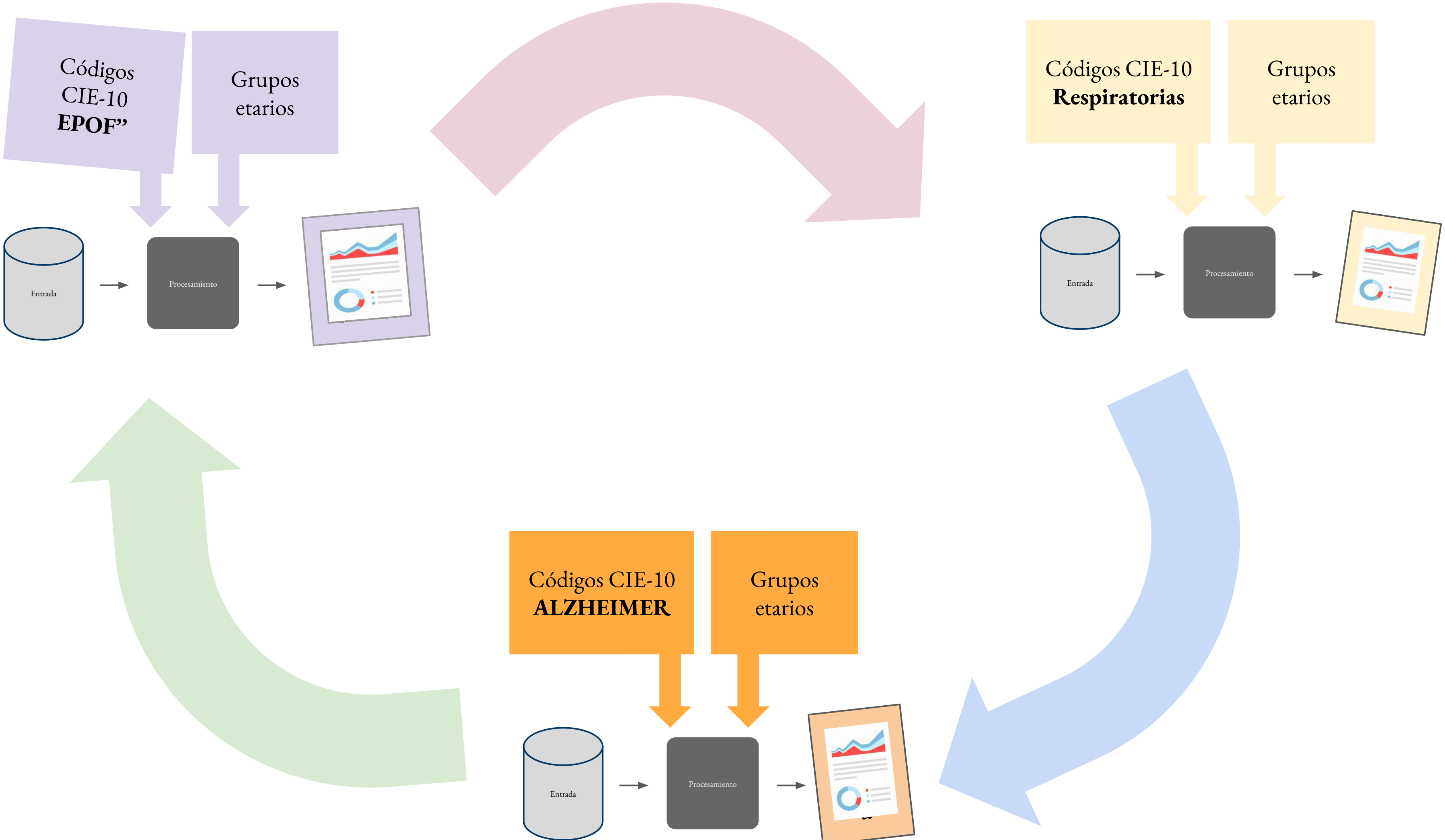

Códigos CIE-10 EPOF
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 Respiratorias
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 ALZHEIMER
Grupos etarios
Entrada
Procesamiento



Códigos CIE-10 EPOF
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 Respiratorias
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 ALZHEIMER
Grupos etarios
Entrada
Procesamiento



Códigos CIE-10 EPOF'
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 Respiratorias
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 ALZHEIMER
Grupos etarios
Entrada
Procesamiento



Códigos CIE-10 EPOF”
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 Respiratorias
Grupos etarios
Entrada
Procesamiento
Códigos CIE-10 ALZHEIMER
Grupos etarios
Entrada
Procesamiento

CONICET IPCSH
Instituto Patagónico de Ciencias Sociales y Humanas
"Dra. María Florencia del Castillo Bernal"
GIBEH
LCI LABORATORIO DE CIENCIAS DE LAS IMAGENES

Herramienta para la exploración de tendencias y detección de patrones epidemiológicos en Argentina

Morales Arturo Leonardo, Figueroa Marcelo , Delrieux Claudio, Ramallo Virginia, Dipierri José Edgardo

Muchas gracias