Clase 028 — Pandas: groupby (split-apply-combine)

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.9 Aggregation and Grouping. · Duración estimada: 90 min.

Clase 028 — Pandas: groupby (split-apply-combine)

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 3 § 3.9 Aggregation and Grouping. · Duración estimada: 90 min.*

Objetivo

Que el alumno aplique el patrón split-apply-combine que es el patrón fundamental de análisis tabular: dividir por grupo, aplicar función, recombinar. Saber elegir entre agg, transform, filter y apply — cada uno tiene su rol.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Agrupar por una o más columnas con groupby y aplicar agregaciones (sum, mean, count).
2. Usar agg con dict para distintas funciones por columna: agg({'a': 'sum', 'b': 'mean'}).
3. transform para preservar la shape original (broadcasting del estadístico de grupo).
4. filter para filtrar grupos enteros según condición.
5. Diferenciar los 4 métodos del groupby y elegir el correcto.

Temas

#	Tema	Por qué importa
1	Split-apply-combine: el patrón	El más común en análisis tabular.
2	agg (= aggregate)	Reduce a una fila por grupo.
3	transform	Misma shape — útil para imputar/normalizar
4	filter	Conserva grupos completos según condición.
5	apply: el más flexible, el más lento	Cuando los 3 anteriores no alcanzan.
6	Múltiples columnas de agrupación	groupby(['a','b']) → MultiIndex.

Definiciones y características

Split-apply-combine

: Patrón: (1) split divide datos por valor de columna(s) → grupos; (2) apply ejecuta función en cada grupo; (3) combine junta resultados. El más usado en análisis tabular.

agg (= aggregate)

: Reduce cada grupo a una fila (sum, mean, count, std). Acepta función nombrada, lista de funciones, o dict por columna: agg({'a': 'sum', 'b': ['min','max']}).

transform

: Aplica función por grupo PERO mantiene la shape original (broadcastea resultado a cada fila). Ideal para z-score por grupo, imputación por grupo, ratios.

filter (groupby)

: Filtra grupos completos (no filas) según una condición. g.filter(lambda x: len(x) > 100) mantiene solo grupos con >100 filas.

apply (groupby)

: El más flexible y el más lento. Cualquier función custom por grupo (puede devolver Series, DataFrame, escalar). Úsalo solo cuando agg/transform/filter no alcanzan.

Named aggregation

: Sintaxis pandas 0.25+: g.agg(total=('monto', 'sum'), n=('id', 'count')). Más legible que el dict tradicional, permite renombrar en el mismo paso.

Dataset / recursos

Palmer Penguins (groupby por species y/o sex).

Ejercicios

1. Agg básico. Penguins agrupado por species: media de cada feature numérica.

2. Agg con dict. Por species: mean de bill_length, max de body_mass, count de filas.

3. Transform: z-score por grupo. Crea columna mass_z = z-score de body_mass dentro de su species.

4. Filter: solo grupos grandes. Conserva solo species con >100 individuos.

5. Apply custom. Por species, devuelve el pingüino con mayor body_mass (un DataFrame por grupo).

Homework verificable

Notebook con penguins: (a) agg múltiple por (species, sex); (b) transform z-score por species; (c) filter species con n>50; (d) apply que devuelva el top-3 más pesado por species; (e) tabla groupby.size() por sex × island.

Criterio de aceptación: z-score por grupo tiene media ≈ 0 y std ≈ 1 dentro de cada species.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
g.apply(...) lanza FutureWarning sobre inc	Pandas 2.2+ cambia comportamiento. Fix: g.
g.mean() solo muestra cols numéricas	Comportamiento intencional (pandas 2+). Fi
g['col'].transform(...) da error "function	Tu función devolvió shape distinta a la en
Resultado de g.agg(...) tiene MultiIndex e	Lista de funciones por columna → MultiInde
groupby(col).size() vs count() dan resulta	size: número de filas por grupo (incluye N

Preguntas frecuentes

¿agg, transform, filter o apply?

agg: reduces a 1 fila por grupo (resumen). transform: mantienes shape original (z-score). filter: excluyes

grupos enteros. apply: lo demás, asumiendo overhead.

¿Cómo agrego columnas usando agg + named?

g.agg(total=('monto', 'sum'), avg=('monto', 'mean'), n=('id', 'count')).reset_index(). Tres columnas nombradas en una sola operación.

¿groupby([a, b]).agg(...).reset_index() o as_index=False?

Equivalentes. as_index=False evita el reset_index() posterior. Para encadenar con merge/concat, as_index=False es más limpio.

¿Cómo agrupo por una expresión derivada?

Pasa Series directa: df.groupby(df['fecha'].dt.year). O crea columna temporal: df.groupby(df['fecha'].dt.year.rename('año')).

¿groupby es lento con miles de grupos?

Con N=1M filas y K=1000 grupos, debería ser <1s. Si es más lento, posibles causas: agg con función custom Python (no built-in), keys con dtype object (string), o falta de sort. Usa sort=False si no necesitas orden.

Referencias

- VanderPlas, cap. 3 § 3.9.
- pandas groupby user guide
- Wickham, "The split-apply-combine strategy for data analysis" (J Stat Software, 2011).

Siguiente clase

Clase 029 — Pandas: pivot tables y crosstab

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import pandas as pd
rng = np.random.default_rng(42)

# Mini-dataset penguin-like
df = pd.DataFrame({
   'species': ['Adelie']5 + ['Chinstrap']4 + ['Gentoo']*5,
   'sex'    : ['M','F','M','F','M', 'M','F','M','F',  'M','F','M','F','M'],
   'masa'   : [3750, 3800, 3650, 3900, 3700,  3500, 3400, 3600, 3550,  5050, 4800, 5200, 4900, 5100],
   'pico'   : [39.1, 39.5, 40.3, 38.8, 39.3,  46.5, 46.0, 46.8, 45.9,  48.6, 47.5, 49.0, 48.2, 48.8],
})
print(df.head())
```

Archivos complementarios

- notebook.ipynb