Metaclasses documentation

Contents

External

Funzione ad alto livello:
Questa metaclasse permette di definire delle variabili che verranno fornite alla classe creata o tramite parametri dell'instanziazione o tramite funzioni specifiche

Definisce anche i metodi per accedere ai risultati

Top level keyword:

- external

Schema della configurazione: è un dizionario che ha come chiavi stringhe. I valori associati sono dei dizionari che possono contenere le seguenti chiavi:

- init: opzionale, valore di default `False`
- default: opzionale, compare solo dove `init` è `True` e indica il valore di default della variabile
- targets: lista di dizionari, ogni dizionario genera una funzione che accede alla stessa variabile ma restituisce informazioni diverse (Opzionale)

Le seguenti chiavi sono mutualmente esclusive con `targets`:

- name: (Opzionale, di default il valore della chiave di primo livello), indica che nome dare alla funzione di accesso
- columns: indica quali colonne restituire (slice di un dataframe), opzionale
- type: Opzionale, converte il risultato, può essere:
  - int
  - float
  - il nome di una classe definita dall'utente

Le chiavi del dizionario interno a `targets` sono:

- name
- columns

Effetti:

- give_{chiave del dizionario}
- get_{chiave del dizionario o name o chiave di external}
- se init è True allora aggiunge un valore che è richiesto in fase di instanziazione, a meno che default sia presente, in tal caso inserisce il valore di default

Lanes

Lanes fornisce due tipi di funzioni:

- Lane_exec che esegue le operazioni risultanti nell'assegnazione ad una PolEnt di un dato numero di seggi

- propose che viene chiamata dai vari lane exec e fornisce una distribuzione e le informazioni

Lane propose

Funzione ad alto livello:
Similmente a totals queste funzioni vengono differenziate dal primo parametro posizionale fornito alla chiamata

Propose viene chiamata con i seguenti parametri:

- nome della funzione che intendo chiamare
- *info: lista di informazioni dalla più recente alla più vecchia
- distribution, keyword argument che indica la distribuzione già definita
- constraint, kwarg, indica la distribuzione da rispettare (per esempio la distribuzione di seggi ad una lista deve rispettare la distribuzione tra coalizioni)

i valori restituiti sono:

- una distribuzione (dataframe con due colonne, la prima contente nomi di PolEnt, la seconda numeri convertibili a interi)
- Un dizionario contenente informazioni

top level keyword: lanes_propose

Schema della configurazione: è un dizionario con chiavi le stringhe, ogni stringa indica il tipo di propose che il dizionario associato va a definire

- source: contiene una funzione (definita tramite source) che viene chiamata per fornire il punto di partenza all'estrazione di informazioni e della distribuzione. Questa funzione può essere definita in qualunque modo ma deve:
  1. Accettare i parametri a kw: information, constraint, distribution
  2. Restituire un dataframe
- distribution: definisce come ricavare una distribuzione dal dataframe che source restituisce. Può essere:
  - una lista di due stringhe, entrambe colonne del dataframe. In questo caso la prima rappresenta la PolEnt che riceverà i seggi e la seconda i seggi assegnati
  - un dizionario
- info_key (opzionale): se vuoto la chiave delle info è la stessa della chiave della distribuzione, altrimenti indica la PolEnt cui associare le informazioni
- info: una lista di stringhe rappresentanti le colonne del dataframe contenenti informazioni

Schema del dizionario di `distribution`:

- key: la colonna cui associare i seggi
- selector: viene riconosciuto da `parse_row_selector_take` o `parse_row_selector_value`

- seats: può essere
  - un intero, in tal caso tutti i candidati validi riceveranno lo stesso numero di seggi
  - stringa, ogni candidato riceverà il numero di seggi associato alla colonna

Restituisce un dataframe con colonne: [key, "Seats"]

Exec lane

Introduzione al sistema di lanes: la lane è l'unita organizzativa che all'interno di una simulazione assegna ad un certo numero di candidati dei seggi. Le lane sono eseguite sequenzialmente e si dividono in due tipi:

1. Multi step, queste lane rappresentano leggi elettorali nelle quali chi viene eletto in una data circoscrizione dipende non solo dai risultati della circoscrizione ma anche da informazioni esterne al distretto specifico. Per esempio la maggior parte delle leggi elettorali proporzionali assegna i seggi ai partiti in base alla proporzione di voto nazionale, tuttavia i candidati vengono eletti nei singoli distretti, in base ai risultati locali
2. Single step, queste lane rappresentano leggi nelle quali la distribuzione di seggi è influenzata solo dalla distribuzione locale dei voti, l'esempio naturale è quello dei sistemi maggioritari, tuttavia questa classificazione si applica anche a sistemi proporzionali dove la lista elettorale di ogni partito sia unica e nazionale

Funzione ad alto livello: questa funzione viene chiamata originariamente dall'Hub e poi dal livello superiore in lane multi-step, la funzione riceve in input le seguenti informazioni:

- nome della lane
- *informazioni, questa è una lista di dizionari, il primo dizionario rappresenta le informazioni relative al distretto specifico che sta eseguendo la funzione e pertanto può essere modificato direttamente, i dizionari successivi sono condivisi e pertanto da considerare read-only
- distribution: un dataframe con due colonne che indica la distribuzione di seggi determinata dai livelli superiori. La prima colonna è una colonna di stringhe rappresentanti nomi di una PolEnt di classe Elector, la seconda una lista di numeri convertibili in interi

Il risultato è una lista di tuple nella forma:

- electoral_district: l'istanza del distretto che ha effettuato l'elezione
- lane_name: il nome della lane
- elector: una stringa riconducibile a un'istanza di Elector
- seats: un numero convertibile in intero che rappresenta il numero di seggi

top level keyword: lane

Schema: un dizionario dove le chiavi sono il nome della lane e i valori sono dizionari contenenti le specifiche

Contenuti dei dizionari:

- node_type: Uno tra
  - only: specifica una lane single step, non accetta parametri oltre il nome
  - head: il punto di partenza di una lane multi step, non accetta parametri a parte il nome
  - node: un generico punto intermedio di una lane multi-step
  - tail: il nodo di una lane multi step che si occupa esclusivamente di restituire la lista di cui sopra e consolidare e distribuire le i nformazioni
- info_name: Ogni nodo aggiunge alle informazioni generate nel nodo stesso una coppia chiave/valore del tipo: `info_name = self.name`

Solo per tipi only:

- distribution: indica una propose da chiamare che fornisce tutte le informazioni e la distribuzione corretta

Solo per only e head:

- order_number: indica l'ordine di esecuzione della lane

Solo per head:

- first_input: stessa funzione di distribution per i tipi only ma rappresenta il punto di partenza per le operazioni da effettuare

Solo per head e node:

- sub_level: indica il tipo del nodo al livello inferiore
- operations: una lista di dizionari che specificano delle operazioni, queste operazioni sono concatenate per trasformare una distribuzione di input in una da passare al livello inferiore e per generare informazioni aggiuntive

Dizionari di operations:

- collect_type: Il tipo di propose da chiamare per raccogliere proposte dai livelli inferiori
- ideal_distribution: può essere
  - '$', per riferirsi all'ultima distribuzione valida
  - una stringa, per chiamare propose e usare la distribuzione ottenuta
  - un dizionario contenente la chiave source e la definizione di una funzione che prende in input il riferimento al distretto corrente e restituisce un dataframe distribuzione
- corrector: una funzione definita in commons che riceve in input:
  - il distretto corrente
  - la distribuzione ideale

  - un dizionario contenente le distribuzioni raccolte dai livelli inferiori
  - un dizionario contenente le informazioni raccolte dai livelli inferiori
  - una lista di informazioni, il primo elemento riguarda esclusivamente il distretto corrente

  e restituisce:

  - un dizionario di distribuzioni da inoltrare ai livelli inferiori
  - un dizionario di informazioni da inoltrare ai livelli inferiori
  - informazioni comuni a tutti i livelli inferiori

- collect_constraint, può essere:
  - None per non porre limitazioni alla distribuzione generata dai livelli inferiori
  - '$' per inoltrare la distribuzione corrente
  - stringa, per inoltrare il risultato di una propose del distretto corrente
- forward_distribution, se True allora questa operazione modifica solo le informazioni e non la distribuzione

Superdivision

Funzione ad alto livello: Superdivision definisce la relazione tra due classi geografiche specificando quali variabili della classe identificano il sottolivello, di che tipo è il sottolivello e quali funzioni esporre dal sottolivello

Top level keyword: subdivisions

Schema della configurazione: un dizionario di dizionari, le chiavi indicano la variabile in cui salvare i nomi delle istanze di sottolivello

I valori sono dizionari con:

- type: il nome della classe
- functions: una lista di dizionari, ogni dizionario ha:
  - name: il nome della funzione da esporre
  - source: una definizione tramite source della funzione da eseguire. Si assume che nell'esecuzione di source "self" nel namespace si riferirà ad una sottodivisione

La funzione data verrà chiamata su ogni istanza delle sottodivisioni e i risultati saranno aggregati:

- Se di tipo "int" verranno sommati
- Se di tipo dataframe verranno concatenati
- Altrimenti concatenati in una lista

Totals

Funzione ad alto livello: questa metaclasse offre un sistema omogeneo per descrivere operazioni su dataframe senza dover scrivere codice.

Vengono offerti due tipi di funzione:

- totals: a queste funzioni si accede chiamando: `self.totals(nome_funzione, *sbarramenti, **kwargs)`
- support: queste funzioni invece vengono chiamate come `self.nome_funzione(**kwargs)`

Oltre a questo l'unica differenza è che quando la funzione è definita come totals e `*sbarramenti` non è vuota il dataframe risultante è filtrato in base al contenuto di sbarramenti. Si veda la documentazione di `totFilter` per dettagli

Top level keyword: `totals` o `totals_support`

Schema: entrambe si compongono di un dizionario dove le chiavi sono `nome_funzione` e i valori sono dei dizionari con chiavi:

- rename: un dizionario stringa-stringa, si comporta come rename in source
- columns: una lista di stringhe, si comporta come rename in source
- type: uno tra
  - aggregate
  - transform
  - combination

Per aggregate e transform:

- source: la source da chiamare per ricevere il dataframe di input. Qui è dove `**kwargs` e, dove necessario, `*sbarramenti` viene inoltrato

Per aggregate:

- keys: una lista di stringhe che indica quali colonne costituiscono la chiave
- ops: un dizionario nome_colonna -> operazione, dove operazione è:
  - mean: per la media aritmetica
  - median
  - sum
  - prod: per il prodotto
  - o il nome di una funzione in commons

Per combination:

- function: una stringa che indica la funzione da chiamare, questa accettà:
  - 1 dataframe passato posizionalmente
  - un numero arbitrario di argomenti posizionali
  - un numero arbitrario di argomenti passati per chiave
- merge_keys: le chiavi da usare per combinare i vari dataframe
- keys: le chiavi da usare per dividere il dataframe combinato
- args: una lista di:
  - stringa o intero o booleano: uno scalare

  - dizionario contenente le chiavi:
    * type: series, dataframe o scalar
    * source

Nota su combination, combination opera facendo il merge (su merge_keys) di tutti i dataframe e facendo groupby sul risultato (usando keys). Per ogni gruppo risultante sarà chiamata la funzione passando:

- La slice di dataframe corrispondente
- Tutti i parametri scalari
- Per ogni series la linea corrispondente alla chiave del gruppo trattandola come un dizionario passato per kwargs.
  Una series è un oggetto di tipo dataframe con almeno le colonne keys, dove per ogni combinazione di keys c'è una e una sola linea

Per transform:

- ops: una lista di dizionari, a seconda del valore associato alla chiave `type` verrà trattato come:
  - transform column: applica un'operazione a tutti gli elementi di una data colonna, sovrascrivendola o creando una nuova colonna
  - transform line: chiama per ogni linea la funzione usando la linea come dizionario di `**kwargs`
  - transform dataframe: applica la funzione a tutto il dataframe

  Le operazioni vengono eseguite in sequenza

Transform ops:

- column:
  - column: la colonna da modificare
  - column_type (opzionale): se specificato converte le celle nel tipo dato (si veda `type` in external)
  - replace_name (opzionale): di default uguale a column, indica il nome della colonna generata
  - source: la funzione da chiamare fornendo come parametro la cella
- line:
  - source: la funzione da chiamare fornendo come kwargs il dizionario equivalente alla linea
  - column_name: che nome assegnare alla colonna risultante
- dataframe:
  - source: la funzione cui passare il dataframe

Subclass

Funzione ad alto livello: questa classe stabilisce che quando l'hub cercherà istanze di classi specificate nella configurazione allora saranno restituiti anche istanze della classe corrente

Top level keyword: `subclass`

Schema: lista di stringhe, ogni stringa rappresenta una classe

Candidate

Funzione ad alto livello: questa classe definisce le due funzioni necessarie per un candidato:

1. Ricevere proposte di seggi dove si è stati eletti (e eventuali informazioni aggiuntive)
2. Dove ci siano opportunità multiple il candidato deve scegliere quale accettare e quali rifiutare

Top level keyword: `candidate`

Schema:

- `info_vars`: una lista di variabili della classe Candidato che saranno aggiunte alle informazioni restituite quando un candidato viene eletto
- `criteria`: può essere:
  - 'first' per indicare la prima proposta ricevuta
  - una lista di stringhe, le proposte vengono ordinate in base ai valori delle informazioni aggiuntive
  - il nome di una funzione

Logger

Questa metaclasse fornisce una funzione per raccogliere Series pandas contenenti informazioni e restituirle sotto forma di un DataFrame quando richiesto

sources_parse

Questa non è una metaclasse e pertanto non deve (e non può) essere specificata in `metaclasses`, tuttavia la configurazione, prima di essere usata per creare la classe, viene analizzata usando questa funzione.

La funzione cerca, a qualunque livello dell'albero della configurazione, la parola chiave `source` e, dove la trovi, il valore della chiave viene sostituito da una funzione che potrà essere chiamata fornendo come primo argomento posizionale un dizionario che specifichi il namespace nel quale l'esecuzione stia operando

Tale funzione può quindi restituire un qualunque valore come specificato nella configurazione

Schema della configurazione: source può avere come valore associato sia uno scalare (intero, booleano o stringa), nel qual caso la funzione restituirà tale scalare, sia un dizionario con le seguenti chiavi:

- type: può essere
  - fun, per indicare che il risultato sarà ottenuto chiamando una funzione
  - att, per indicare che il risultato sarà un attributo o una variabile
  - kwarg, per indicare che il risultato sarà un argomento a keyword del namespace originale
- columns: una lista di stringhe nella forma: `colonna( -> nuovo_nome)?`, presuppone che il risultato sia un dataframe e retituisce solo la slice formata dalle colonne specificate (nell'ordine specificato) eventualmente con il nome modificato
- rename: un dizionario del tipo: `colonna: nuovo_nome`, assume un risultato di tipo dataframe e rinomina le colonne usando il dizionario. In un conflitto tra rename implicito nelle columns e esplicito in rename il rename specificato in questo dizionario ha precedenza
- store: una stringa nella forma
  - `#nome` il risultato sarà salvato in `self.nome`
  - `$nome.chiave` il risultato sarà salvato nel dizionario `nome` come `chiave`
  - `nome` il risultato sarà aggiunto al namespace nella variabile `nome`
- options: una lista di stringhe che modificano il comportamento
  - `NoForward`, se presente, non inoltra gli argomenti posizionali e a keyword con cui si chiama `source` alle funzioni sottostanti

Se type è fun:

- name: il nome della funzione (viene trovata facendo `eval(name, globals(), locals)`)
- args: una lista di valori accettabili come configurazioni di source, passati come argomenti posizionali
- kwargs: come args ma un dizionario con chiavi stringhe e passato come kwargs