ES-PEC4-enun

June 3, 2020

1 Programación para la ciencia de datos - PEC4

En este Notebook encontraréis un ejercicio que supone la cuarta actividad de evaluación continua (PEC) de la asignatura. Esta PEC consta de un único ejercicio a resolver, que engloba muchos de los conceptos vistos durante la asignatura.

El objetivo de este ejercicio es desarrollar un **paquete de Python**, fuera del entorno de Notebooks, que nos permita resolver el problema dado. Este tendrá que incluir el correspondiente código organizado lógicamente (separado en módulos, organizados por funcionalidades), la documentación del código (docstrings) y los tests. Además, se tendrán que incluir los correspondientes archivos de documentación de alto nivel (README), así como los archivos de licencia y dependencias (`requirements.txt`).

2 Ejercicio

Se nos pide que implementemos un paquete de Python que sea capaz de realizar un análisis sencillo de datos de Bitcoin. Bitcoin es una criptomoneda descentralizada, que opera enviando *transacciones* a través de una red P2P. Estas transacciones se agrupan en una estructura de datos conocida como *bloque*.

Tendréis que generar funciones que os permitan encontrar: 1. El número de transacciones por cada bloque. 2. El valor de cada transacción de cada bloque. 3. El tiempo entre bloques. 4. La media del tamaño del bloque por hora. 5. El número de transacciones por hora.

Además, tendréis que generar el código que permita representar todos los resultados gráficamente. Para cada métrica, es importante que penséis que tipo de gráfica es la más conveniente para representarla.

El código tendrá que estar correctamente comentado, incluyendo la documentación de las funciones, y correctamente testeado.

2.1 Los datos

Los datos para analizar nos son facilitados en dos archivos separados: `blocks.json` y `txs.json`.

`blocks.json` contiene, en cada línea del fichero, la información de un bloque en formato JSON. De forma análoga, `txs.json` contiene, en cada línea, la información de una transacción también en formato JSON.

Los archivos contienen información sobre 144 bloques, lo que equivale, aproximadamente, a un dia de datos. Las transacciones de un archivo `txs.json` corresponden a los bloques del archivo `blocks.json`.

Echando un vistazo a los archivos proporcionados, podéis ver que tanto las transacciones como los bloques contienen bastante información. Para resolver el ejercicio propuesto, solo será necesario utilitzar una cantidad muy reducida de esta.

2.1.1 Bloque

Por lo que se refiere a un bloque, nos interesa saber que transacciones contiene (campo `tx`), cuando fue generado (campo `time`) y cual es su tamaño (campo `size`).

2.1.2 Transacción

Por lo que se refiere a una transacción, nos interesa saber a que bloque pertenece y cual es el valor de esta transacción.

Por lo que se refiere a que bloque pertenece una transacción, el identificador que aparece en la lista `tx` dentro de cada bloque se corresponde con el campo `txid` dentro de la transacción. Además, la transacción incluye el campo `blockhash` que se corresponde con el campo `hash` del bloque.

Por otro lado, el valor de cada transacción no lo encontraréis directamente en ningún campo, sino que lo tendréis que calcular. Para hacerlo, tendréis que sumar los valores del campo `value` dentro del campo `vout`. El campo `vout` es una lista de diccionarios.

Ejemplo Dado el bloque siguiente:


```
{
    "hash": "0f84abb78891a4b9e8bc9637ec5fb8b4962c7fe46092fae99e9d69373bf7812a",
    "confirmations": 1,
    "strippedsize": 216,
    "size": 252,"weight": 900,
    "height": 6126,
    "version": 536870912,
    "versionHex": "20000000",
    "merkleroot": "254d5cc8d2b1889a2cb45f7e3dca8ed53a3fcfa32e8b9eac5f68c4f09e7af7bd",
    "tx": [
    "254d5cc8d2b1889a2cb45f7e3dca8ed53a3fcfa32e8b9eac5f68c4f09e7af7bd"
    ],
    "time": 1590830080,
    "mediantime": 1590170439,
    "nonce": 1,
    "bits": "207fffff",
    "difficulty": 4.656542373906925e-10,
    "chainwork": "00000000000000000000000000000000000000000000000000000000000002fde",
    "nTx": 1,
    "previousblockhash":
    "553ace55253369ed716ae9864ad307f1368285974762576e54441b255842f462"
}
```


Podéis ver que su tamaño es `252`, que fué creado el `1590830080` (*unix timestamp*) y contiene una única transacción que tiene como identificador `254d5cc8d2b1889a2cb45f7e3dca8ed53a3fcfa32e8b9eac5f68c4f09e7af7bd`.

Si buscamos esta misma transacción en el fichero de transacciones:


```
{
    "txid": "254d5cc8d2b1889a2cb45f7e3dca8ed53a3fcfa32e8b9eac5f68c4f09e7af7bd",
    "hash": "a8e125eb6d7ab883177d8ab228a3d09c1733d1ca49b7b2dff4b057eeb80ff9be",
    "version": 2,
    "size": 171,
    "vsize": 144,
    "weight": 576,
    "locktime": 0,
    "vin": [
        {
        "coinbase": "02ee170101",
        "sequence": 4294967295
        }
    ],
    "vout": [
            {
            "value": 12.00000000,
            "n": 0,
            "scriptPubKey": {
                "asm": "OP_HASH160 cd5b833dd43bc60b8c28c4065af670f283a203ff OP_EQUAL",
                "hex": "a914cd5b833dd43bc60b8c28c4065af670f283a203ff87",
                "reqSigs": 1,
                "type": "scripthash",
                "addresses": [
                "2NBy4928yJakYBFQuXxXBwXjsLCRWgzyiGm"
                ]
            }
        },
        {
        "value": 5.00000000,
        "n": 1,
        "scriptPubKey": {
                "asm": "OP_HASH160 cd5b833dd43bc60b8c28c4065af670f283a203ff OP_EQUAL",
                "hex": "a914cd5b833dd43bc60b8c28c4065af670f283a203ff87",
                "reqSigs": 1,
                "type": "scripthash",
                "addresses": [
                "2NBy4928yJakYBFQuXxXBwXjsLCRWgzyiGm"
                ]
            }
        }
    ],
    "hex":
    "020000000001010000000000000000000000000000000000000000000000000000000000000000
    ffffffff0502ee170101ffffffff020000000000000017a914cd5b833dd43bc60b8c28c4065af670f283a
    203ff8700000000000000000266a24aa21a9ede2f61c3f71d1defd3fa999dfa36953755c69068979996
```

```
    2b48bebd836974e8cf9012000000000000000000000000000000000000000000000000000000000000
    0000000000000000",
    "blockhash": "0f84abb78891a4b9e8bc9637ec5fb8b4962c7fe46092fae99e9d69373bf7812a",
    "confirmations": 1,
    "time": 1590830080,
    "blocktime": 1590830080
}
```


Podemos ver que su identificador (`txid`) se corresponde con el que habiamos visto en el bloque, este tiene un valor de `17.00000000` BTC, que corresponde a la suma de los campos `value` de la lista `vout`.

2.2 Criterios de corrección

Esta PEC se valorará siguiendo los critérios siguientes:

- **Funcionalidad** (5 puntos): Se valorará que el código implemente correctamente lo que se pide en el enunciado.
- **Documentación** (0.5 puntos): Todas las funciones del ejercicio de esta PEC tendrán que ser correctamente documentadas utilizando docstrings (en el formato que prefiráis).
- **Modularidad** (1 punto): Se valorará la modularidad del código (tanto la organización del código en ficheros como la creación de funciones).
- **Estilo** (0.5 puntos): El código tiene que seguir la guia de estilo de Python (PEP8), exceptuando los casos donde hacerlo complique la legibilidad del código.
- **Tests** (2 puntos): El código tiene que contener una o diversas *suites* de tests que permitan comprobar el buen funcionamiento de las funciones implementadas.
- **Requerimientos** (0.5 puntos): Tiene que haber un fichero de requerimientos que liste (solo) las librerías necesarias para ejecutar el código.
- **README** i **licencia** (0.5 puntos): Se valorará la creación de un fichero README, que presente el proyecto y explique como ejecutarlo, así como la inclusión de una licencia bajo la que se distribuya el código (podéis escoger la que queráis).

Importante:

Nota 1: De la misma manera que en la PEC2, los critérios transversales se valoraran de manera proporcional a la parte de la funcionalidad implementada.

Por ejemplo, si el código solo implementa la mitad de la funcionalidad que se pide, y la documetación de esta parte esta perfecta, entonces la puntuación correspondiente a la parte de documentación sería de 0.25.

Nota 2: Es imprescindible que el paquete que se entregue se ejecute correctamente en la máquina virtual, y que el fichero de README que incluyáis explique claramente como se tiene que ejecutar vuestro código pare generar las gráficas resultantes del análisis.

Nota 3: Entregad el paquete como un único archivo .zip en el Registro de Evaluación Continua.