Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 3

IIC2026 – Visualización de información
Entrega: martes 19 de noviembre de 2019, 19:59

Esta tarea pretende profundizar los conceptos de visualización y **D3.js** aprendidos durante las últimas semanas en clases y ayudantías. En especial abordaremos la visualización de un *grafo*. Esta tarea es **individual** y se entrega en su repositorio creado a partir de la siguiente **invitación**.

1. Implementación propia de Gephi [5 puntos]

1.1. Contexto

Dado que durante la actividad relacionada a visualización de Networks utilizando la herramienta Gephi hubieron problemas en algunos sistemas operativos, has tomado la gran misión y desafío de programar tu propia versión del programa[1], usando tu librería de *JavaScript* favorita: D3, de manera de que no importa desde dónde se utilice el programa, no habrán problemas para utilizarlo. Es de gran interés no solo poder visualizar la red (o grafo), sino que además poder representar correctamente distintas métricas intrínsecas a esta.

1.2. Formato

Los grafos a visualizar vendran en formato json, el cual tendrá la siguiente estructura:

```
{
    "nodes": [
        {
            "id": int
        },
        {
            "id": int
        },
            ...
    ],
    "links": [
        {
```

[1] La versión Alpha

```
            "target": int,
            "source": int
        },
        {
            "target": int,
            "source": int
        },
        ...
    ]
}
```

Dónde cada elemento de `"nodes"` corresponde a un nodo que solo tiene el atributo `id`, y cada elemento de `"links"` corresponde a una arista del grafo, la cual tiene un `source` y `target` que corresponden a `id` de los respectivos nodos.

1.3. Carga de Datasets

El programa debe ser capaz de cargar cualquier *dataset* que cumpla con este formato, para lograr esto, se deberá implementar un form en HTML que permita ingresar el `url` dónde se encuentra el json, y cargarlo a su visualización.

Para que puedan probar efectivamente su visualización, hay 3 tests ya disponibles para que puedan probar su código, con distinto número de nodos:

1. **20 Nodos**
2. **100 Nodos**
3. **1000 Nodos**

1.4. Implementación

Lo que se les solicita, en resumen, es lograr representar los grafos de cada dataset. Cabe mencionar que los nodos de este grafo no necesariamente están ordenados, pueden quedar superpuestos, con links atravesando uno a otro, etc.

Se les solicita, además, que su visualización cumpla con las siguientes características e interacciones:

1.4.1. Drag n' Drop

Se puede arrastrar un nodo, y la estructura del grafo debe ajustarse correctamente a la configuración con la que queda el grafo. Para esto debe utilizar el módulo de D3: **Force Directed Graph** para que se ajuste correctamente el grafo. Adicionalmente, **debe mantener el grafo dentro de un marco que los nodos no podrán atravesar**. El tamaño del marco queda a su criterio.

1.4.2. Zoom

Se debe poder hacer zoom sobre el gráfico, y este zoom debe cumplir con las siguientes condiciones:

1. En caso de que una parte de un nodo se salga de la vista actual (mucho zoom), entonces se deja de ver esa parte del nodo.
2. Es posible recorrer el grafo cuando está con zoom, y se debe representar la vista actual dentro del marco en un minimapa. En otras palabras, implementar el *idiom* visto en la Clase 11 de Facet & embed llamado *bird's-eye maps*.
3. Debe haber un rango máximo y mínimo de zoom a realizar en el grafo, es decir, no puedo hacer *zoom in* infinitamente o lo mismo con *zoom out*.

1.4.3. Representación de Métricas

Dado que queremos obtener información extra de nuestros grafos a través de nuestra visualización, queremos poder calcular métricas de **centralidad**, y poder representarlas correctamente en nuestra página con algún canal de los nodos (por ejemplo, con su radio). El canal que elijan será evaluado en cuanto a su efectividad para las tareas identificadas.

Las métricas que solicitamos que visualicen, en este caso, corresponden a las siguientes medidas de centralidad:

1. De Grado («degree centrality»)
2. De Cercanía («closeness centrality»)
3. De Intermediación («betweenness centrality»)
4. De Excentricidad («Eccentricity centrality»).

Para calcular estas, recomendamos hacer uso de la **librería** provista en el Syllabus del curso, la cual les permite obtener fácilmente esas estadísticas para cada nodo.

2. Presentación de resultados [1 puntos]

Para esta parte, se permitirán tres formatos de entrega:

2.1. Formato Clásico: HTML + JS + CSS

2.1.1. Informe

Debe crear un informe donde presente los resultados obtenidos en esta tarea. Este debe ser un documento **HTML** que contenga lo siguiente

1. Todas las visualizaciones solicitadas anteriormente
2. Una breve explicación de las decisiones de diseño (visual encoding[2]) tomadas a lo largo del desarrollo de su tarea

Recuerde hacer uso de los diferentes *tags* que posee **HTML** para mostrar la información de forma ordenada y amigable para el lector. Se espera que al menos el documento contenga lo siguiente:

- Contenido centrado (textos e imágenes) y texto justificado.

[2] *How según framework de Tamara Munzner*

- Cambios adicionales a un archivo **CSS** correspondiente, que enriquezcan la presentación de su informe.

No olvide que este documento será su informe de la tarea, por lo que adicionalmente se evaluará que cumpla con una correcta redacción y ortografía.

IMPORTANTE: No se permitirá correr un localhost para abrir los datasets (`python -m http.server 8000`), en su lugar deben cargar los datos desde los links entregados directamente con D3. Pueden revisar la **ayudantía 6** para ver como lograr ésto.

Además, puede incluir un archivo `README.md`, pero solo se evaluará como informe lo que está contenido en su **HTML**.

2.1.2. Formato

En esta tarea debe respetar los siguientes formatos:

- Entregar *scripts* escritos en **JavaScript** en archivos separados al código **HTML**. Todo el código **JavaScript** que utilice para esta visualización, a excepción de la librería provista, debe respetar el estandar **ES6**
- Debe utilizar la versión 5 de **D3.js** y programar con un paradigma **declarativo**. Por ejemplo, no se espera ver un `for` para hacer agregar cada elemento **SVG**, sino que utilizar el *data-join* visto en ayudantías para agregar elementos.

2.1.3. Entregables

Usted debe entregar su tarea en el repositorio que se le crea automáticamente al ingresar al **enlace** señalado anteriormente. Los entregables que se esperan para éste formato de tarea son:

- **Presentación resultados**: un informe en formato **HTML** que detalle lo realizado y logrado en las anteriores secciones y un archivo **CSS** que enriquezca su presentación.
- `scripts.js`: *scripts* (Pueden ser varios) **JavaScript** que haga uso de **D3.js** para generar las visualizaciones solicitadas (en este caso, el grafo).
- `README.md`: un archivo en formato **Markdown** en caso de detallar alguna instrucción previa para visualizar la tarea. Esto es opcional.

2.2. Formato ObservableHQ

2.2.1. Informe

Debe crear un informe donde presente los resultados obtenidos en esta tarea en ObservableHQ. Este, al igual que en el otro formato, debe ser un documento que contenga lo siguiente

1. Todas las visualizaciones solicitadas anteriormente.

2. Una breve explicación de las decisiones de diseño (visual encoding[3]) tomadas a lo largo del desarrollo de su tarea. Esta parte debe ser escrita con código **HTML** y no código **Markdown**.

Se espera, además, que al menos el documento contenga lo siguiente:

- Contenido centrado (textos e imágenes) y texto justificado.

No olvide que este documento será su informe de la tarea, por lo que adicionalmente se evaluará que cumpla con una correcta redacción y ortografía.

2.2.2. Formato

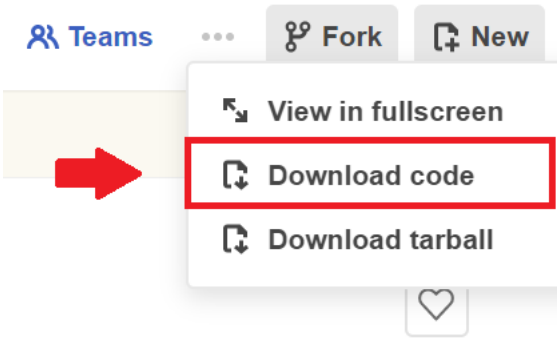
En esta tarea debe respetar los siguientes formatos:

- Debe utilizar la versión 5 de **D3.js** y programar con un paradigma **declarativo**. Por ejemplo, no se espera ver un `for` para hacer agregar cada elemento **SVG**, sino que utilizar el *data-join* visto en ayudantías para agregar elementos.
- Todo el código **JavaScript** que utilice para esta visualización, a excepción de la librería provista, debe respetar el estandar **ES6**

2.2.3. Entregables

Usted debe entregar su tarea en el repositorio que se le crea automáticamente al ingresar al **enlace** señalado anteriormente. Los entregables que se esperan para éste formato de tarea son:

- Código de su documento realizado en Observable, el cual pueden descargar en esta parte del documento:



- `README.md`: un archivo en formato **Markdown** con el *link* a su documento Observable. No olvides oprimir en *Enable link sharing* para que el link sea accesible por los ayudantes. Recuerde seguir los pasos de la **siguiente issue.**

  Importante: Cuando los ayudantes corrijan la tarea, ellos volverán a bajar el código con el botón *Download code* y verán las diferencias entre dicho código y el entregado por ustedes en el repositorio. Si se detecta alguna diferencia entre ambos códigos,

[3] *How según framework de Tamara Munzner*

su tarea será evaluada con nota mínima y no podrá apelar. Por lo tanto, luego de entregar la tarea, **no oprima en *Publish***.

3. Bonus (+ 0.5 puntos) CanvasJS

Se otorgará un bonus de 5 décimas a quienes integren **CanvasJS** a su entrega de modo que sea capaz de procesar y visualizar una mayor cantidad de nodos que **D3.js** no es capaz de visualizar por si solo. Una cantidad grande de ejemplo es de 5000 nodos.

Links de utilidad para esta parte pueden ser **este *post*** y **esta visualización**.

Imporante: El bonus de 5 décimas es sobre la nota final, y se aplicará **solo si se obtiene una nota igual o mayor a 5.0 si es que no se considera el bonus**.

4. Política de atraso

En la eventualidad de entregar pasada la fecha de entrega, se aplicará un **descuento** a la nota final obtenida en su tarea.

De haber atraso, el descuento comienza desde las 5 décimas. El descuento aumenta linealmente hasta las 24 horas posteriores del plazo inicial hasta un total de 20 décimas. Pasado un día del plazo inicial, el descuento es de 70 décimas, es decir, nota final **1**. La figura 3 muestra la función de descuento en función a las horas de atraso.

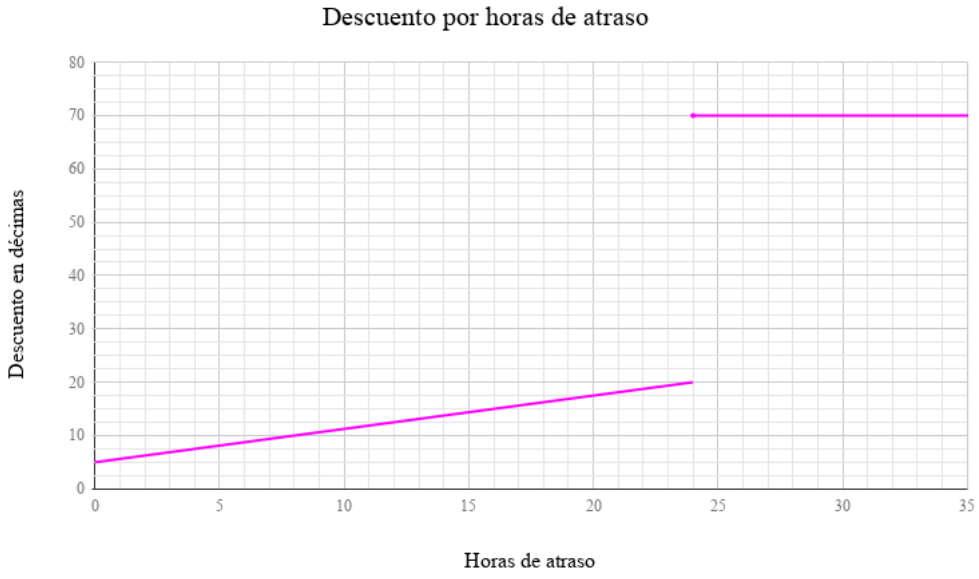


Figura 1: Descuento en nota según horas de atraso.