## ANEXO 1

```
Declare
        ttable text;
        ncols integer;
        cstart integer;
        cname text;
Begin
        cstart = 7; --start of the properties of the phenomenon
        ttable = 'discretedattributes';
        select count(*) into ncols from information_schema.columns where
table_name = ttable;
        raise notice '--SCRIPT TO ELIMINATE SPACES--------------------------------------
-';
        FOR cid IN 2..ncols LOOP
                select column_name into cname from information_schema.columns
where table_name = ttable and ordinal_position = cid;
                raise notice $$update % set % = replace(%,'
','');$$,ttable,cname,cname;
        END LOOP;
        raise notice '--SCRIPT TO PUT PREFIX--------------------------------------------';
        FOR cid IN cstart..ncols LOOP
                select column_name into cname from information_schema.columns
where table_name = ttable and ordinal_position = cid;
                raise notice $$update % set % =
'%_'||%;$$,ttable,cname,cname,cname;
        END LOOP;
End;
```

## ANEXO 2

```
DECLARE
        ezona varchar;
        etiempo varchar;
        elatitud double precision;
        elongitud double precision;
        czona varchar;
```

```
          ctiempo varchar;
          clatitud double precision;
          clongitud double precision;
          flag integer := 1;
          cadena varchar := '';
          registro record;
BEGIN
          delete from listofsequences;
          FOR registro IN select region, river, year, latitude, longitude, temp, ph, ss, bod
from discretedattributes order by region,river,year LOOP
                    IF flag = 1 THEN
                              ezona := registro.region||'_'||registro.river;
                              etiempo := registro.year;
                              elatitud := registro.latitude;
                              elongitud := registro.longitude;
                              flag := 0;
                    END IF;
                    czona := registro.region||'_'||registro.river;
                    ctiempo := registro.year;
                    IF czona = ezona THEN
                              IF ctiempo = etiempo THEN
                                        cadena := cadena || registro.temp || ' ' || registro.ph || ' '
|| registro.ss || ' ' || registro.bod || ' ';
                              ELSE
                                        cadena := cadena || '-1 ';
                                        cadena := cadena || registro.temp || ' ' || registro.ph || ' '
|| registro.ss || ' ' || registro.bod || ' ';

                              END IF;
                    ELSE
                              cadena := cadena || '-1 ' || '-2';
--                            raise notice 'z:%  s:%',ezona, cadena;
                              insert                                                  into
listofsequences(river_name,sequences,latitude,longitude,the_geom)             values
(ezona,cadena, elatitud, elongitud, ST_SetSRID (ST_MakePoint (elongitud , elatitud)
, 4326));
                              cadena := '';
                              cadena := cadena || registro.temp || ' ' || registro.ph || ' ' ||
registro.ss || ' ' || registro.bod || ' ';
                    END IF;
                    ezona := czona;
                    etiempo := ctiempo;
                    elatitud := registro.latitude;
                    elongitud := registro.longitude;
          END LOOP;
          cadena := cadena || '-1 ' || '-2';
          insert into
listofsequences(river_name,sequences,latitude,longitude,the_geom) values
(ezona,cadena, elatitud, elongitud,ST_SetSRID (ST_MakePoint (elongitud , elatitud)
, 4326)); --inserta las secuencias de la ultima zona
END;
```

**ANEXO 3**

**Programa para la creación de bases de datos de secuencias**

```
   public static String NextRandomData(int property){
      Random rnd = new Random();
      String c = "default";
      switch(property){
         case 1:
            c = "temp_" + (char)(rnd.nextInt(4) + 'a');
            break;
         case 2:
            c = "ph_" + (char)(rnd.nextInt(4) + 'e');
            break;
         case 3:
            c = "ss_" + (char)(rnd.nextInt(4) + 'i');
            break;
         case 4:
            c = "bod_" + (char)(rnd.nextInt(4) + 'm');
            break;
      }
      return c;
   }
```

```
    public static int NextRandomProperty(){
        Random rnd = new Random();
        return rnd.nextInt(4)+1;
    }
    public static int NextRandomNoItems(){
        Random rnd = new Random();
        return rnd.nextInt(4)+1;
    }

public static void buildSequences(){
        ArrayList<String> paths = new ArrayList<>();
        paths.add("C:\\Users\\Public\\sequencesTESTING0.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING1.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING2.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING3.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING4.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING5.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING6.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING7.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING8.txt");
        paths.add("C:\\Users\\Public\\sequencesTESTING9.txt");
        for(int a=0; a<paths.size(); a++){
            File file = new File(paths.get(a));
            try{
            FileWriter fw = new FileWriter(file.getAbsoluteFile());
            BufferedWriter bw = new BufferedWriter(fw);

            for(int i = 0; i<21; i++){
                for(int j = 0; j<34; j++){
                    int nitems = NextRandomNoItems();

                    ArrayList<Integer> possibleitems = new ArrayList<Integer>();
                    for(int z = 1; z<= 4; z++) possibleitems.add(new Integer(z));
    //                      for(int z = 1; z<= 4; z++) System.out.println(possibleitems.get(z-
1)+"chau");
                    for(int k = 0; k<nitems; k++){
                        int p = NextRandomProperty();
                        boolean encontrado = false;
                        int pos = 0;
                        for(int z = 1; z<= possibleitems.size(); z++){
                            if(possibleitems.get(z-1).intValue()==p){
                                encontrado = true;
                                pos = z-1;
                                break;
                            }
                        }
                        if(encontrado){
                            bw.write(NextRandomData(p)+" ");
                            possibleitems.remove(pos);
                        }
                    }
                    bw.write("-1 ");
                }
                bw.write("-2\n");
            }
            bw.close();
            }
            catch(Exception e){
                System.out.println("ERROR");
            }
        }
    }
}
```

**Ejemplo de base de datos de secuencias para la interpretación cuantitativa**

| No de Secuencia | Secuencia |
|---|---|
| 1 | ss_l -1 bod_n ss_k ph_g -1 bod_m temp_a ph_g -1 ss_l temp_a -1 temp_d -1 temp_b ss_i -1 ss_l -1 temp_d bod_n ss_k -1 ss_k temp_c -1 bod_o ph_h temp_d -1 bod_n ph_e temp_b -1 ph_f bod_p -1 ph_h temp_d ss_i -1 ss_i bod_p -1 ss_j -1 temp_d -1 bod_m ss_k -1 temp_a -1 temp_d -1 temp_d -1 ph_f -1 ph_e temp_c bod_n -1 bod_o temp_a |

| | |
|---|---|
| | -1 temp_a ph_h bod_p ss_i -1 temp_c bod_m -1 ph_e temp_a ss_l -1 ss_k temp_a -1 bod_p temp_b -1 ss_l ph_g -1 bod_o temp_b ss_k -1 bod_n -1 temp_d -1 ph_f -1 ph_g temp_a -1 -2 |
| 2 | ss_j ph_e bod_p -1 bod_n ss_k -1 bod_n ss_k -1 bod_p -1 ss_i -1 bod_n temp_d ss_k -1 ss_i ph_f -1 bod_o ss_i -1 temp_c -1 ss_k temp_a ph_e -1 bod_p -1 bod_n -1 ph_f temp_c ss_j -1 ph_g -1 bod_m ss_k temp_c ph_f -1 ph_g -1 ph_f bod_m -1 temp_c ph_f -1 temp_b -1 temp_c -1 temp_d ph_h -1 temp_a ss_j -1 bod_n -1 bod_n ss_l -1 ph_g temp_c -1 ss_j -1 ss_k -1 ss_i ph_f -1 temp_c -1 ph_f temp_a bod_p -1 bod_m -1 bod_p -1 temp_d ph_g -1 bod_n ss_i -1 -2 |
| 3 | bod_p ph_h -1 ph_f temp_c bod_p -1 bod_n temp_c -1 bod_o ss_k -1 bod_o temp_c ss_l -1 ph_f bod_m -1 ph_h bod_m -1 ss_j bod_p temp_d -1 temp_a -1 ph_f temp_d -1 ph_e ss_j temp_c -1 bod_p temp_d ss_j -1 bod_o ph_e -1 bod_o ph_e ss_i temp_d -1 ph_g bod_o -1 bod_p temp_a ss_i -1 temp_a ss_l bod_m -1 temp_c ph_h -1 bod_m -1 bod_m temp_a -1 ph_f -1 ss_l -1 temp_d bod_n ss_i -1 temp_a -1 temp_d -1 bod_o ph_e temp_d -1 temp_a -1 temp_c bod_m ph_e -1 ph_h ss_j temp_c -1 ph_h temp_b -1 temp_c bod_m -1 temp_d ph_e ss_i -1 ph_h -1 ph_e temp_a -1 -2 |
| … | … |
| 21 | ss_l bod_m -1 ss_j temp_c -1 temp_a bod_n ss_j -1 bod_m -1 bod_o ss_k ph_f -1 ss_i -1 ss_l -1 temp_c bod_m ph_f ss_i -1 bod_m -1 ph_e temp_d -1 bod_p -1 bod_m -1 bod_n ss_j -1 bod_o ss_l ph_f -1 bod_o -1 ph_g -1 ph_f ss_j -1 ph_g bod_o -1 temp_b ph_e bod_o ss_k -1 bod_m ss_j temp_b -1 temp_d ss_l -1 temp_a ph_h bod_o -1 bod_m ph_g temp_d -1 temp_d ph_h -1 bod_n ss_k ph_e -1 ph_e bod_p -1 ss_l -1 ss_j -1 temp_c ph_e ss_k bod_p -1 ph_f bod_p -1 bod_n ph_f temp_d -1 bod_o temp_a -1 temp_c ss_j -1 bod_m ss_l -1 -2 |

## ANEXO 4

```
Main {
        String inputPath = "C:\\Users\\Public\\output.txt";
        BestSequenceSearcher bssearcher = new BestSequenceSearcher();
        ArrayList<Sequence> frecuent_sequences =
        bssearcher.readFrecuentSequences(inputPath);
```

```
ArrayList<Sequence> best_sequences = bssearcher.findBest10FrecuentSequences(frecuent_sequences);
```

**//URL de archivos KML. Son cargados a las diez mejores secuencias luego de haber encontrado los ríos donde actúan (Anexo 4)**

```
ArrayList<String> urlKmls = new ArrayList<>();
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.keh3XjwKf3-I");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.koagnZN09vD4");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kJr36YYK8QR0");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kHCBYiL5FaIo");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.klrWQZP0m13o");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kfTS8oZcxb5M");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kjdU-9kkaVmM");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kO3DMQm-ODb8");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kTECBlgiSlNU");
urlKmls.add("https://www.google.com/maps/d/embed?mid=zajizRXubkHo.k7hsz2oYQSsY");
for(Sequence bestseq : best_sequences){
        bestseq.urlKml = urlKmls.get(best_sequences.indexOf(bestseq));
}
```

**//Impresión a Json para su posterior lectura por la herramienta de visualización, como entrada a la tabla de secuencias**

```
PrintWriter writer = new PrintWriter("C:\\xampp\\htdocs\\thesiswebsite\\json\\best_sequences.json", "UTF-8");
Gson gson = new GsonBuilder().disableHtmlEscaping().create(); //json will consider < >, since it considers that as html by default
```

```
            String json = gson.toJson(best_sequences);
            writer.println(json);
            writer.close();
}
```

**Clase BestSequenceSearcher** {

```
    public static final int N_filterChangedItems = 100;
    public static final int N_filterNoItems = 50;
    public static final int N_filterNoItemsets = 10;
     public BestSequenceSearcher(){

    }
        public ArrayList<Sequence> readFrecuentSequences(String path) throws
IOException {
        ArrayList<Sequence> frequent_sequences = new ArrayList<>();
        String thisLine; // variable to read each line.
        BufferedReader myInput = null;
        try {
            FileInputStream fin = new FileInputStream(new File(path));
            myInput = new BufferedReader(new InputStreamReader(fin));
            // for each line until the end of the file, read the frequent_sequence
            while ((thisLine = myInput.readLine()) != null) {
                Sequence frequent_sequence = new Sequence();
                frequent_sequence.support = getSupport(thisLine);
                frequent_sequence.itemsets = getItemsets(thisLine);
                frequent_sequences.add(frequent_sequence);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (myInput != null) {
                myInput.close();
            }
        }
        return frequent_sequences;
    }
        public String getSupport(String thisLine){
```

```
        String support = null;
        ArrayList<String> str_itemsets = new ArrayList<>();
        str_itemsets.addAll(Arrays.asList(thisLine.split(" -1 ")));
        support = str_itemsets.get(str_itemsets.size() - 1);
        support = support.replaceFirst(" #SUP: ", "");
        return support;
    }
        public ArrayList<Itemset> getItemsets(String seq) {
        ArrayList<Itemset> itemsets = new ArrayList<Itemset>();
        ArrayList<String> str_itemsets = new ArrayList<>();
        str_itemsets.addAll(Arrays.asList(seq.split(" -1 ")));
        //remove last itemset since it's -2
        str_itemsets.remove(str_itemsets.size() - 1);
        //for each itemset in the sequence, which is a list of itemsets
        for (String str_itemset : str_itemsets) {
            Itemset itemset = new Itemset();
            ArrayList<String> str_items = new ArrayList<>();
            str_items.addAll(Arrays.asList(str_itemset.split(" ")));
            //for each item in the itemset, which is a list of items
            for (String str_item : str_items) {
                Item item = new Item();
                item.property = str_item;
                itemset.items.add(item);
            }
            itemsets.add(itemset);
        }
        return itemsets;
    }

    public              ArrayList<Sequence>              findBest10FrecuentSequences
(ArrayList<Sequence> frequent_sequences){
        ArrayList<Sequence> best_sequences = new ArrayList<Sequence>();
        best_sequences                                                              =
getBestSequencesBasedOnMostChangedItems(frequent_sequences);
        best_sequences = getBestSequencesBasedOnNumberItems(best_sequences);
//greatest number items, 1st filter
```

```
        best_sequences                                                              =
getBestSequencesBasedOnNumberItemsets(best_sequences);   //shortest   number
itemsets, 2nd filter
        return best_sequences;
    }

    public                                                      ArrayList<Sequence>
getBestSequencesBasedOnMostChangedItems(ArrayList<Sequence> sequences){
        ArrayList<Sequence> sq = new ArrayList<Sequence>();
        ArrayList<Property> properties = getPropertiesInSequences(sequences);

Collections.sort(getBestSequencesWithNoChangedItems(properties,sequences),
new NumberChangedItemsComparator());
        for(int i=0;i<N_filterChangedItems;i++){ //N is define as 100
            sq.add(sequences.get(i));
        }
        return sq;
    }

    public                                                      ArrayList<Sequence>
getBestSequencesWithNoChangedItems(ArrayList<Property>                properties,
ArrayList<Sequence> sequences){
        for(Sequence sequence: sequences){
            for(Property property: properties){
                boolean flag = true;
                for(Itemset itemset: sequence.itemsets){
                    for(Item item: itemset.items){
                        ArrayList<String> str_items = new ArrayList<>();
                        str_items.addAll(Arrays.asList(item.property.split("_")));
                        String str_p = str_items.get(0);
                        if(str_p.compareTo(property.name_property)==0){
                            if(flag){ //executes only once, to assign lastvalue the first time
                                property.lastproperty = item.property;
                                property.lastItem = item;
                                property.lastItem.sufferedchange=false;
                                flag = false;
                            }
                            if(property.lastproperty.compareTo(item.property)!=0){
                                property.lastproperty = item.property;
                                property.lastItem.sufferedchange = true; //last item
                                item.sufferedchange = true; //current item
                                property.n_changes += 1;
                            }
                            property.lastItem = item;
                            break;
                        }
                    }
                }
            }
            //Add all the changes into changed_properties (belongs to sequence)
            for(Property property:properties){
                sequence.changed_properties += property.n_changes;
            }
            //Reset property n_changes values
            for(Property property:properties){
                property.n_changes = 0;
            }
        }
        return sequences;
    }

    public    ArrayList<Property>    getPropertiesInSequences(ArrayList<Sequence>
sequences){
        ArrayList<Property> properties = new ArrayList<Property>();
        for(Sequence sequence: sequences){
            for(Itemset itemset: sequence.itemsets){
                for(Item item: itemset.items){
                    ArrayList<String> str_items = new ArrayList<>();
                    str_items.addAll(Arrays.asList(item.property.split("_")));
                    String str_p = str_items.get(0);
                    if(properties.size()==0){
                        Property property = new Property();
                        property.name_property = str_p;
                        properties.add(property);
                    }
                    else{
                        if(!foundProperty(str_p,properties)){
                            Property property = new Property();
                            property.name_property = str_p;
                            properties.add(property);
                        }
                    }
                }
            }
        }
        return properties;
    }

    public boolean foundProperty(String p,ArrayList<Property> properties){
        boolean found = true;
        for(Property property: properties){
            if(property.name_property.compareTo(p)==0){
                return found;
            }
        }
        return !found;
    }

    public                                                    ArrayList<Sequence>
getBestSequencesBasedOnNumberItems(ArrayList<Sequence> sequences){
        ArrayList<Sequence> sq = new ArrayList<Sequence>();
        Collections.sort(sequences, new NumberItemsComparator());
        for(int i=0;i<N_filterNoItems;i++){ //N is define as 50
            sq.add(sequences.get(i));
        }
        return sq;
    }

    public                                                    ArrayList<Sequence>
getBestSequencesBasedOnNumberItemsets(ArrayList<Sequence> sequences){
        ArrayList<Sequence> sq = new ArrayList<Sequence>();
        Collections.sort(sequences, new NumberItemsetsComparator());
        for(int i=0;i<N_filterNoItemsets;i++){ //N is define as 10
            sq.add(sequences.get(i));
        }
        sq = assignID(sq);
        return sq;
    }

    public ArrayList<Sequence> assignID(ArrayList<Sequence> sq){
        int n = 0;
        for(Sequence s : sq){
            s.id = Integer.toString(++n);
        }
        return sq;
    }

    class NumberItemsComparator implements Comparator<Sequence> {
    @Override
    public int compare(Sequence a, Sequence b) { //descending
        return a.getNumberItems() < b.getNumberItems() ? 1 : a.getNumberItems() ==
b.getNumberItems() ? 0 : -1;
    }
    }

    class NumberItemsetsComparator implements Comparator<Sequence> {
    @Override
    public int compare(Sequence a, Sequence b) { //ascending
        return    a.itemsets.size()    <    b.itemsets.size()    ?    -1    :    a.itemsets.size()    ==
b.itemsets.size() ? 0 : 1;
    }
    }

    class NumberChangedItemsComparator implements Comparator<Sequence> {
    @Override
    public int compare(Sequence a, Sequence b) { //descending
        return a.changed_properties < b.changed_properties ? 1 : a.changed_properties
== b.changed_properties ? 0 : -1;
```

```
        }
    }

    class Property {
        String name_property;
        int n_changes;
        String lastproperty;
        Item lastItem;

        public Property(){
            name_property = "";
            n_changes = 0;
            lastproperty = "";
        }
    }
}
```

**ANEXO 5**

```
Main{
            String inputPath = "C:\\Users\\Public\\best_sequences.txt";
            RiverSearcher searcher = new RiverSearcher();
            ArrayList<Sequence> db_sequences = searcher.readSequences();
            ArrayList<Sequence>                    best_sequences                    =
            searcher.readBestSequences(inputPath);
            best_sequences                                                           =
            searcher.findRiversToBestSequences(best_sequences,db_sequences);
}

Clase RiverSearcher {

    public RiverSearcher() {

    }

    public ArrayList<Sequence> readSequences() {
        ArrayList<Sequence> sequences = new ArrayList<>();
        Database.startConnection();
        ArrayList<ArrayList<String>> str_sequences = Database.getSequences();
        boolean f = true;
        for (ArrayList<String> str_seq : str_sequences) {
            Sequence sequence = new Sequence();
            sequence.id = str_seq.get(0);
            sequence.river = str_seq.get(1);
            sequence.itemsets = getItemsets(str_seq.get(2));
            sequence.latitude = Double.parseDouble(str_seq.get(3));
            sequence.longitude = Double.parseDouble(str_seq.get(4));
            sequences.add(sequence);
        }
        Database.closeConnection();
        return sequences;
    }

    public ArrayList<Itemset> getItemsets(String seq) {
        ArrayList<Itemset> itemsets = new ArrayList<Itemset>();
        ArrayList<String> str_itemsets = new ArrayList<>();
        str_itemsets.addAll(Arrays.asList(seq.split(" -1 ")));
        //remove last itemset since it's -2
        str_itemsets.remove(str_itemsets.size() - 1);
        //for each itemset in the sequence, which is a list of itemsets
        for (String str_itemset : str_itemsets) {
            Itemset itemset = new Itemset();
            ArrayList<String> str_items = new ArrayList<>();
            str_items.addAll(Arrays.asList(str_itemset.split(" ")));
            //for each item in the itemset, which is a list of items
            for (String str_item : str_items) {
                Item item = new Item();
                item.property = str_item;
                itemset.items.add(item);
            }
            itemsets.add(itemset);
        }
        return itemsets;
    }
    public ArrayList<Sequence> readBestSequences(String path) throws IOException
{
        ArrayList<Sequence> best_sequences = new ArrayList<>();
        String thisLine; // variable to read each line.
        BufferedReader myInput = null;
        try {
            FileInputStream fin = new FileInputStream(new File(path));
            myInput = new BufferedReader(new InputStreamReader(fin));
            // for each line until the end of the file, read the best_sequence
            while ((thisLine = myInput.readLine()) != null) {
                Sequence best_sequence = new Sequence();
                best_sequence.support = getSupport(thisLine);
                best_sequence.itemsets = getItemsets(thisLine);
                best_sequences.add(best_sequence);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (myInput != null) {
                myInput.close();
            }
        }
        return best_sequences;
    }

    public String getSupport(String thisLine){
        String support = null;
        ArrayList<String> str_itemsets = new ArrayList<>();
        str_itemsets.addAll(Arrays.asList(thisLine.split(" -1 ")));
        support = str_itemsets.get(str_itemsets.size() - 1);
        support = support.replaceFirst(" #SUP: ", "");
        return support;
    }

    public  ArrayList<Sequence>  findRiversToBestSequences  (ArrayList<Sequence>
best_sequences,ArrayList<Sequence> db_sequences){
        for (Sequence best_sequence : best_sequences){
            for (Sequence db_sequence : db_sequences){
                if(frequentSequenceFound(best_sequence,db_sequence)){

best_sequence.riversFound.add(db_sequence.id+","+db_sequence.river+","+db_se
quence.latitude+","+db_sequence.longitude);
                }
            }

createTablesForRivers(best_sequences.indexOf(best_sequence)+1,best_sequence
);
        }
        return best_sequences;
    }

    public boolean frequentSequenceFound (Sequence best_sequence, Sequence
db_sequence){
        boolean sequenceFound = false;
        boolean itemsetFound = false;
        int lastPosFound = 0;
        for (Itemset frequent_itemset : best_sequence.itemsets){
            for (int i = lastPosFound; i<db_sequence.itemsets.size(); i++){

if(itemsetFound=frequentItemsetFound(frequent_itemset,db_sequence.itemsets.get(
i))){
                    if(frequent_itemset                                    ==
best_sequence.itemsets.get(best_sequence.itemsets.size()-1)){
                        sequenceFound = true;
                    }
                    else{
                        lastPosFound = i + 1; //optimization, the next frequent_itemset will be
searched from the last postion in db_itemsets where the previous frequent_itemset
was found
                    }
                    break;
                }
            }
            //if a frequent_itemset has NOT been found, so the frequent_sequent is not
going to be found. Return false
            if(!itemsetFound){
                break;
            }
            //if the last frequent_itemset has been found, so the best_sequence has been
found. No more search into db_itemsets. Return true
            if(sequenceFound){
                break;
            }
        }
        return sequenceFound;
    }

    public boolean frequentItemsetFound (Itemset frequent_itemset, Itemset
db_itemset){
        boolean itemsetFound = false;
        boolean itemFound = false;
        for (Item frequent_item : frequent_itemset.items){
            for (Item db_item : db_itemset.items){
                if(itemFound = compareItems(frequent_item.property,db_item.property)){

if(frequent_item.property.compareTo(frequent_itemset.items.get(frequent_itemset.it
ems.size()-1).property)==0){
                        itemsetFound = true;
                    }
                    break;
                }
            }
            //if a frequent_itemset has NOT been found, so the frequent_sequent is not
going to be found. Return false
            if(!itemFound){
                break;
            }
            //if the last frequent_itemset has been found, so the best_sequence has been
found. No more search into db_itemsets. Return true
            if(itemsetFound){
                break;
            }
        }
        return itemsetFound;
    }

    public boolean compareItems (String frequent_item, String db_item){
        boolean itemFound = false;
        if(frequent_item.compareTo(db_item)==0){
            itemFound = true;
        }
        return itemFound;
    }

    public void createTablesForRivers(int index,Sequence best_sequence){
        String table_name = "rivers_bestsequence_"+index;
        Database.startConnection();
        Database.createTable(table_name,"id","river_name","latitude","longitude");
        Database.insertValuesOnTable(table_name,best_sequence);
        Database.createColumnTheGeom(table_name);
        Database.closeConnection();
    }
}
```

**ANEXO 6**

```
public class Sequence {
    /*attributes for db sequence and attribute sequence*/
    public String id;
    public String river;
    public ArrayList<Itemset> itemsets = new ArrayList<Itemset>();
    public double latitude;
    public double longitude;
    /*attributes only for frequent/best sequence*/
    public String support;
    public ArrayList<String> riversFound = new ArrayList<String>();
    public String urlKml;
    public int changed_properties;

    public Sequence(){
        id = "0";
        river="defecto";
        support = "0";
        latitude = 0;
        longitude = 0;
        urlKml = "defecto";
        changed_properties = 0;
    }

    public void additemset(Itemset itemset){
        this.itemsets.add(itemset);
    }

    public int getNumberItems(){
        int n=0;
        for(Itemset itemset:itemsets){
            n += itemset.items.size();
        }
        return n;
    }

    public void printSequence(){
        for(Itemset itemset : itemsets){
            for(Item item : itemset.items){
                System.out.print(item.property+" ");
            }
            System.out.print("-1 ");
        }
        System.out.println(" #SUP: "+this.support);
    }

    public void printSequenceToFile(PrintWriter writer){
        try{
        String str = "";
```

```
            for(Itemset itemset : itemsets){
                for(Item item : itemset.items){
                    str += item.property+" ";
                }
                str += "-1 ";
            }
            str += " #SUP: " + this.support;
            writer.println(str);
            }
            catch(Exception e){
                System.out.println(e);
            }
        }

        public void printRiversFound(){
            System.out.print(riversFound.size()+"\t");
            for(String river : riversFound){
                System.out.print(river+"; ");
            }
            System.out.println();
        }

        public void printToJson(){
            Gson gson = new GsonBuilder().disableHtmlEscaping().create(); //json will
    consider < >, so it'll cnosider html
            String json = gson.toJson(this);
            System.out.println(json);
        }

    }
```

**public class Itemset {**

```
        public ArrayList<Item> items = new ArrayList<Item>();

        public Itemset(){
        }

        public void additem (Item item){
            this.items.add(item);
        }
    }

    public class Item {
        public String property = "default";
        public boolean sufferedchange = false;
    }

    public class Database {
        private static String database =
    "jdbc:postgresql://localhost:5432/uk_rivers_postgis";
        private static String user = "postgres";
        private static String password = "wayokparce";
        public static Connection connection = null;

        public static String getDatabase() {
            return database;
        }

        public static String getUser() {
            return user;
        }

        public static String getPassword() {
            return password;
        }

        public static void startConnection() {
            try {
                Class.forName("org.postgresql.Driver");
            } catch (ClassNotFoundException e) {
                System.out.println("Where is your PostgreSQL JDBC Driver? "
                        + "Include in your library path!");
                e.printStackTrace();
            }
            try {
                connection = DriverManager.getConnection(
                    Database.getDatabase(),
                    Database.getUser(),
                    Database.getPassword());
                if (connection == null) {
                    System.out.println("Failed to make connection!");
                }
            } catch (SQLException e) {
                System.out.println("Connection Failed! Check output console");
                e.printStackTrace();
            }
        }

        public static void closeConnection(){
            try {
                connection.close();
            }catch(SQLException e) {
                System.err.println("SQLException while closing connection. "+
    e.getMessage());
            }
        }

        public static ArrayList<ArrayList<String>> getSequences(){
            ArrayList<ArrayList<String>> str_sequences = new
    ArrayList<ArrayList<String>>();
            Statement stmt = null;
            ResultSet rs = null;
            try{
                // Creating Statement for query execution
                stmt = connection.createStatement();
                // creating Query String
                String query = "SELECT id,river_name,sequences,latitude,longitude \n" +
                               "FROM listofsequences\n";
                // excecuting query
                rs = stmt.executeQuery(query);
                while (rs.next()) {
                    ArrayList<String> str_sequence = new ArrayList<String>();
                    str_sequence.add(rs.getString("id"));
                    str_sequence.add(rs.getString("river_name"));
                    str_sequence.add(rs.getString("sequences"));
                    str_sequence.add(rs.getString("latitude"));
                    str_sequence.add(rs.getString("longitude"));
                    str_sequences.add(str_sequence);
                }
            }
            catch(SQLException e){
                System.out.println("Connection Failed! Check output console");
                e.printStackTrace();
            }
            return str_sequences;
        }

        public static void createTable(String table_name,String col1,String col2,String
    col3,String col4){
            Statement stmt = null;
            try{
                // Creating Statement
                stmt = connection.createStatement();
                // creating SQL for DROP TABLE IF EXISTS Statement
                String sqldrop = "DROP TABLE IF EXISTS " + table_name;
                stmt.executeUpdate(sqldrop);
                // creating SQL for CREATE TABLE Statement
                String sqlcreate = "CREATE TABLE " + table_name +
                            " (id serial PRIMARY KEY NOT NULL," +
                            " river_name TEXT, " +
                            " latitude double precision, " +
                            " longitude double precision) ";
                stmt.executeUpdate(sqlcreate);
                stmt.close();
            }
        catch(Exception e){
            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
        }
    }
    public static void insertValuesOnTable(String table_name, Sequence
best_sequence){
        Statement stmt = null;
        try{
            // Creating Statement
            stmt = connection.createStatement();
            for(String river : best_sequence.riversFound){
                String[] parts = river.split(",");
                parts[parts.length-1] = parts[parts.length-1].substring(0, parts[parts.length-
1].length()); //removing character ";" in the last part[], which is longitud
                // creating INSERT SQL Statement
                String sql = "INSERT INTO " + table_name + "
(river_name,latitude,longitude) " +
                        "VALUES ('" + parts[1] + "', " + parts[2] + ", " + parts[3] + ");";
//river_name, latitude, longitude
                stmt.executeUpdate(sql);
            }
            stmt.close();
        }
        catch(Exception e){
            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
        }
    }

    public static void createColumnTheGeom(String table_name){
        Statement stmt = null;
        try{
            // Creating Statement
            stmt = connection.createStatement();
            // creating ALTER TABLE SQL Statement
            String sqlalter = "ALTER TABLE " + table_name +
                        " ADD COLUMN the_geom geometry (POINT, 4326);";
            stmt.executeUpdate(sqlalter);
            // creating UPDATE SQL Statement
            String sqlupdate = "UPDATE " + table_name +
                        "  SET  the_geom  =  ST_SetSRID(ST_MakePoint(\"longitude\",
\"latitude\"), 4326);";
            stmt.executeUpdate(sqlupdate);
            stmt.close();
          }
          catch(Exception e){
            System.out.println("Connection Failed! Check output console");
            e.printStackTrace();
          }
      }
}
```

**ANEXO 7**

**CSS:**

```
body {
    background-image:                              url("http://rule-of-three.co.uk/wp-
content/themes/ruleofthree/images/blackheaderbg.jpg");
}

iframe {
    float: left;
    margin: 0 10px 0 0;
}

img {
            float: left;
}

#title{
            float:left;
}
```

```
#wrapper {
	margin: 20px;
}

#table{
	overflow-y: scroll;
	height: 484px;
	width: 48.9%;
}

/*Text styles*/

h1 { color: #c9d0d4; font-family: 'Arial', sans-serif; font-size: 46px; font-weight: 100; text-align: center; line-height: 50px; letter-spacing: 1px; padding: 0 0 20px; border-bottom: double #555; }

h2 { color: #bbc3c8; font-family: 'Arial', sans-serif; font-size: 16px; text-align: center; line-height: 26px; text-indent: 30px; margin: 0; margin: 20px; padding: 20px 0; border-top: double #555; }

p { color: #bbc3c8; font-family: 'Arial', sans-serif; font-size: 16px; text-align: justify; line-height: 26px; margin: 20px; padding: 20px 0;  }

a { color: #c64119; border-bottom: 1px solid #c64119; text-decoration: none; }



/* Component styles */
table {
	float: left;
    border-collapse: collapse;
    background: #fff;
}
td, th {
    padding: 0.75em 1.5em 1em;
    text-align: center;
    font-family: 'Arial', sans-serif;
}
            td.err {
                        background-color: #e992b9;
                        color: #fff;
                        font-size: 0.75em;
                        text-align: center;
                        line-height: 1;
            }
th {
    background-color: #31bc86;
    font-weight: bold;
    color: #fff;
    white-space: nowrap;
}
tbody th {
            background-color: #496C90;
}
tbody tr:nth-child(2n-1) {
    background-color: #f5f5f5;
    transition: all .125s ease-in-out;
}
tbody tr:hover {
    background-color: rgba(129,208,177,.3);
}

.foo {
    display: inline-block;
    width: 10px;
    height: 10px;
    margin: 5px 5px 0 5px;
    border-width: 1px;
    border-style: solid;
}
```

**JS:**

```
var   f1   =   "https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kxgJkEW-
Fh2k";

$(window).load(function() {
        $.ajax({
                type: 'GET',
                url: 'json/best_sequences.json',
                dataType: 'json',
                success: function(json) {
                        var cad = [];
                        var j = 0;
                        cad[0] = '<tr><th> ID </th>'+'<th> Secuencia Frecuente </th>' +
'<th> Soporte </th></tr>';
                        for (var i = 0; i < json.length; i++) {
                                cad[++j] = '<tr>';
                                cad[++j] = '<td>'+json[i].id+'</td>';
                           cad[++j] = '<td>'+buildSequence(json[i])+'</td>';
                           cad[++j] = '<td>'+json[i].support+'</td>';
                           cad[++j] = '</tr>';
                        }
                        $('#tb_sequences').html(cad.join(''));
                },
                error: function(data, status, error) {
             console.log(data);
             console.log(status + ' ' + error);
                }
        });
});

function buildSequence(object){
        var timeSeparator = '&#8594 ';
        var propertySeparator = ' ';
        var cad = '';
        for(var i = 0; i < object.itemsets.length; i++){
                for(var j = 0; j < object.itemsets[i].items.length; j++){
                        if(object.itemsets[i].items[j].sufferedchange){
                                var color;
```

```
                                        color = getColor(object.itemsets[i].items[j].property);
                                        cad += '<font color="'+color+'">';
                                        cad += object.itemsets[i].items[j].property;
                                        cad += '</font>'
                                }
                                else{
                                        cad += object.itemsets[i].items[j].property;
                                }
                                cad += propertySeparator;
                        }
                        if(i!=object.itemsets.length-1)
                                cad += timeSeparator;
                }
                return cad;
        }

        function getColor(item){
                var color;
                var object = item.split("_");
                switch(object[0]){
                        case "temp":
                                color = "4169E1";
                                break;
                        case "ss":
                                color = "00CD00";
                                break;
                        case "ph":
                                color = "FF6103";
                                break;
                        case "bod":
                                color = "8E388E";
                                break;
                }
                return color;
        }

        $('#tb_sequences').click(function(e) {
            $.ajax({
                        type: 'GET',
                        url: 'json/best_sequences.json',
                        dataType: 'json',
                        /*async: false,*/
                        success: function(json) {
                                    var id;
                                    id = getSequenceID(e);
                                    if(id!=0){
                                                foundJson = findSequence(id,json);
                                                var cad = '<iframe src="' +
                                                           foundJson.urlKml +
                                                          '" width="50%" height="480px"></iframe>';
                                                $('#map').html(cad);
                                                console.log(cad);
                                    }
                        },
                        error: function(data, status, error) {
                    console.log(data);
                    console.log(status + ' ' + error);
                        }
            });
});


function getSequenceID(e){
            e = e || window.event;
    var data = [];
    var target = e.target;
    while (target && target.nodeName !== "TR") {
        target = target.parentNode;
    }
    if (target) {
        var cells = target.getElementsByTagName("td");
        return cells[0].innerHTML;
    }
    else{
            alert('Error al buscar el ID de secuencia, seleccione una fila');
            return 0;
    }
}

function findSequence(id,json){
            for (var i = 0; i < json.length; i++) {
                        if(json[i].id == id){
                                    return json[i];
                        }
            }
            alert('Error al buscar json')
            return [];
}
```

**HTML:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>UK Rivers</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <script src="js/jquery-1.11.3.js"></script>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body>
    <header id="titulo">
      <img src="http://fisica.pucp.edu.pe/images/pucp_logo.jpg" width="220"
height="95" alt="Computer Hope">
      <h1>Extracción de patrones secuenciales para la caracterización de fenómenos
espacio-temporales</h1>
    </header>
    <section>
      <p>La siguiente tabla muestra las diez mejores secuencias frecuentes extraidas
de la base de datos del fenómeno de <strong><u>contaminación de ríos del Reino
Unido</u></strong>, luego de haber utilizado el <strong><font
color="00BFFF"><u>algoritmo de patrones secuenciales
```

```
PrefixSpan</u></font></strong> con <strong><font color="00BFFF"><u>soporte
minimal igual a 16</u></font></strong>. De esta manera, es posible ubicar en qué
ríos ocurrieron cada una de estas secuencias, para ello se debe seleccionar alguna
de las secuencias frecuentes mostradas. Las diez mejores secuencias son aquellas
que poseen la mayor cantidad de cambios en el tiempo.</p>
    </section>
    <section>
      <h2>Leyenda de características<br>
        <div class="foo" style="background-color:#4169E1;"></div>temp: Temperatura
        <div class="foo" style="background-color:#00CD00;"></div>ss: Restos Sólidos
        <div class="foo" style="background-color:#FF6103;"></div>ph: Nivel de Acidez
        <div class="foo" style="background-color:#8E388E;"></div>bod: Demanda de
Oxígeno Biológico</h2>
    </section>
    <section id="wrapper">
      <div id="map">
        <iframe
src="https://www.google.com/maps/d/embed?mid=zajizRXubkHo.kXuvwMSTyigw"
width="50%" height="480px"></iframe>
      </div>
      <div id= "table">
        <table id="tb_sequences">
          <th>ID</th>
          <th>Secuencias frecuentes</th>
          <th>Soporte</th>
        </table>
      </div>
    </section>
    <footer>
      <h2>Elaborado por : Rodrigo Ricardo Maldonado Cadenillas <br>
      Proyecto de fin de carrera (2015)</h2>
    </footer>
    <script src="js/script.js"></script>
  </body>
</html>
```