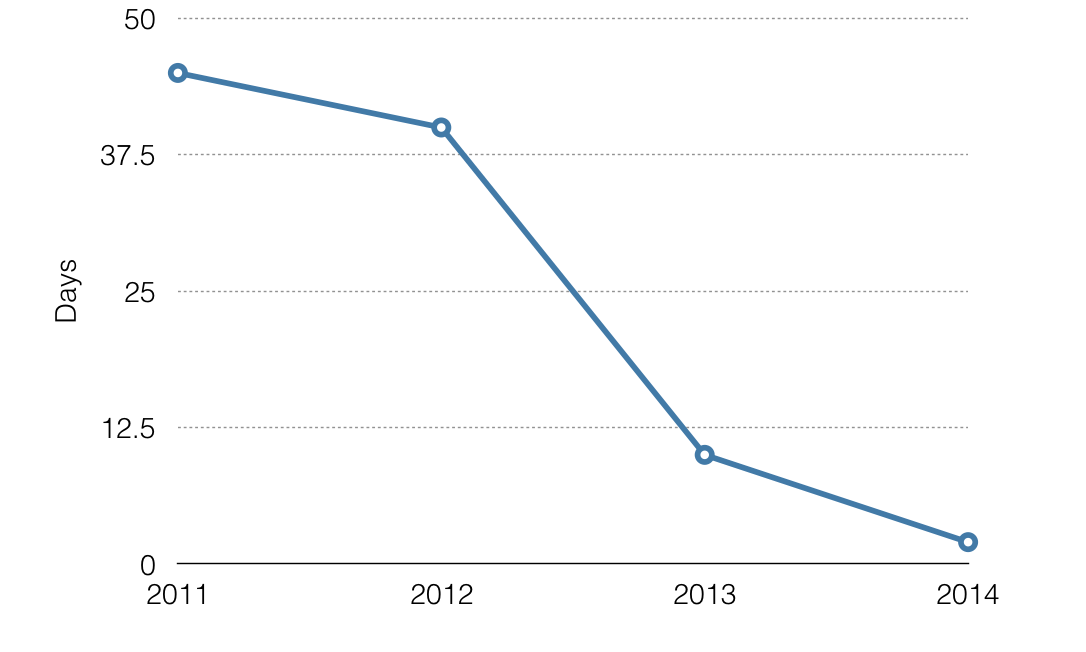
# Introduction

This section explains the *technical* side of my work, on analysis, maps and automation.

The reason why much of my time has gone into the development, improvement and maintenance of the scripts is simple - next time the same task is requested again, I have an existing method, or previously working solution, that could be re-used or easily adapted. This saves time and increases productivity

Wherever possible, I try to automate. In some cases, this obviously represents a poor investment of time, i.e., doing it manually step by step may be quicker. Automation for one-off tasks is over-engineering. I would argue, however, any time spent on making a tool, or spent on mastering a potentially more productive tool is a just investment that brings benefits in the long term.

For example, the different implementations of the comparative analysis have led to a reduced workload significantly. See below a graph depicting the amount of days to do the spatial overlay between the plethora of datasets and new nominations. This frees time for species richness, irreplaceability, improved maps, just to name a few.



time-spent-ca

Some of the tasks present similar technical challenges. Especially for frequent, recurring tasks, it is imperative that I develop libraries or templates to minimise repetitive work, and that more time is made available for creative tasks and increased productivity.

# Geoprocessing libraries

The geoprocessing library (located at geoprocessing\library) is a collection of commonly used functions, which evolve from specific solutions to tasks sharing similarities. Most of them relate to spatial analyses, usually involving the use of the arcpy library, which requires the installation of ArcGIS and a license.

In order to use these libraries, the scripts need to be visible to Python. The easiest way to achieving this is to set system variable PYTHONPATH to include the geoprocessing\library path.

The library consist the follow components:

1. [Yichuan10.py](https://github.com/Yichuans/geoprocessing/blob/master/library/Yichuan10.py) holds a collection of mostly utility functions (with a humble and long history, starting with ArcGIS version 10, which still reflects in the name of the library). Commonly used functions include the two [‘decorators’](https://www.thecodeship.com/patterns/guide-to-python-function-decorators/) to track memory usage, and time required for executing functions, and the GetUniqueValuesFromFeatureLayer\_mk2 function that finds unique values in any given field in a feature layer. The latter simplifies the common task of finding unique IDs, such as the wdpaid for the WDPA or id\_no for Red List.

def GetUniqueValuesFromFeatureLayer\_mk2(inputFc, inputField):  
 """<string>, <string> -> pythonList  
 can be both feature class or feature layer"""  
 pySet = set()  
 with arcpy.da.SearchCursor(inputFc, inputField) as cursor:  
 for row in cursor:  
 pySet.add(row[0])  
  
 return list(pySet)

1. [YichuanDB.py](https://github.com/Yichuans/geoprocessing/blob/master/library/YichuanDB.py) includes a number of useful functions to connect and manipulate postgres databases. For the most part, I use the ConnectionParameter class to simplify making connections to the postgres database. Convenient functions like clean\_view enables testing and debugging of the comparative analyses, which relies on direct access and manipulation of data in the database.
2. [YichuanM.py](https://github.com/Yichuans/geoprocessing/blob/master/library/YichuanM.py) evolved from a suite of utility functions originally resided in the Yichuan10.py that specifically aimed at tasks relating to the automatic/batch production of maps (use pre-authored map template, to be discussed in detail in the [map batch template](#Map-batch-template))
3. [YichuanRAS.py](https://github.com/Yichuans/geoprocessing/blob/master/library/YichuanRAS.py) contains functions that manipulates raster datasets (the rest on vectors) using both the arcpy but also gdal, an open source but more lower level alternative. Many of the functions here underpin the [analysis of landcover change](world-heritage-knowledge-lab.md#links-to-prototypes) which was mostly implemented using gdal
4. [YichuanSR.py](https://github.com/Yichuans/geoprocessing/blob/master/library/YichuanSR.py) has many functions relating to coordinate systems. For example, the function get\_desirable\_sr tries to guess a best optimal coordinate system based on the geometry (the input expects arcpy.Geometry or arcpy.Extent object). This is useful to identify the most appropriate coordinate system when automating map productions - it’s never a good idea to use a global coordinate system for a site scale map - use a specific UTM instead! NB. This function pre-dates a similar [CalculateUTMZone](http://desktop.arcgis.com/en/arcmap/10.3/tools/cartography-toolbox/calculate-utm-zone.htm) function, now included by default in newer releases of ArcGIS.

# Map batching template

# Richness template

# Parallel template

# Generic data analysis

# ArcGIS and Anaconda

# Miscellaneous