# SDG paper Visualizations SOBOL results

June 16, 2020

## 1 SDG paper Visualizations SOBOL results

### 1.0.1 First, import packages

In [9]:
```
from ema_workbench import (Model, RealParameter, ScalarOutcome, Constant, Policy, perfo
                           TimeSeriesOutcome, perform_experiments, save_results, load_r
from ema_workbench.connectors.vensim import (VensimModel)
from ema_workbench.em_framework import CategoricalParameter
from ema_workbench.em_framework.evaluators import LHS, SOBOL

import ema_workbench.analysis.plotting_util as plt_util
from ema_workbench.analysis.plotting import (group_by_envelopes, single_envelope, plot_
                                             lines, envelopes, kde_over_time, multiple_e
from ema_workbench.analysis import clusterer, plotting, Density, pairs_plotting,get_ex
from ema_workbench.analysis import scenario_discovery_util as sdutil
from SALib.analyze import sobol
from ema_workbench.em_framework.salib_samplers import get_SALib_problem

import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import altair as alt
```

### 1.0.2 Set WD

In [4]:
```
wd = 'C:/Users/erika/Desktop/EMA_runs/Final/'
ema_logging.log_to_stderr(ema_logging.INFO)
```

Out[4]:
```
<Logger EMA (DEBUG)>
```

### 1.0.3 Define model

In [6]:
```
Cobalt_model = VensimModel('Vensim', wd = wd, model_file=wd+'Cobalt_June15.vpmx')
```

In [18]:
```
Cobalt_model.uncertainties = [
    #     Switches
    #         CategoricalParameter('Switch opportunity cost fixed stock',(1,2),pff = True
```

```
            CategoricalParameter('Switch SSP', (1,2,3,4,5)),# ,pff = True),
            CategoricalParameter('Switch carbon policy',(0,1) ),
            CategoricalParameter('Switch energy price growth scenario', (1,2,3) ),
            CategoricalParameter('Switch real price', (1,2)),
        #     Floats
            RealParameter('Percentage lost during artisanal mining',0.4 ,0.6 ),
            RealParameter('Percentage of primary scrap',0.25 ,0.4 ),
            RealParameter('Initial average lifetime of metal in use',5 ,15 ),
            RealParameter('Collection rate metal products',0.4 ,0.8 ),
            RealParameter('Minimum usage smelting and refining capacity', 0.7,0.9),
            RealParameter('Productivity of artisanal mining', 800, 1600),
            RealParameter('Maximum increase recovery rate', 0.05, 0.25),
            RealParameter('Slowing of increase in demand stationary storage', 0.88 , 0.96)
            RealParameter('Battery capacity BEV', 30 , 120),
            RealParameter('Power for oregrades', 0.38, 0.42 ),
        ]

        Cobalt_model.outcomes =  [
        # #     General
            TimeSeriesOutcome('TIME'),
            TimeSeriesOutcome('Total demand[Cobalt]'),
            TimeSeriesOutcome('Artisanal ore trade[Cobalt]'),
            TimeSeriesOutcome('Cumulative artisanal ore trade[Cobalt]')
        ]
```

#### 1.0.4 Load results

```
In [7]: #import results,  1 = Fixed stock, 2 = Opportunity cost
        exp_1,out_1 = load_results('C:/Users/erika/Desktop/EMA_runs/Final/sobol_1.tar.gz')
        exp_2,out_2 = load_results('C:/Users/erika/Desktop/EMA_runs/Final/sobol_2.tar.gz')
```

```
[MainProcess/INFO] results loaded succesfully from C:\Users\erika\Desktop\EMA_runs\Final\sobol_
```

### 1.1 SOBOL

```
In [10]: def get_sobol_indices_overtime (problem,variable,outcome_dataset):
             y = outcome_dataset[variable]
             all_scores = []
             top_x_S1 = set()
             for i in range(2, y.shape[1], 2):
                 data = y[:, i]
                 scores = sobol.analyze(problem, data , calc_second_order = True)
                 S1_scores = pd.DataFrame(scores['ST'],index = list(problem['names']))
         #         top_x_S1 |= set(S1_scores.nlargest(top_nr, 0).index.values)
                 all_scores.append(S1_scores)
             all_scores = pd.concat(all_scores, axis=1, sort=False)
         #     all_scores = all_scores.loc[top_x_S1, :]
```

```
        all_scores.columns = np.arange(2000, 2050, 2)
        all_scores = all_scores.sort_values(by = [2000], ascending = False)
#     for i in all_scores.T:
#         if max(all_scores.T[i]) < 0.15:
#             all_scores_transposed = all_scores.T.drop[i]
        return (all_scores)
```

```
In [11]: def get_sobol_topx (problem,variable,top_nr,outcome_dataset):
        y = outcome_dataset[variable]
        all_scores = []
        top_x_S1 = set()
        for i in range(2, y.shape[1], 2):
            data = y[:, i]
            scores = sobol.analyze(problem, data , calc_second_order = True)
            S1_scores = pd.DataFrame(scores['ST'],index = list(problem['names']))
            top_x_S1 |= set(S1_scores.nlargest(top_nr, 0).index.values)
            all_scores.append(S1_scores)
        all_scores = pd.concat(all_scores, axis=1, sort=False)
        all_scores = all_scores.loc[top_x_S1, :]
        all_scores.columns = np.arange(2000, 2050, 2)
        all_scores = all_scores.sort_values(by = [2000], ascending = False)
#     for i in all_scores.T:
#         if max(all_scores.T[i]) < 0.15:
#             all_scores_transposed = all_scores.T.drop[i]
        return (all_scores)
```

```
In [12]: def plot_heatmap (scores,title):
        sns.heatmap(scores, cmap='viridis')
        fig = plt.gcf()
        ax = fig.get_axes()
        fig.autofmt_xdate()
        fig.set_size_inches(10,5)
        fig.suptitle('Sobol indices for variables with highest impact on '+title)
        shorttitle = title.replace(" ","")
        fig.savefig(wd+shorttitle)
        plt.show()
```

```
In [13]: def plot_heatmap_overtime (scores,title):
        sns.heatmap(scores, cmap='viridis')
        fig = plt.gcf()
        ax = fig.get_axes()
        ax[0].set_xticklabels(np.arange(2000, 2051, 2))
        fig.autofmt_xdate()
        fig.set_size_inches(15,5)
        fig.suptitle('Sobol indices for variables with highest impact on '+title)
        shorttitle = title.replace(" ","")
        fig.savefig(wd+shorttitle)
        plt.show()
```

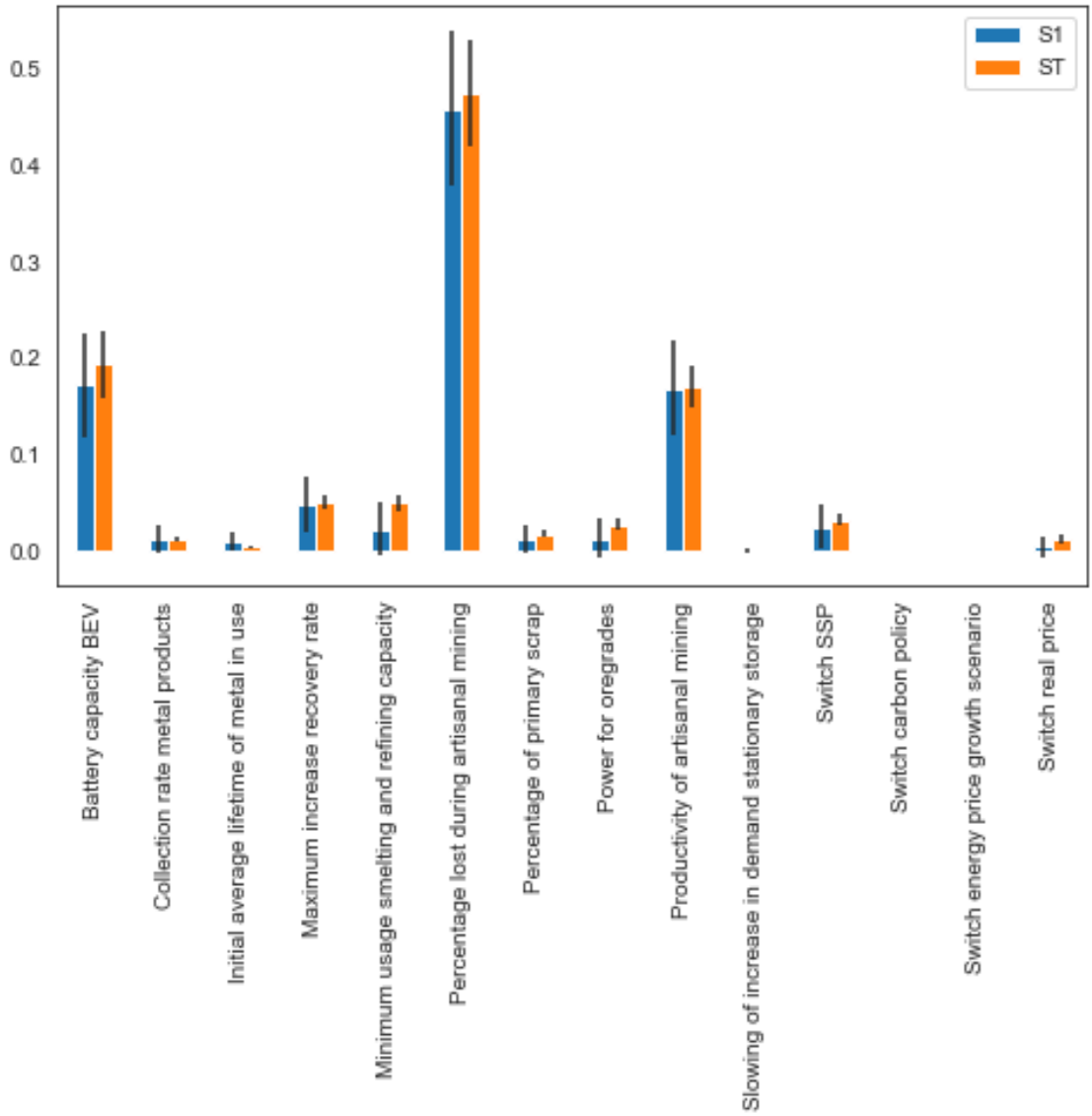
### 1.1.1 For cumulative values

```
In [15]: demand_mean_1 = np.mean(out_1['Total demand[Cobalt]'],axis = 1)/2204600
         demand_cumulative_1 = np.sum(out_1['Total demand[Cobalt]'],axis = 1)/2204600

         demand_mean_2 = np.mean(out_2['Total demand[Cobalt]'],axis = 1)/2204600
         demand_cumulative_2 = np.sum(out_2['Total demand[Cobalt]'],axis = 1)/2204600

         artisanal_mean_1 = np.mean(out_1['Artisanal ore trade[Cobalt]'],axis = 1) /2204600
         artisanal_cumulative_1 = np.sum(out_1['Artisanal ore trade[Cobalt]'],axis = 1)/220460

         artisanal_mean_2 = np.mean(out_2['Artisanal ore trade[Cobalt]'],axis = 1) /2204600
         artisanal_cumulative_2 = np.sum(out_2['Artisanal ore trade[Cobalt]'],axis = 1)/220460
```

### 1.1.2 Fixed stock SOBOL

Artisanal mining

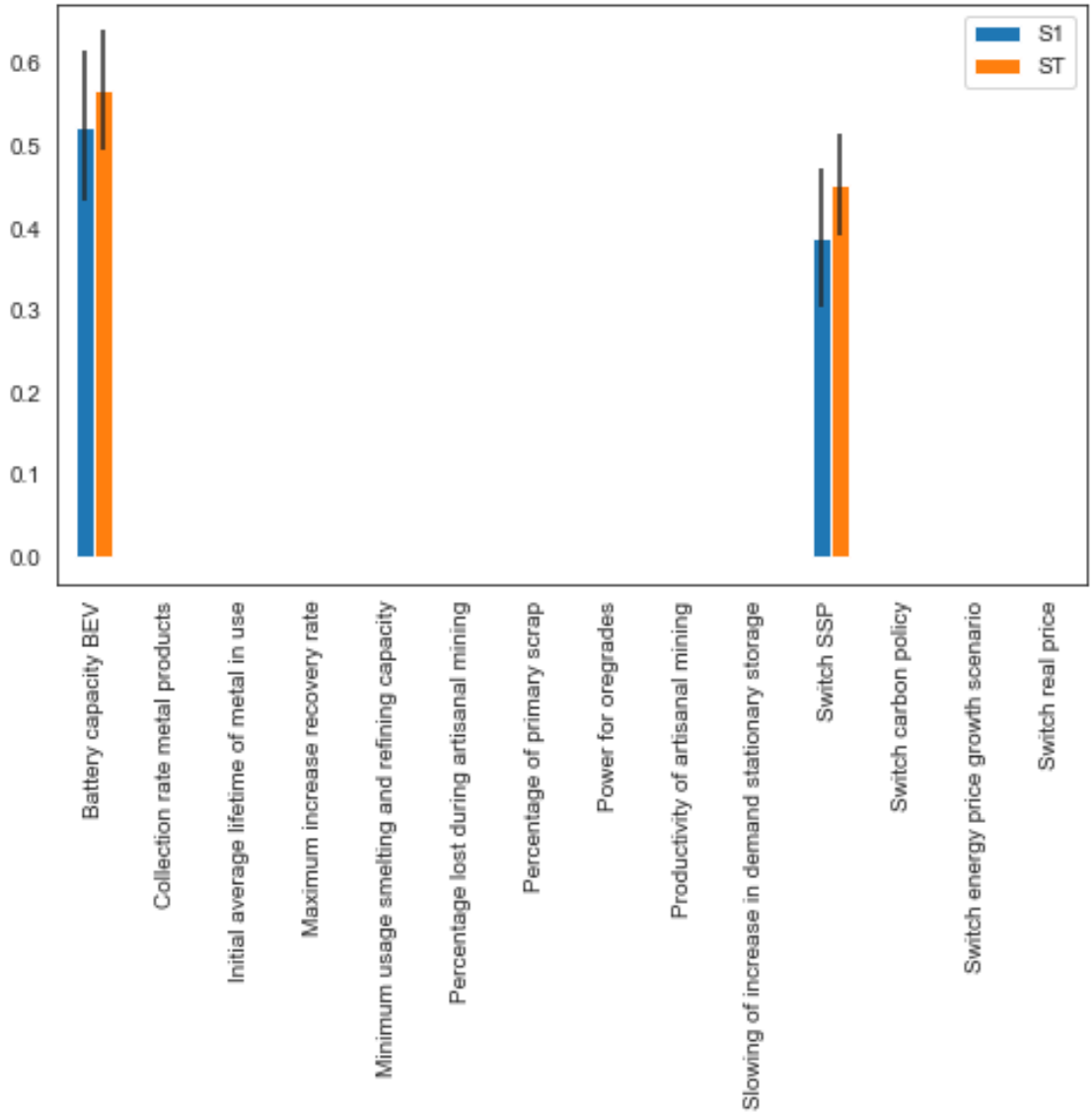
```
In [19]: problem = get_SALib_problem((Cobalt_model.uncertainties))
         Si_artisanal = sobol.analyze(problem,artisanal_cumulative_1,
                             calc_second_order = True)
         scores_filtered = {k:Si_artisanal[k] for k in ['ST','ST_conf','S1','S1_conf']}
         Si_artisanal_df = pd.DataFrame(scores_filtered, index=problem['names'])

         sns.set_style('white')
         fig, ax = plt.subplots(1)

         indices = Si_artisanal_df[['S1','ST']]
         err = Si_artisanal_df[['S1_conf','ST_conf']]

         indices.plot.bar(yerr=err.values.T,ax=ax)
         fig.set_size_inches(8,6)
         fig.subplots_adjust(bottom=0.3)
         plt.show()
```



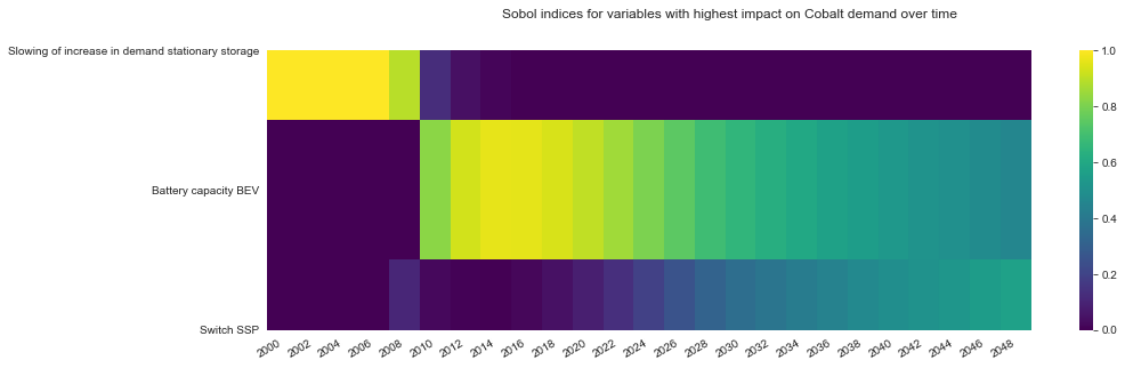

Total demand

```
In [21]: Si_demand = sobol.analyze(problem, demand_cumulative_1, calc_second_order = True)

         scores_filtered = {k:Si_demand[k] for k in ['ST','ST_conf','S1','S1_conf']}
         Si_demand_df = pd.DataFrame(scores_filtered, index=problem['names'])

         sns.set_style('white')
         fig, ax = plt.subplots(1)

         indices = Si_demand_df[['S1','ST']]
         err = Si_demand_df[['S1_conf','ST_conf']]
```

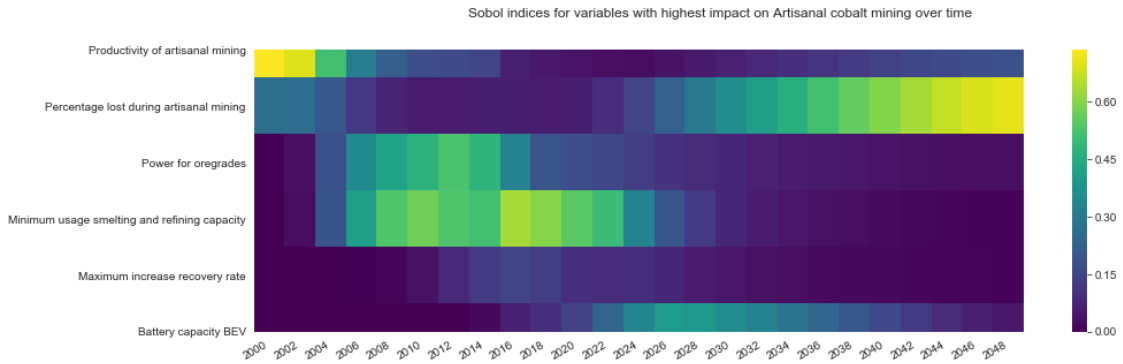
```
indices.plot.bar(yerr=err.values.T,ax=ax)
fig.set_size_inches(8,6)
fig.subplots_adjust(bottom=0.3)
plt.show()
```




In [23]:
```
Si_Demand_overtime = get_sobol_topx(problem,'Total demand[Cobalt]',2,out_1)
plot_heatmap_overtime(Si_Demand_overtime,'Cobalt demand over time')
```



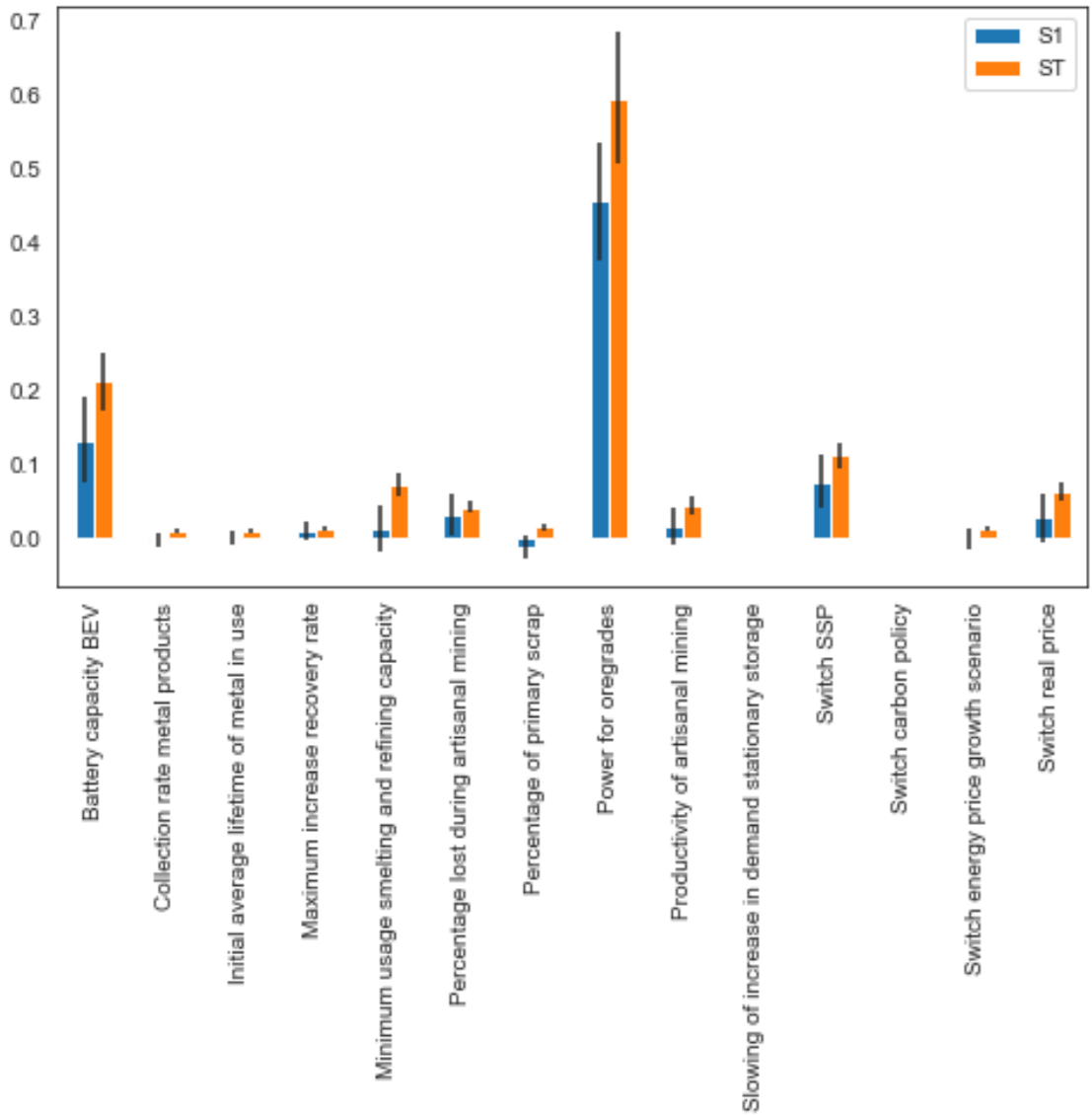

In [24]:
```
Si_Artisanal_overtime = get_sobol_topx(problem,'Artisanal ore trade[Cobalt]',3,out_1)
plot_heatmap_overtime(Si_Artisanal_overtime,'Artisanal cobalt mining over time')
```




### 1.1.3 Opportunity cost SOBOL

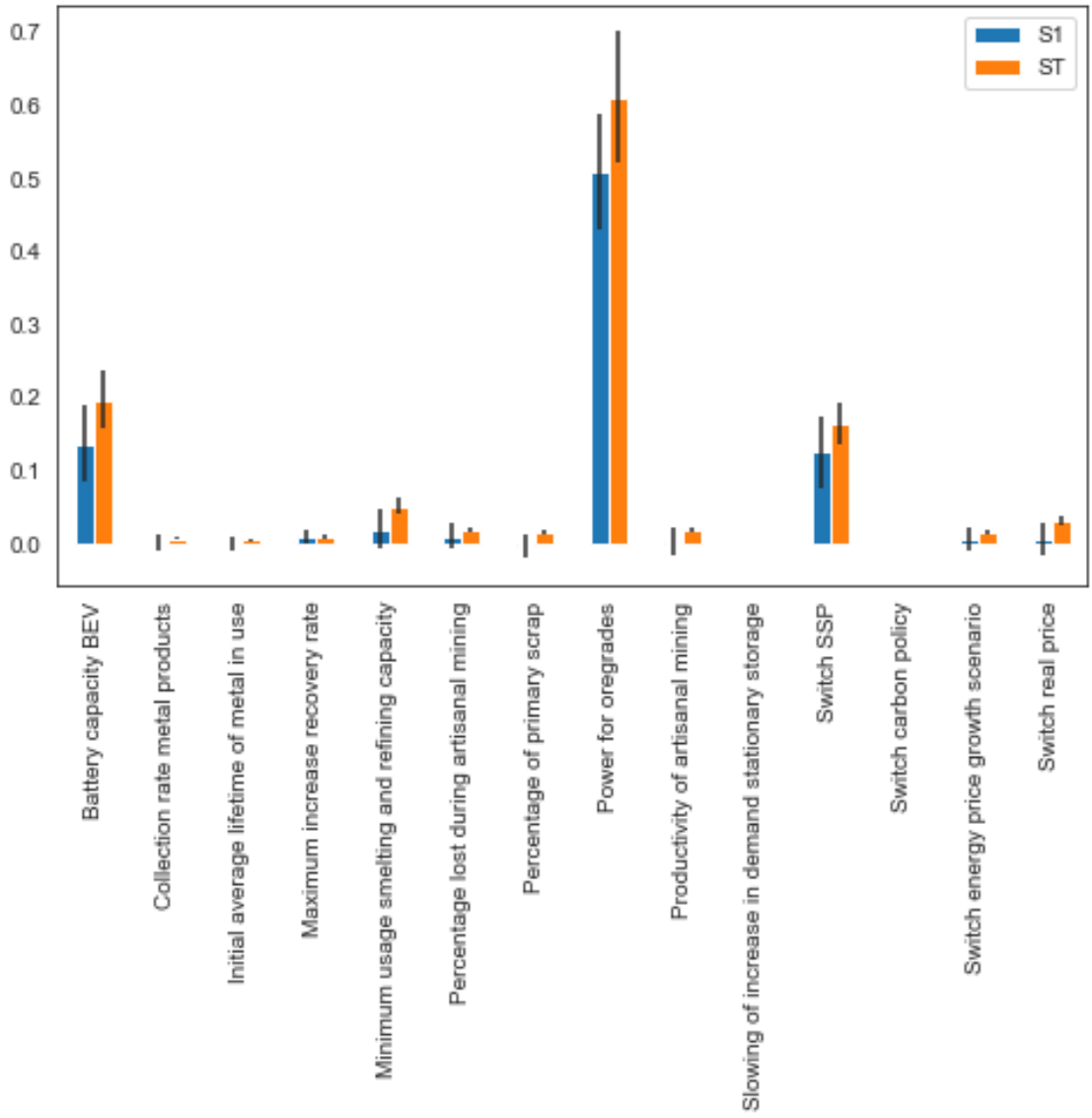
In [25]:
```
problem = get_SALib_problem((Cobalt_model.uncertainties))

Si_artisanal_2 = sobol.analyze(problem,artisanal_cumulative_2,
                   calc_second_order = True)

scores_filtered = {k:Si_artisanal_2[k] for k in ['ST','ST_conf','S1','S1_conf']}
Si_artisanal_df_2 = pd.DataFrame(scores_filtered, index=problem['names'])

sns.set_style('white')
fig, ax = plt.subplots(1)

indices = Si_artisanal_df_2[['S1','ST']]
err = Si_artisanal_df_2[['S1_conf','ST_conf']]

indices.plot.bar(yerr=err.values.T,ax=ax)
```

```
fig.set_size_inches(8,6)
fig.subplots_adjust(bottom=0.3)
plt.show()
```

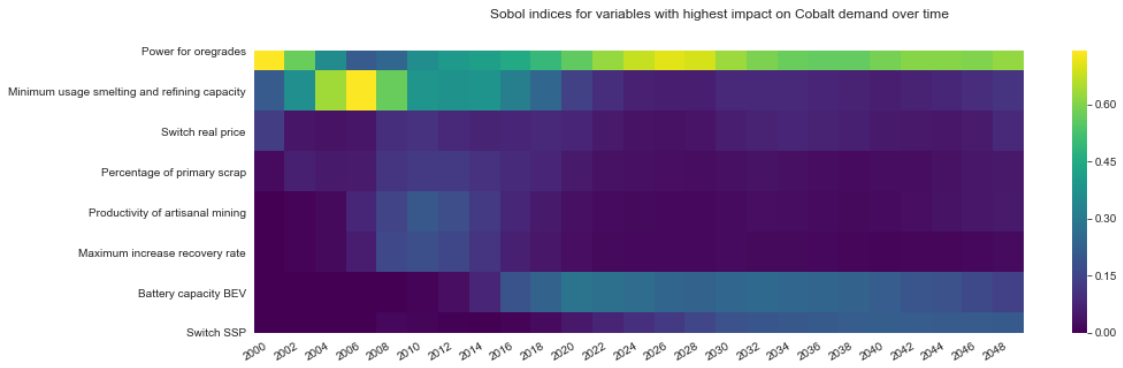



In [26]:
```
Si_demand_2 = sobol.analyze(problem, demand_cumulative_2, calc_second_order = True)

scores_filtered = {k:Si_demand_2[k] for k in ['ST','ST_conf','S1','S1_conf']}
Si_demand_df_2 = pd.DataFrame(scores_filtered, index=problem['names'])

sns.set_style('white')
fig, ax = plt.subplots(1)
```

```
indices = Si_demand_df_2[['S1','ST']]
err = Si_demand_df_2[['S1_conf','ST_conf']]

indices.plot.bar(yerr=err.values.T,ax=ax)
fig.set_size_inches(8,6)
fig.subplots_adjust(bottom=0.3)
plt.show()
```




In [27]:
```
Si_Demand_overtime_2 = get_sobol_topx(problem,'Total demand[Cobalt]',3, out_2)
plot_heatmap_overtime(Si_Demand_overtime_2,'Cobalt demand over time')
```




In [28]:
```
Si_Artisanal_overtime_2 = get_sobol_topx(problem,'Artisanal ore trade[Cobalt]',3,out_
plot_heatmap_overtime(Si_Artisanal_overtime_2,'Artisanal cobalt mining over time')
```