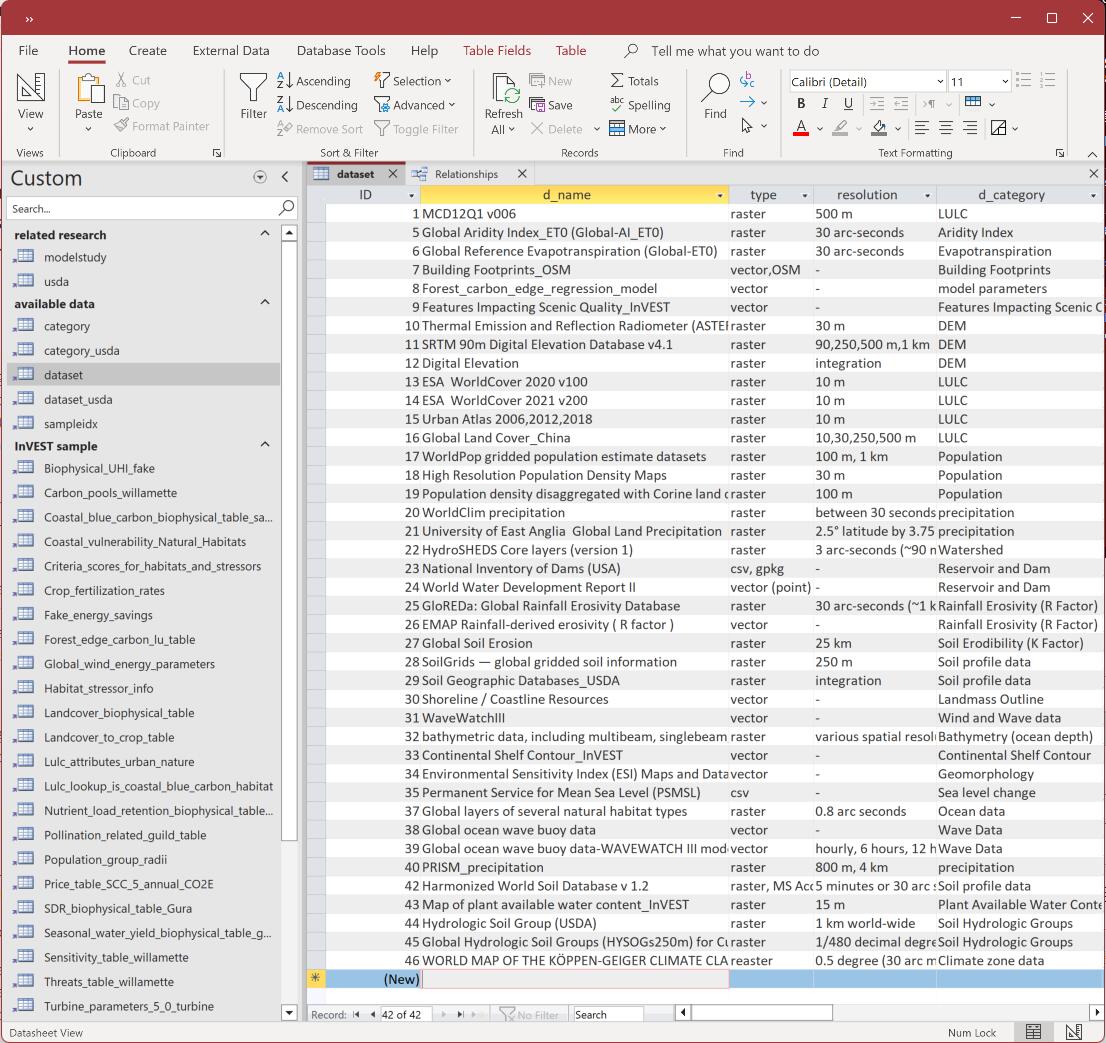
# 数据集检索

为了方便检索、查看数据集和观察各个模型使用了哪些数据，及模型间共同调用同类数据的情况，用[Microsoft Access（MS Access）](https://www.microsoft.com/en-us/microsoft-365/access)①数据库管理系统（database management system，DBMS）录入数据建立数据库。然后用 Python 从建立的数据库中读入数据，并用[Dash（plotly）](https://dash.plotly.com/installation)②建立网页交互式应用程序，用于数据库检索查看。数据集按两种情况划分，一种是本书章节知识点对应使用的数据集；另一种是增加了诸如[InVEST](https://naturalcapitalproject.stanford.edu/software/invest)③等研究模型或相关论文中模型所使用的数据。MS Access界面如下，



因为可视化数据需要数据集或数据库等数据文件，数据文件一般文件较大，因此并没有将交互式可视化数据应用布局到 [USDA](https://richiebao.github.io/USDA_PyPI/#/) ④库中，而是单独建立了[usda-dashboard](https://pypi.org/project/usda-dashboard/) ⑤PyPI包，可以通过pip install usda-dashboard方法安装。通过执行usda\_dashboard.dash\_usda\_datasetidx()可以直接打开一个交互式网页应用。

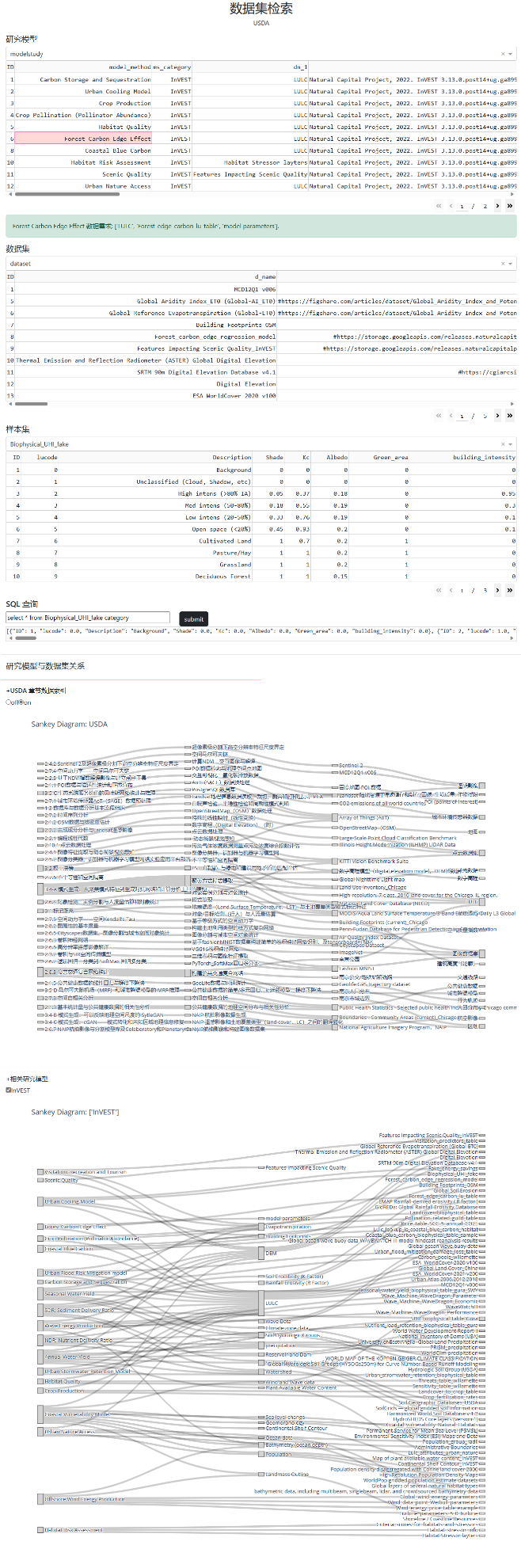
数据集检索应用布局内容包括两个部分，一部分是检索数据库中的表内容，可以在下拉菜单中选取表查看，也可以通过 SQL 语句查询。其中研究模型为本书章节和 InVEST等扩展模型对应的数据集的表；数据集为所用数据集的信息；样本集为模型所用的基本参数表。另一部分是用 Sankey 图表述本书各个章节和InVEST等扩展模型间所用数据集的关系，并对模型所用数据有一个宏观上的概览。

import usda\_dashboard  
usda\_dashboard.dash\_usda\_datasetidx()

Dash is running on http://127.0.0.1:8050/  
  
 \* Serving Flask app 'usda\_dashboard.\_dash\_USDA\_dataIndex'  
 \* Debug mode: off  
  
  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
 \* Running on http://127.0.0.1:8050  
Press CTRL+C to quit  
127.0.0.1 - - [17/Jun/2023 09:20:10] "GET /dash-nba/ HTTP/1.1" 200 -  
127.0.0.1 - - [17/Jun/2023 09:20:10] "GET /\_dash-layout HTTP/1.1" 200 -



直接调用应用，其结果如上图。可以通过检索的不同方式查看数据，如下图。



源代码托管于GitHub 上的[USDA\_dashboard](https://github.com/richieBao/USDA_dashboard) ⑥代码库。附数据集检索部分的代码如下，

import warnings  
warnings.filterwarnings('ignore')  
  
import pandas as pd  
pd.set\_option('display.max\_columns', None)  
  
import pyodbc  
from sqlalchemy import create\_engine  
import json  
import itertools  
import numpy as np  
  
from dash import Dash, dcc, html, Input, Output, dash\_table, State,no\_update  
import plotly.express as px  
import dash\_bootstrap\_components as dbc  
import plotly.graph\_objects as go  
  
from sqlalchemy import create\_engine  
  
flatten\_lst=lambda lst: [m for n\_lst in lst for m in flatten\_lst(n\_lst)] if type(lst) is list else [lst]   
def update\_df\_foreign\_keys(df,dn\_foreign\_keys\_mapping):  
 for col in df.columns:  
 if col in dn\_foreign\_keys\_mapping.keys():  
 related\_df=pd.read\_sql('select \* from %s'%(dn\_foreign\_keys\_mapping[col][0]),conn)  
 df[col]=df[col].apply(lambda idx:related\_df[related\_df.ID==idx][dn\_foreign\_keys\_mapping[col][1]].item())   
   
 return df  
  
def sql\_query2list(statement,conn):  
 cursor=conn.cursor()  
 cursor.execute(statement)   
 fetched\_data=[dict((cursor.description[i][0], value) for i, value in enumerate(row)) for row in cursor.fetchall()]  
   
 return fetched\_data  
  
# datasets query  
import pkg\_resources  
db\_fn=pkg\_resources.resource\_filename('usda\_dashboard', 'data/USDA\_dataIndex.accdb') # db\_fn='.data/USDA\_dataIndex.accdb'  
conn=pyodbc.connect(r'Driver={Microsoft Access Driver (\*.mdb, \*.accdb)};DBQ=%s;'%db\_fn)  
  
dn\_foreign\_keys\_mapping={'dn\_1':['category','d\_category'],  
 'dn\_2':['sampleidx','sample\_idx'],  
 'dn\_3':['category','d\_category'],  
 'dn\_4':['category','d\_category'],  
 'dn\_5':['sampleidx','sample\_idx'],  
 'dn\_6':['sampleidx','sample\_idx'],  
 'dn\_7':['category','d\_category'],  
 'dn\_8':['category','d\_category'],  
 'dn\_9':['category','d\_category'],  
 'dn\_10':['category','d\_category'],  
 'dn\_11':['category','d\_category'],  
 'dn\_12':['category','d\_category'],  
 'dn\_13':['sampleidx','sample\_idx'],  
 'd\_category':['category','d\_category'],  
 'usda\_dn':['dataset\_usda','d\_name'],  
 'usda\_category':['category\_usda','usda\_category']}  
  
# dashboard  
# external\_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']  
app = Dash(\_\_name\_\_, external\_stylesheets=[dbc.themes.BOOTSTRAP]) #  
app.config.suppress\_callback\_exceptions = True  
  
sample\_data\_list\_=sql\_query2list('select a.sample\_idx from sampleidx as a',conn)  
sample\_data\_list=[i['sample\_idx'] for i in sample\_data\_list\_ if i['sample\_idx']!='Null']  
  
modelstudy\_ms\_category=np.unique([i['ms\_category'] for i in sql\_query2list('select a.ms\_category from modelstudy as a',conn)])  
usda\_points=[i['point'] for i in sql\_query2list('select a.point from usda as a',conn)]  
  
table\_dict={'related\_research':['usda','modelstudy'],  
 'available\_data':['category','dataset','sampleidx','category\_usda','dataset\_usda'],  
 'sample\_data':sample\_data\_list}  
  
styles = {  
 'pre': {  
 'border': 'thin lightgrey solid',  
 'overflowX': 'scroll'  
 },  
 'h7':{  
 'color': 'k',  
 'font-size': '16px',  
 'text-transform': 'uppercase'   
 }  
}  
  
app.layout = dbc.Container(  
 [  
 dbc.Row(  
 [  
 dbc.Col(  
 [html.H2("数据集检索", style={"text-align": "center"}),html.P('USDA',style={"text-align": "center"})]  
 ),   
 ]  
 ),  
 dbc.Row([  
 dbc.Col([  
 html.H5('研究模型'),  
 dcc.Dropdown(id='d\_related\_research',options=table\_dict['related\_research']),  
 html.Div(id='t\_related\_research'),  
 # dash\_table.DataTable(id='dt\_related\_research'),   
 dbc.Alert(id='alert\_model\_datasets'),  
 ]),  
 dbc.Col([  
 html.H5('数据集'),  
 dcc.Dropdown(id='d\_available\_data',options=table\_dict['available\_data']),  
 html.Div(id='t\_available\_data'),  
 ]),   
 dbc.Col([  
 html.H5('样本集'),  
 dcc.Dropdown(id='d\_sample\_data',options=table\_dict['sample\_data']),  
 html.Div(id='t\_sample\_data'),  
 ]),   
 ]),  
 dbc.Row([html.H5('SQL 查询'),]),  
 dbc.Row([   
 dbc.Col([   
 dcc.Input(id='query',value='',placeholder="e.g.: select a.ref\_citation from modelstudy as a",type='text',style={'width':'100%'}),  
 ],width=4),   
 dbc.Col([   
 dbc.Button('submit',id='button\_query\_submit',color="dark",n\_clicks=0, ),  
 ],width='auto'),   
 dbc.Col([  
 # dbc.Alert(id='alert\_query'),  
 html.Pre(id='pre\_query', style=styles['pre']),  
 ],),  
 ]),   
 dbc.Row([   
 html.Hr(style={'marginTop': '1em'}),  
 html.H5('研究模型与数据集关系'),  
 html.Hr(style={'borderColor':'red','marginTop': '1em', "width": "50%"}),  
 html.H6('+USDA 章节数据索引'),  
 dcc.RadioItems(id='r\_usda\_points',options=['off','on'],inline=True,value='off'),  
 dcc.Graph(id='g\_usda\_sankey'),  
 html.Hr(style={'borderColor':'red','marginTop': '1em', "width": "50%"}),  
 html.H6('+相关研究模型'),  
 dcc.Checklist(id='cl\_modelstudy',options=modelstudy\_ms\_category,inline=True),  
 dcc.Graph(id='g\_sankey'),  
 ]),   
  
 ]  
 )  
  
@app.callback(  
 Output('t\_related\_research','children'),  
 Input('d\_related\_research',"value"),   
 )  
def related\_research\_table(table\_name):  
 if table\_name is not None:  
 df=pd.read\_sql('select \* from %s'%(table\_name),conn)   
 df=update\_df\_foreign\_keys(df,dn\_foreign\_keys\_mapping)  
 df=df.reset\_index().rename(columns={"index": "id"})  
   
 return dash\_table.DataTable(df.to\_dict('records'),  
 [{"name": i, "id": i} for i in df.columns if i != 'id'],  
 style\_table={'overflow':'auto'},  
 page\_size=10,  
 # page\_action='native',  
 id='dt\_related\_research')  
   
@app.callback(  
 Output('t\_available\_data','children'),  
 Input('d\_available\_data',"value"),   
 )  
def available\_data\_table(table\_name):  
 if table\_name is not None:  
 df=pd.read\_sql('select \* from %s'%(table\_name),conn)   
 return dash\_table.DataTable(df.to\_dict('records'),  
 [{"name": i, "id": i} for i in df.columns],  
 style\_table={'overflow':'auto'},  
 page\_size=10,)  
   
@app.callback(  
 Output('t\_sample\_data','children'),  
 Input('d\_sample\_data',"value"),   
 )  
def sample\_data\_table(table\_name):  
 if table\_name is not None:  
 df=pd.read\_sql('select \* from %s'%(table\_name),conn)   
 return dash\_table.DataTable(df.to\_dict('records'),  
 [{"name": i, "id": i} for i in df.columns],  
 style\_table={'overflow':'auto'},  
 page\_size=10,)  
  
@app.callback(  
 Output('alert\_model\_datasets','children'),  
 Input('dt\_related\_research','active\_cell'),  
 Input('dt\_related\_research','data'),  
 Input('dt\_related\_research','columns'),   
 )  
def model\_datasets(active\_cell,rows,columns):   
 df = pd.DataFrame(rows, columns=[c['name'] for c in columns])  
 if active\_cell:   
 row\_values=df.iloc[active\_cell['row\_id']]  
 row\_values\_dn=row\_values[[col for col in df.columns if col in dn\_foreign\_keys\_mapping.keys()]]  
 row\_values\_dn\_values=[i for i in row\_values\_dn.values if i!='Null']  
 try:  
 mm=row\_values['model\_method']  
 except:  
 mm=row\_values['point']  
 return f'{mm} 数据需求: {row\_values\_dn\_values}.'   
   
@app.callback(  
 Output('pre\_query','children'),  
 Input("button\_query\_submit", "n\_clicks"),  
 State('query','value'),  
 )  
def sql\_query(n\_clicks,statement):   
 if n\_clicks:   
 cursor=conn.cursor()  
 try:  
 cursor.execute(statement) # select \* from Biophysical\_UHI\_fake category;   
 fetched\_data=[dict((cursor.description[i][0], value) for i, value in enumerate(row)) for row in cursor.fetchall()]  
 return json.dumps(fetched\_data)  
 except:  
 return json.dumps('SQL statement is wrong!')  
   
 return json.dumps('Null')  
  
@app.callback(  
 Output('g\_sankey','figure'),  
 Input('cl\_modelstudy',"value"),   
 )  
def display\_sankey\_modelstudy(related\_reserch):  
 modelstudy=pd.read\_sql('select \* from %s'%('modelstudy'),conn)   
 modelstudy=update\_df\_foreign\_keys(modelstudy,dn\_foreign\_keys\_mapping)  
   
 dataset=pd.read\_sql('select \* from %s'%('dataset'),conn)   
 dataset=update\_df\_foreign\_keys(dataset,dn\_foreign\_keys\_mapping)   
   
 source\_target\_links=[]  
 if related\_reserch is not None:  
 if len(related\_reserch)>0:  
 modelstudy\_selection=modelstudy[modelstudy.ms\_category.isin(related\_reserch)]  
 for dn in dn\_foreign\_keys\_mapping.keys():   
 if dn in modelstudy.columns:  
 source\_target\_links.append(list(zip(modelstudy['model\_method'],modelstudy[dn])))   
   
 category\_filter=[i[1] for i in flatten\_lst(source\_target\_links)]   
 for dn in dn\_foreign\_keys\_mapping.keys():  
 if dn in dataset.columns:   
 dataset\_selection=dataset[dataset[dn].isin(set(dataset[dn]).intersection(set(category\_filter)))]  
 source\_target\_links.append(list(zip(dataset\_selection[dn],dataset\_selection['d\_name'])))   
   
 source\_target\_links=flatten\_lst(source\_target\_links)   
 labels=list(set(list(itertools.chain(\*source\_target\_links))))  
 source\_target\_links\_df=pd.DataFrame(source\_target\_links,columns=['source','target'])  
 source\_target\_links\_df=source\_target\_links\_df[(source\_target\_links\_df['source']!='Null') & (source\_target\_links\_df['target']!='Null')]  
 source\_target\_links\_df['source\_id']=source\_target\_links\_df.source.apply(lambda x:labels.index(x))  
 source\_target\_links\_df['target\_id']=source\_target\_links\_df.target.apply(lambda x:labels.index(x))  
   
 style\_sankey=dict(  
 pad = 15,  
 thickness = 15,  
 line = dict(color = "black", width = 0.5),  
 color = "lightgray")   
   
 node=dict(label = labels)  
 node.update(style\_sankey)  
   
 link= dict(  
 source =source\_target\_links\_df['source\_id'],   
 target =source\_target\_links\_df['target\_id'],  
 value = [1]\*len(source\_target\_links)  
 )   
   
 fig = go.Figure(data=[go.Sankey(  
 node = node,  
 link =link,  
 )])  
   
 fig.update\_layout(title\_text=f"Sankey Diagram: {related\_reserch}", font\_size=13,autosize=False,height=1200)   
 return fig   
 else:  
 return {}  
 else:  
 return {}  
   
@app.callback(  
 Output('g\_usda\_sankey','figure'),  
 Input('r\_usda\_points',"value"),   
 )  
def display\_sankey\_usda(selection):  
 if selection=='on':  
 usda=pd.read\_sql('select \* from %s'%('usda'),conn)   
 usda=update\_df\_foreign\_keys(usda,dn\_foreign\_keys\_mapping)  
   
 dataset\_usda=pd.read\_sql('select \* from %s'%('dataset\_usda'),conn)   
 dataset\_usda=update\_df\_foreign\_keys(dataset\_usda,dn\_foreign\_keys\_mapping)  
   
 source\_target\_links=[]  
 source\_target\_links.append(list(zip(usda['chapter'],usda['point'])))   
 source\_target\_links.append(list(zip(usda['point'],usda['usda\_dn'])))   
 source\_target\_links.append(list(zip(dataset\_usda['d\_name'],dataset\_usda['usda\_category'])))   
   
 source\_target\_links=flatten\_lst(source\_target\_links)   
 labels=list(set(list(itertools.chain(\*source\_target\_links))))  
 source\_target\_links\_df=pd.DataFrame(source\_target\_links,columns=['source','target'])  
 source\_target\_links\_df=source\_target\_links\_df[(source\_target\_links\_df['source']!='Null') & (source\_target\_links\_df['target']!='Null')]  
 source\_target\_links\_df['source\_id']=source\_target\_links\_df.source.apply(lambda x:labels.index(x))  
 source\_target\_links\_df['target\_id']=source\_target\_links\_df.target.apply(lambda x:labels.index(x))  
   
 style\_sankey=dict(  
 pad = 15,  
 thickness = 15,  
 line = dict(color = "black", width = 0.5),  
 color = "lightgray")   
   
 node=dict(label = labels)  
 node.update(style\_sankey)  
   
 link= dict(  
 source =source\_target\_links\_df['source\_id'],   
 target =source\_target\_links\_df['target\_id'],  
 value = [1]\*len(source\_target\_links)  
 )   
   
 fig = go.Figure(data=[go.Sankey(  
 node = node,  
 link =link,  
 )])  
   
 fig.update\_layout(title\_text=f"Sankey Diagram: USDA", font\_size=13,autosize=False,height=900)   
 return fig   
 else:  
 return {}  
   
   
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run\_server(debug=True)

注释（Notes）：

① Microsoft Access（MS Access）数据库管理系统，（<https://www.microsoft.com/en-us/microsoft-365/access>）。

② Dash（plotly）网页交互式应用程序，（<https://dash.plotly.com/installation>）。

③ InVEST（Integrated Valuation of Ecosystem Services and Tradeoffs），（<https://naturalcapitalproject.stanford.edu/software/invest>）。

④ USDA 本书配套Python包PyPI库，（<https://richiebao.github.io/USDA_PyPI>）。

⑤ usda-dashboard，本书配套Python包PyPI库，（<https://pypi.org/project/usda-dashboard/>）。

⑥ USDA\_dashboard 本书配套用，托管于GitHub的代码库，（<https://github.com/richieBao/USDA_dashboard>）。