Česká zemědělská univerzita v Praze

Fakulta životního prostředí

Katedra prostorových věd

Nástroj pro výzkum změn spektrálních charakteristik výskytu druhů

Bakalářská práce

Vedoucí práce: doc. Ing. Petra Šímová, Ph. D.

Bakalant: Michael Kučera

2022

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Fakulta životního prostředí

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Michael Kučera

Geografické informační systémy a dálkový průzkum Země v životním prostředí

Název práce

Nástroj pro výzkum změn spektrálních charakteristik výskytu druhů

Název anglicky

Changes in spectral characteristics of the species occurence: a research tool

---

Cíle práce

Databáze o výskytu rostlinných a živočišných druhů obvykle trpí nerovnoměrností mapování, obecně označovanou jako sampling bias. Kvantifikace sampling bias v konkrétní druhové databázi by mohla výrazně zkvalitnit její využití ve výzkumu i praxi. Práce vychází z myšlenky, že (i) jelikož prostředí daného druhu se v období několika let významně nemění, může míra meziročních změn realizované niky druhu pomoci sampling bias kvantifikovat a že (ii) prostředí druhu lze popsat spektrálními charakteristikami odvozenými z dat DPZ. Hlavním cílem práce je vytvořit cloudový nástroj umožňující uživateli rychlý průzkum změn spektrálních charakteristik míst výskytu druhů dle Nálezové databáze ochrany přírody. Formulace dílčích cílů je úkolem autora.

Metodika

Autor zpracuje rešerši zaměřenou zejména na druhové databáze, modelování distribuce druhů, problematiku sampling bias a případně možnosti využití neklasifikovaných multispektrálních dat DPZ jako popisu biotopu či niky druhu.

V praktické části navrhne a naprogramuje nástroj, který s využitím cloudového prostředí (např. Google Earth Engine) a Nálezové databáze ochrany přírody (NDOP) poskytne uživateli spektrální charakteristiky výskytu zvoleného druhu ve zvoleném období a ve zvolené polohové přesnosti druhových záznamů. Autor zároveň navrhne a do nástroje zakomponuje možnost vizualizace a vyhodnocení změn spektrálních charakteristik druhu (tedy jakési realizované spektrální niky) v čase. Funkcionalitu nástroje bude demonstrovat na několika druzích živočichů z NDOP.

Doporučený rozsah práce

40 – 60 stran

Klíčová slova

Google Earth Engine, multispektrální družicové snímky, sampling bias, NDOP

Doporučené zdroje informací

Gábor, L., Moudrý, V., Barták, V., & Lecours, V. (2019). How do species and data characteristics affect species distribution models and when to use environmental filtering? International Journal of Geographical Information Science. https://doi.org/10.1080/13658816.2019.1615070

Moudrý, V., & Šímová, P. (2012). Influence of positional accuracy, sample size and scale on modelling species distributions: a review. International Journal of Geographical Information Science, 26(11), 2083–2095. https://doi.org/10.1080/13658816.2012.721553.

Palmer, M. W., Earls, P. G., Hoagland, B. W., White, P. S., & Wohlgemuth, T. (2002). Quantitative tools for perfecting species lists. Environmetrics, 13, 121–137. https://doi.org/10.1002/env.516

Rocchini, D., Salvatori, N., Beierkuhnlein, C., Chiarucci, A., de Boissieu, F., Förster, M., … Féret, J. B. (2021). From local spectral species to global spectral communities: A benchmark for ecosystem diversity estimate by remote sensing. Ecological Informatics, 61, 101195. https://doi.org/10.1016/J.ECOINF.2020.101195.

Předběžný termín obhajoby

2021/22 LS – FZP

Vedoucí práce

doc. Ing. Petra Šímová, Ph.D.

Garantující pracoviště

Katedra prostorových věd

Elektronicky schváleno dne 21. 3. 2022

doc. Ing. Petra Šímová, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 21. 3. 2022

prof. RNDr. Vladimír Bejček, CSc.

Děkan

V Praze dne 31. 03. 2022

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma *Nástroj pro výzkum změn spektrálních charakteristik výskytu druhů* vypracoval samostatně pod vedením doc. Ing. Petry Šímové, Ph. D. a že jsem uvedl všechny literární prameny, ze kterých jsem čerpal.

Prohlašuji, že tištěná verze se shoduje s verzí odevzdanou přes Univerzitní informační systém.

V Praze dne 31.3.2022 _____

(podpis autora práce)

Poděkování

Děkuji vedoucí práce, doc. Ing. Petře Šímové Ph. D., za námět tématu a vedení práce, cenné rady a připomínky, trpělivost a motivaci.

Bakalářská práce byla řešena v rámci projektu TAČR SS02030018 Centrum pro krajinu a biodiverzitu (DivLand).

V Praze dne 31.3.2022 _____

(podpis autora práce)

Abstrakt

Bakalářská práce se zabývá tvorbou nástroje pro výzkum vývoje spektrální charakteristiky druhového habitatu v čase na základě nálezových dat z Nálezové databáze ochrany přírody (NDOP) a multispektrálních družicových snímků Landsat. Nástroj může být aplikován pro kvantifikaci sampling bias nálezových dat.

Literární rešerše se zaměřuje na teoretický základ pro kvantifikaci sampling bias pomocí nálezových dat a multispektrálních dat dálkového průzkumu Země.

V praktické části je vytvořen nástroj s desktopovým grafickým uživatelským rozhraním (GUI) pro kvantifikaci sampling bias nálezových dat. Je využita cloudová platforma Google Earth Engine (GEE), ve které probíhá získání spektrálních charakteristik nálezových dat. Nálezová data uživatelem specifikovaného druhu jsou automatický získány z NDOP. Nástroj slouží k vizualizaci spektrální charakteristiky druhového habitatu a umožnuje export vizualizací i samotných dat.

Funkcionalita nástroje je demonstrována na ukázce výstupu pro plcha velkého (*Glis glis*). Výstupy nástroje jsou interpretovány a vyhodnoceny.

Vytvořený nástroj je prostředím pro výzkum Nálezové databáze ochrany přírody.

Klíčová slova:

NDOP, Google Earth Engine, multispektrální družicové snímky, sampling bias

Abstract

The Bachelor thesis deals with the development of an tool for the evaluation of temporal species habitat spectral characteristics based on species occurrence data from NDOP (czech species occurence database) and Landsat multispectral satelite imagery. The tool has potential to be used to quantify sampling bias of species occurrence data.

The review part focuses on the theoretical base of occurrence data sampling bias quantification based on multispectral satelite imagery.

A reaserch tool with and desktop graphical user interface (GUI) for sampling bias quantification is created. The tool uses the Google Earth Engine (GEE) cloud platform for spectral characterization of user specified specie occurrence data automaticaly aquired from NDOP. The tool visualizes the spectral habitat characteristics and allows visualization and data export.

The functionality of the reaserch tool is showcased on european edible dormouse (*Glis glis*). The output is interpreted and evaluated.

The created tool provides and environment for the reasearch of NDOP.

Key words:

NDOP, Google Earth Engine, multispectral satelite imagery, sampling bias

Obsah

1. Úvod a cíle práce

Poznání biodiverzity, mezidruhových vztahů a druhových distribucí je zásadní pro praktickou ekologii a ochranu přírody (Boenigk a kol. 2015). Pro kvantifikaci komplexních přírodních fenoménů a vztahů jsou často využívány druhové distribuční modely, které poskytují vhled do vztahu druhu a (přírodního) prostředí, které daný druh využívá (Miller 2010, Elith 2013). Tyto modely jsou založeny na vstupních datech, které dělíme na environmentální faktory (závislá proměnná) prostředí a biologická nálezová data (vysvětlující proměnná) (Moudrý 2015). Bakalářská práce se zaměřuje na nálezová data z nálezové databáze ochrany přírody (NDOP) a jejich nejednoznačnost, konkrétně na sampling bias. Sampling bias nálezových dat je překážkou v cestě ke kvalitním výstupům druhových distribučních modelů a následné možnosti hlubšímu porozumění přírodních vztahů, právě na základě druhových distribučních modelů (Støa a kol. 2018). Jedním z hlavních nedostatků nálezových dat je nerovnoměrnost mapování (sampling bias), jedná se o efekt upřednostňovaní dostupnějších nebo atraktivnějších lokalit pro sběr nálezových dat (Fourcade a kol. 2014). Kvantifikace sampling bias nálezových dat je potenciálním způsobem pro zlepšení využitelnosti nálezových dat pro praktické ekologické využití (ochrana přírody) nebo vědecký výzkum.

Přístup kvantifikace nerovnoměrnosti mapování daného druhu v Nálezové databázi ochrany přírody je založen na předpokladech: (i) platnosti teorie o konzervatismu ekologické niky (stálost ekologické niky v dlouhém časovém období) (Peterson 1999, Wiens a kol. 2010) a (ii) vhodnosti využití dat neklasifikovaných multispektrálních družicových snímků pro popis habitatu nebo ekologické niky druhu (Laurent a kol. 2005).

Hlavním cílem bakalářské práce bylo vytvoření desktopového nástroje pro vyhodnocení charakteristiky spektrální niky libovolného druhu. Jako dílčí cíle byly stanoveny: (i) zpracovat teoretický základ pro kvantifikaci sampling bias na základě spektrálních charakteristiky druhového habitatu, (ii) vytvořit algoritmus založený na cloudovém prostředí Google Earth Engine pro získání a vizualizaci spektrální charakteristiky druhového habitatu, (iii) vytvořit grafické uživatelské rozhraní nástroje s napojením na algoritmus a (iv) provést ukázku funkcionality nástroje pro zvolený druh a následně interpretovat výsledky.

2. Literární rešerše

2.1. Ekologie

Život lze najít kdekoliv na Zemi, od nejhlubších oceánských příkopů po nejvyšší velehory, od nejstudenějších polárních pouští po nejsušší horké pouště. Zemská přírodní rozmanitost (biodiverzita) je výsledkem evoluce trvající přes 4 miliardy let, v jejichž průběhu došlo k mnoha výkyvům, vyhubením a depresím vedoucím k velkému snížení biodiverzity. Dnes s velkým přičiněním člověka a jeho činnosti pokles biodiverzity nejspíše zdaleka překonal přírodou způsobené poklesy z předešlých 4 miliard let.

Biodiverzita je důležitá pro fungování planety Země jako globálního ekosystémů, ovlivňuje klima, úrodnost půdy, vodní cyklus, zemědělství, lov ryb, fosilní paliva, obnovitelné zdroje a řadu dalších klíčových faktorů pro přežití lidstva. Přesto je biodiverzita relativně neprobádaná. Podle odhadů známe jen zlomek druhů žijících na Zemi (je odhadováno jedno procento). Porozumění biodiverzitě a druhových distribucí je zásadní pro poznání, výzkum a ochranu přírody. Vztahy mezi živočichy a prostředím jsou složité, je velmi náročné jim porozumět. Dalším faktorem je nerovnoměrné uspořádání biologických jevů na povrchu planety Země, způsobené gradienty prostředí, rozložením kontinentů, antropogenními vlivy, přírodními katastrofami nebo mezidruhovými interakcemi (Boenigk a kol. 2015).

Michener & Jones (2012) popisují moderní ekologii jako vědu založenou na velkém množství dat, která se skládá z (i) empirických (popis přírodních jevů), (ii) teoretických (modelování a generalizace komplexní reality) a (iii) výpočetních (simulace) přístupů. Moderní ekologie je multidisciplinární vědou, která ve velké míře využívá výpočetní technologie.

2.1.1. Ekologická nika

Druhová ekologie se zabýva vztahem druhu a prostředí. Různé druhy a jedinci využívají různé části prostředí různými způsoby. Pro popis výřezu prostředí, které využívá jeden druh se v ekologii používá pojem ekologická nika nebo druhová nika . Ekologická nika je kvantifikací druhového habitatu (Storch & Mihulka 2000).

Ekologická nika, část prostředí obývaná jedním druhem, tvoří teoretický základ pro popis reakce druhu na prostředí. Existuje mnoho možností jak ekologickou niku

definovat. Pro moderní ekologii je zásadní popis ekologické niky Hutchinsona (1992), který ekologickou niku definuje jako mnohorozměrný environmentální prostor (n-dimensional hypervolume), jehož osy tvoří jednotlivé faktory prostředí, ve kterém je druh schopen přežít. Z principu Hutchinsonovské ekologické niky (matematický model niky) vychází modelování druhových distribucí (viz kapitola 2.2.1.) (Pearman a kol. 2008, Sillero a kol. 2021).

Mezi další důležité definice ekologické niky patří (i) Grinnellova (1917) definice neinteraktivní ekologické niky, zaměřující se na (abiotické) faktory prostředí (soubor požadavků druhu na podmínky a zdroje) a (ii) Eltonova (1927) definice interaktivní ekologické niky založená na dynamice mezidruhových (biotických) vztahů (funkce druhu v ekosystému). Z těchto dvou definic vychází rozdíl mezi základní (fundamentální) a skutečnou (realizovanou) ekologickou nikou. Fundamentální ekologická nika je souborem podmínek, které druh požaduje k přežití (pokud nejsou splněny nemůže druh v takovém prostředí dlouhodobě přežít). Realizovaná ekologická nika je část prostředí skutečně využívaná druhem nebo také podmnožina fundamentální ekologické niky. Je výsledkem mezidruhových interakcí (kompetice, predace a další) a geografických (fyzických) bariér (například pohoří nebo moře) (Boenigk a kol. 2015). Při překryvu fundamentálních nik dvou druhů může dojít ke konkurenčnímu vyloučení (posun realizovaných nik) (Townsend a kol. 2010).

Ekologickou niku můžeme popsat pomocí pozice a šířky. Tento přístup je založen na principu ekologické valence, rozmezí mezi hodnotami faktorů prostředí vhodných pro přežití druhu, která vychází z Schelfordova zákona tolerance. Pozice ekologické niky je určená optimem – stavy (hodnoty) faktorů prostředí, které jsou pro druh nejvýhodnější (největší fitness). Šířka ekologické niky je rozpětí hodnot faktorů prostředí, ve kterých je vyšší porodnost než úmrtnost. Podle šířky ekologické niky dělíme druhy na stenoekní (specialisté) a euryekní (generalisté) (Storch & Mihulka 2000).

Dalším přístupem pro popis ekologické niky je BAM diagram, který popisuje tři základní faktory ovlivňující druhovou distribuci v geografickém prostoru: (i) biotické (B), (ii) abiotické (A) a (iii) mobilita (M). Jednotlivé faktory jsou reprezentovány jako množiny. B je množina obsahující lokality s vhodnými faktory prostředí, jedná se o ekvivalent ekologické niky definované Hutchinsonem. Množina A je tvořena prostorem s žádoucími výskyty nebo absencemi dalších druhů. Množina M

reprezentuje oblasti, které jsou pro druh dostupné vzhledem k jeho aktuálnímu rozšíření a schopnostem pohybu (mobility nebo movement). Průnik všech tří množin je reprezentací skutečné geografické distribuce druhu, průnik množin A a B reprezentuje potenciální distribuci (Sobéron & Peterson 2005).

2.1.2. Konzervatismus ekologické niky

Wiens a kol. (2010) popisují konzervatismus ekologické niky jako zachování vlastností ekologické niky v čase. Podle Petersona (1999) je konzervatismus ekologické niky platný i v evolučním čase (stovky až miliony let). Wiens a kol. (2010) dále popisují možné využití konceptu konzervatismu pro ekologický výzkum, například pro poznání potravních řetězců, mezidruhových interakcí nebo tolerance druhu k náhlé změně prostředí. Někteří autoři s teorií konzervatismu ekologické niky nesouhlasí a zavádí pojem posunu ekologické niky (niche shift) (Rodder & Lotters 2009).

Pearman a kol. (2008) považují konzervatismus ekologické niky za předpoklad pro druhové distribuční modelování, současně ale připouští, že ekologická nika má dynamický charakter. Problémem je kvantifikace nebo prokázání konzervatismu nebo dynamiky ekologické niky, protože druhové nálezy jsou v podstatě vždy zatíženy nedůvěryhodností.

Peterson & Holt (2003) využívají druhové distribuční modely pro vyhodnocení vývoje ekologické niky. Vysvětlují možné příčiny změny modelované ekologické niky v čase, zmiňují (i) posun ekologické niky způsobený evolucí druhu, (ii) nedůvěryhodnosti ve vstupních lokalizacích druhů, (iii) různorodost částí zájmového území a (iv) nemožnost vyjádření některých faktorů ekologické niky, jako jsou mezidruhové interakce.

2.2. Nálezová data

Nálezová data jsou cenným zdrojem pro poznání lokální i globální biodiverzity a mezidruhových vztahů (Whittaker a kol. 2005). Nálezová data poskytují informaci o současných, ale i historických druhových distribucích. Zásadní výzvou je získání, digitalizace (historických nálezových dat) a distribuce nálezových dat pomocí prostorových databází (online druhové atlasy) a geografických informačních systémů (GIS) a jejich následná aplikace v praxi. Další zásadní otázkou je (ne)kvalita nálezových dat a jejich (ne)vhodnost pro danou praktickou aplikaci (viz kapitola 2.2.3.) (Chapman 2005, Robertson a kol. 2010).

Rozlišujeme různé typy nálezových dat (i) prezenční (presence-only) a (ii) prezenčně-absenční (presence-absence). U prezenčních dat se zaznamenávají pouze pozitivní výskyty. Prezenčně-absenční data obsahují informaci, zda se v dané lokalitě vyskytoval, nebo nevyskytoval. Prezenční a prezenčně-absenční data se liší způsobem sběru dat. Většina dostupných dat časoprostorových lokalizací výskytů druhů je prezenčního typu (Guillera-Arroita a kol. 2015).

2.2.1. Druhové distribuční modely

Druhové distribuční modely (species distribudion models – SDM), jeden ze základních nástrojů pro ekology a biogeografy, slouží ke kvantifikaci vztahu mezi druhovou distribucí a faktory přírodního prostředí (Miller 2010, Elith 2013). Můžeme se setkat s různými termíny pro distribuční modely, mezi které například patří modely klimatické niky, prediktivní modely habitatu nebo modely ekologické niky. Vztah druhu a prostředí lze využít pro předpověď druhové distribuce, která je v ekologické praxi například využitelná pro hodnocení dopadů klimatické změny (Thomas a kol. 2004), mapování invazních druhů a predikci jejich (potencionálního) rozšíření (Peterson 2004) nebo určení lokalit s potenciálně vysokou přírodní rozmanitostí (Liu a kol. 2013) a řadu dalších aplikací (Guisan & Zimmermann 2000, Guillera-Arroita a kol. 2015).

Vstupními daty modelů druhové distribuce jsou biologická a environmentální data. Biologická data jsou lokalizace druhových nálezů, které lze vyjadřovat jako prezence, absence, nebo početnosti (Miller 2010). Environmentální data jsou faktory prostředí zkoumaného druhu, mezi nejčastěji používané proměnné patří klimatické (teplota, vlhkost), topografické (sklon, nadmořská výška, orientace) a krajinné (typ krajinného

pokryvu) (Pearson 2007). Druhové distribuční modely v podstatě nezahrnují mezidruhové (biotické) interakce (pouze ve formě biologických vstupních dat – pozorované lokalizace druhů jsou ovlivněny mezidruhovým působením).

Distribuce druhu je rozložení v geografickém prostoru (na zemském povrchu). Model druhové distribuce v environmentálním prostoru (Hutchinsonova fundamentální nika) zobrazuje vhodnost (pravděpodobnost potenciálního výskytu druhu) všech možných kombinací faktorů prostředí pro daný druh (na základě nálezových dat). Hodnoty jsou následně z environmentálního prostoru přeneseny do geografického prostoru a vzniká mapa. V oblasti druhového modelování rozlišujeme pojmy mapa a model. Mapa je produktem a model je procesem (Miller 2010).

Druhové distribuční modely jsou založeny na matematickém vztahu závislých proměnných (lokalizace (ne)přítomností druhu) a vysvětlujících proměnných (faktory prostředí) (Miller 2010, Moudrý 2015). Pro distribuční modely je zásadní výběr vysvětlujících proměnných, na které druh skutečně reaguje a jejich rozlišení. Vhodné rozlišení je závislé na vlastnostech druhu (Araújo & Guisan 2006).

Guisan & Zimmermann (2000) popisují obecný postup pro modelování distribuce druhů. Modelování má šest postupných kroků: (i) návrh koncepce, (ii) získání a příprava dat, (iii) tvorba statistického/matematického modelu, (iv) úprava (kalibrace) modelu pro konkrétní zájmový druh, (v) předpověď prostorového rozložení a (vi) zhodnocení modelu.

Různé modelovací metody popisují Guisan & Zimmermann (2000), Elith & Leathwick (2009) a Miller (2010) a rozlišují mezi základnímy přístupy pro modelování druhových distribucí: (i) regresní, (ii) neparametrické a (iii) prezenční. Každý přístup má charakteristické vlastnosti, podle kterých je vhodný pro určitý typ vstupních dat nebo praktický účel modelování.

2.2.2. Původ nálezových dat

Nálezová data pocházejí z různých zdrojů (Gábor a kol. 2020, Petersen 2021). Zdroje můžeme dělit na (i) systematické (ii) náhodné (nesystematické) pozorování. Oba zdroje mají své uplatnění, ale i úskalí. Výhodou systematických průzkumů je kvalita dat a často prezenčně-absenční charakter. Cenou za kvalitu dat je náročnost získání, kvůli kterému mají systematické průzkumy omezené pokrytí (časové, prostorové nebo

taxonomické). Pro náhodné pozorování je hlavní výhodou kvantita dat, která ale přináší nedůvěryhodnosti a obvykle pouze prezenční záznamy.

Data pocházející z různých zdrojů jsou následně schromažďována v řadě online portálů, v případě druhových nálezových dat se obvykle používá pojem nálezová databáze (species occurrence database) nebo atlas druhů. Uživatel při využití nálezových dat z těchto portálů získá rozmanitou datovou sadu (pocházející z různých zdrojů), která trpí různými nedostatky (systematicky sbíraná data a data občanské vědy mají rozdílné předpoklady k nedostatkům). Vzhledem k rozmanitosti zdrojů je náročné tyto nedostatky kvantifikovat a odstranit, nebo alespoň mitigovat jejich vliv (Robertson a kol. 2010).

Jednotlivé nálezové databáze jsou zaměřeny na různou množinu druhů nebo lokalitu. Mezi významné nálezové databáze patří například GBIF – data o výskytech rostlin, zvířat, hub a mikroorganismů z celého světa, NatureServer – data o rostlinách, živočiších a ekosystémech Severní a Jižní Ameriky, eBird – globální data lokalizací ptáků nebo Nálezová databáze ochrany přírody – výskyty rostlin, zvířat a hub na území ČR (Miller 2010, Chobot a kol. 2011).

Chandler a kol. (2016) popisují příspěvek občanské vědy (citizen science) k poznání biodiverzity. Občanská věda je v případě nálezových dat sběr dat veřejností. Hlavní výhodou občanské vědy je získání velkého množství dat za nízké náklady. Nevýhodou je neurčitost nebo nedůvěryhodnost kvality takto získaných dat.

2.2.3. Nedůvěryhodnosti nálezových dat

Nálezová data jsou častěji získávána náhodně než systematicky, což způsobuje nedůvěryhodnosti. Mezi základní patří (i) geografická nerovnoměrnost vzorkovánív (sampling bias nebo geographic bias), (ii) špatná identifikace druhu (misidentification) a (iii) polohová chyba (positioning error) (Rondinini a kol. 2006).

Druhové záznamy pocházející z náhodných pozorování nebo muzejních záznamů jsou často zatíženy sampling biasem. Tyto záznamy jsou nejčastěji soustředěny v oblastech, které jsou hojně navštěvovány, díky lepší dostupnosti (například vzdálenost od komunikací, ale i svažitost nebo hustota porostu) nebo atraktivitě lokality pro pozorovatele (Fourcade a kol. 2014). Nadměrné vzorkování obvykle probíhá v určitých částech zájmového území, například v blízkosti cest, řek, osídlených oblastí, ale i nížin nebo pobřeží a v dalších lokalitách (Chauvier a kol. 2021). Pokud není

v druhových distribučních modelech nedůvěryhodnost zohledněna, mohou výsledné distribuční mapy reprezentovat spíše prostorové rozložení úsilí průzkumu (survey effort) (Phillips a kol. 2009). Záznamy získané z plánovaných (systematických) průzkumů (inventarizací) jsou podle Fourcada a kol. (2014) obvykle spolehlivější. Podle Støa a kol. (2018) má (ne)kvalita vstupních nálezových dat zásadnější vliv na výsledky druhových distribučních modelů než výběr modelovací metody.

2.3. Dálkový průzkum Země v optické části spektra

Nejpřesnější data pro odhadování biodiverzity jsou terénní pozorování a sběr dat (in situ), problémem této metody je neudržitelnost a částečná nevěrohodnost (viz kapitola 2.2.3.). Dálkový průzkum Země umožňuje kontinuální a konzistentní sběr environmentálních dat v pravidelném intervalu, které mají velký potenciál jako zdroj dat pro mapování biodiverzity a druhových distribucí. Pro ekologický výzkum a praxi je zásadní fúze dat získaných v terénu a dálkového průzkumu Země, kvůli rozdílným vlastnostem získávání dat. Dálkový průzkum Země není vhodné pro zkoumání některých mikroekologických jevů, ale pro sledování jevů větších měřítek je vhodnější než in situ pozorování (Gillespie a kol. 2008, Nagendra 2001, Rocchini 2021).

2.3.1. Princip optického DPZ

Dálkový průzkum Země je pořizování informace bez přímého kontaku se zkoumaným prvkem. Bezkontaktní získávání informace je založeno na působení zkoumaného objektu na své okolní silové pole (gravitační, elektromagnetické, magnetické a další). Tato silová pole je možné dálkově sledovat měřením přicházejícího záření (daného typu) pomocí senzoru. Senzory dálkového průzkumu Země jsou umístěny na různých nosičích (satelity, letadla, drony). Pro globální datové pokrytí jsou zásadní satelitní systémy, díky často opakovaným přeletům nad jedním místem na zemském povrchu (vysoké časové rozlišení viz kapitola 2.3.3). Pro dálkový průzkum Země je využíváno elektromagnetické záření, dle zdroje záření rozlišujeme (i) aktivní (s umělým zdrojem záření) a (ii) pasivní (s přirozeným zdrojem záření) dálkový průzkum Země. Elektromagnetické záření se šíří formou elektromagnetického vlnění, složeného z elektrické a magnetické vlny (tyto vlny svírají pravý úhel). Elektromagnetické vlnění lze popsat pomocí základních veličin (i) vlnová délka (λ [m]) a (ii) frekvence (f [Hz]), elektro magnetické vlnění se pohybuje rychlostí světla (c [m/s]). Dle těchto veličin

(vlnové délky a frekvence) lze elektromagnetické záření rozdělit do spektra od dlouhých rádiových vln až po krátké vlny gamma (Vysoudil 1993)..

Elektromagnetické záření není ovlivňováno pouze zkoumaným objektem. Při průchodu atmosférou (při cestě k zemskému povrchu nebo zkoumanému objektu a následné cestě ke snímači), dochází ovlivnění elektromagnetického záření. V atmosféře dochází k (i) rozptylu (scatter), (ii) pohlcování (absorption) a propustnosti (transmission). Rozptyl je způsoben drobnými částicemi, které změní směr elektromagnetického záření. Atmosférický rozptyl v dálkovém průzkumu Země způsobuje například zesílení krátkých optických vlnových délek (Rayleighův rozptyl), šum snižující kvalitu snímku (Mieův rozptyl) nebo bílou barvu mraků a jejich neprostupnost v optické části elektromagnetického spektra (neselektivní rozptyl). Pro odstranění vlivů atmosféry jsou využívány atmosférické korekce, které jsou zásadní při analýzách různých míst nebo různých časech (bez atmosférických korekcí by různý stav atmosféry ovlivnil výsledky). Produkty dálkového průzkumu Země dělíme dle atmosférických korekcí na (i) snímky s hodnotami odrazivosti na vrcholu atmosféry (top of atmosphere – TOA) a (ii) snímky s hodnotami odrazivost na zemském povrchu (surface reflectance – SR nebo bottom of atmosphere – BOA). Snímky SR jsou atmosféricky korigované, snímky TOA naopak nejsou atmosféricky korigované. Mimo atmosférické korekce obrazu dálkového průzkumu Země jsou prováděny (i) radiometrické korekce, které se využívají pro vyrovnání naměřených hodnot při sezónních rozdílech (rozdílná výška Slunce v průběhu roku) nebo při radiometrických chybách a (ii) geometrické korekce snímků jsou potřebné při kolísání nosiče nebo například vlivech atmosféry (Kropáček a kol. 2020).

Optický dálkový průzkum Země využívá od viditelných po středně dlouhé infračervené vlnové délky, zdrojem záření v této části spektra je Slunce, tudíž se jedná o pasivní formu dálkového průzkumu Země. Část elektromagnetického spektra využívanou optickým dálkovým průzkumem Země dále rozdělujeme na takzvaná pásma (band) nebo kanály (channel), ze kterých se následně skládá multispektrální snímek. Optický dálkový průzkum Země je založen na interakci slunečního záření a zkoumaného objektu, který jednotlivé vlnové délky záření (pásma) různě pohlcuje a odráží, pro každý typ objektů lze vytvořit takzvanou spektrální křivku (spectral reflectance curve), která znázorňuje závislost odrazivosti na vlnové délce. Spektrální křivka (charakteristika) je pro každý typ povrchu jedinečná (Pavelka 1999).

2.3.2. Spektrální charakteristiky jako popis (přírodního) prostředí

Environmentální prostorové analýzy často využívají předzpracovaná data vytvořená ze snímků dálkového průzkumu Země – indikátory (Madani a kol. 2016). Při tvorbě těchto datových sad ze surových (satelitních) snímků obvykle dochází k částečné ztrátě informace. Nejvíce informace obsahuje právě neklasifikovaný (satelitní) snímek ve formě hodnot (digital number – DN) jednotlivých pixelů v daných spektrálních pásmech (Conner 2002). Laurent a kol. (2005), Pasher a kol. (2007), Estes a kol. (2008) a Stickler & Southworth (2008) se zabývají využitím surové spektrální informace multispektrálních družicových snímků pro popis druhových habitatů (samostatné nebo v kombinaci s konvenčně využívanými environmentálními proměnnými). Častými metodami jsou (i) využití vybraných pásem nebo indexů, které reagují na určité části přírodního prostředí (například index vegetace - NDVI) a (ii) redukce dimenzionality spektrální informace (například PCA).

Sickler & Southworth (2008) na příkladu modelování distribuce primátů (*Cercopithecus ascanius*) v národním parku Kibale (Uganda) demonstrují možnost využití multispektrálních družicových snímků jako prediktorů distribuce porovnáním využitelnosti multispektrálních družicových snímků Landsat a Quickbird. Srovnáním snímků Landsat a Quickbird je demonstrován tradeoff mezi spektrálním a prostorovým rozlišením, zabývají se vlivem rozlišení snímků na model. Poukazují na význam spektrálního rozlišení snímků, které považují za vlivnější než prostorové rozlišení. Předpokladem je, že spektrální hodnoty jsou ukazatelem vlivu lidské (antropogení) činnosti.

Pasher a kol. (2007) vytvořili model potenciálního habitatu lesňáčka kápového (*Wilsonia citrina*), který vychází z výzkumu struktury porostu v místech hnízdění lesnáčka. Předpokladem je možnost získání informace o struktuře lesního porostu pomocí rozdílných spektrálních charakteristik. Ukázala se vhodnost mapování menších lokalit s využitím snímků s vysokým prostorovým rozlišením (Ikonos, Quickbird), regionální mapování založené na datech Landsat mělo slabší výsledky.

Laurent a kol. (2005) se zabývali využití snímků Landsat pro predikci distribucí několika druhů ptáků (Seiurus aurocapilla, Leiothlypis ruficapilla a Setophaga virens). Modely všech druhů dosáhly hodnot Kappa > 0.3, Kappa je metrikou kvality distribučního modelu modelu, která nabývá hodnot od 0 (náhodná shoda s realitou) do

1 (dokonalý model) (Allouche a kol. 2006). Dále je poukazováno na vliv velikosti mapované buňky, která by ideálně měla odpovídat rozloze teritoria zkoumaného druhu.

Estes a kol. (2008) zkoumají kombinaci dat dálkového průzkumu Země a dat pořízených v terénu pro popis habitatu antilopy bongo (*Tragelaphus eurycerus*). Data dálkového průzkumu Země považují za vhodné pro popis větších částí habitatu, které je navíc náročné mapovat in situ metodami.

Rocchini a kol. (2021) považují pásma multispektrálních družicových snímků jako možné osy Hutchinsonova mnohorozměrného konceptu ekologické niky pro účel modelování druhových distribucí.

2.3.3. Rozlišení (měřítko)

Rozlišení dat dálkového průzkumu Země, ale i ostatních (geo)dat je zásadním faktorem pro prostorové analýzy. Při nevhodně zvoleném měřítku dochází k chybám nebo nepřesnostem ve výstupech analýz. Je nutné vyhodnotit, zda jsou data v daném rozlišení vhodné pro záměr analýzy. Rozlišení může pro danou analýzu jak příliš podrobné, tak příliš hrubé. Problematika rozlišení dat je nevyhnutelná vzhledem k nutnosti generalizace skutečného světa (Degbelo a kol. 2018). Vhodně zvolené rozlišení nebo měřítko dat má velký vliv na výstup (Longley a kol. 2005).

Pro popis dat dálkového průzkumu Země bereme v úvahu 4 druhy rozlišení: (i) spektrální (spectral), (ii) časové (temporal), (iii) prostorové (spatial) a (iv) radiometrické (radiometric). Spektrální rozlišení je rozsah snímaných vlnových délek a citlivost snímače. Časové rozlišení je doba, po které daný systém dálkového průzkumu Země snímá stejné místo na zemském povrchu. Na prostorové rozlišení lze pohlížet dvěma způsoby: (i) velikost jednoho pixelu na povrchu Země, někdy se označuje jako velikost zrna (grain size) a (ii) rozsah jednoho snímku na zemském povrchu (extent). Radiometrické rozlišení udává rozsah hodnot, kterých může jeden pixel nabývat, běžnou hodnotou je 256 hodnot (hodnoty 0-255, což odpovídá 8 bitům) (Vysoudil 1993).

Vliv nevhodně zvoleného rozlišení dle druhu popisuje Nagendra (2001). Popisuje závislost mezi velikostí zrna a rozsahem, s rostoucím rozsahem klesá udržitelná míra zachyceného detailu, naopak s menším rozsahem roste možná míra detailu. Prostorové rozlišení musí odpovídat rozsahu zkoumaného jevu. Při hrubém rozlišení hrozí, že

zkoumaný jev nebude zachycen v dostatečném detailu, při příliš vysokém rozlišení může být množství informace překážkou pro kvalitu analýz. Výzvou při zpracování dat s vysokým spektrálním rozlišením je výběr pásem, která jsou vhodné pro popis zkoumaného jevu. U časového rozlišení je potřeba brát ohled na srovnatelnost (například fáze fenologické sezóny). Dalším otázkou u časového rozlišení je, zda je dostatečné pro sledování daného jevu.

3. Metodika

3.1. Využitý software

3.1.1. Google Earth Engine

Google Earth Engine (GEE) je cloudová výpočetní platforma pro prostorové analýzy. Skládá se z (i) výpočetních serverů a jim přizpůsobených tříd a funkcí - od základních matematických operací až po strojové učení, (ii) online knihovny (geo)dat - především satelitních snímků, ale i vektorových datasetů (například polygony států). Online knihovna obsahuje petabity prostorových dat a každý den jsou nahrávány nové, Google Earth Engine umožňuje nahrání vlastních dat z lokálního do webového (serverového) prostředí, (iii) webového vývojového prostředí a rozhraní pro programování aplikace (API) pro několik programovacích jazyků (Gorelick a kol. 2017).

Výhodou Google Earth Engine je snadný přístup k výpočetním zdrojům (s potřebným výkonem) pro zpracování globálních prostorových datových sad a samotným (geo)datům bez nutnosti řešení základních obtíží (nedostatečná kapacita - výpočetní nebo paměti, získání dat), které vznikají při lokálním zpracováním velkých (prostorových) datasetů. GEE umožňuje provádět časoprostorové analýzy v globálním měřítku, jako například globální změna lesního porostu (Hansen a kol. 2013) a globální změna povrchové vody (Pekel a kol. 2016).

Google Earth engine usnadňuje přístup k vybraným datovým sadám, hlavní výhodou je možnost analýz bez potřeby stahování dat a s omezením potřebné lokální výpočetní kapacity.

3.1.2 Programovací jazyk Python

Programovací jazyk Python (verze 3.8.*) je jedním z nejpopulárnějších programovacích jazyků současnosti. Příčinou popularity je flexibilita programovacího jazyka a uživatelská přívětivost (rychlost a menší náročnost vývoje). Flexibilita jazyka Python je velkou výhodou. Díky velkému množství knihoven (balíčků) lze Python využívat pro řadu různých typů aplikací od strojového učení a umělé inteligence po vývoj webových stránek nebo desktopových aplikací. Programovací jazyk Python jsem pro tvorbu nástroje pro výzkum spektrálních charakteristik druhového habitatu zvolil, protože umožnuje vývoj všech částí nástroje.

Pro editaci a tvorbu skriptů jsem využil vývojové prostředí (IDE) Spyder 5, které má funkci nejen textového editoru, ale i interaktivní konzole. Umožnuje i spouštění příkazů příkazového řádku v iPython konzoli.

Pro tvorbu a správu virtuálního prostředí (virtual environment) jsem využil správce prostředí Anaconda. Virtuální prostředí umožnují snadnou práci s různými instalacemi (verzemi) programovacího jazyku a instalaci a aktualizace potřebných knihoven – zajištění kompatibility.

Vytvořil jsem virtuální prostředí s potřebnými knihovnami. Hlavní knihovny využité pro nástroj jsou *GeoPandas*, *ndop-downloader*, *earthengine-api*, *PyQt5*, *Pandas*, dále správce prostředí knihovny požadované pro funkci těchto balíků (dependencies). Identické virtuální prostředí lze vytvořit na základě souboru *requirements.txt* (příloha č. 5), který je součástí příloh práce, pomocí příkazu (Obr. 1) v prostředí příkazového řádku (Command Prompt). Pro tvorbu virtuálního prostředí je potřeba nainstalovaná aplikace Anaconda.

```
conda create --name benv --file requirements.txt
```

Obr. 1: Příkaz pro tvorbu virtuálního prostředí z souboru requirements.txt

3.1.3. Qt Desinger

Qt Desinger je open-source (licence GNU GPL 3, volné užití pro nekomerční využití) nástroj pro tvorbu grafického uživatelského rozhraní (GUI), které je vytvářeno v grafickém prostředí. Tvorba grafického rozhraní v Qt Desingeru je uživatelsky přívětivější než ruční programování (možná na úkor možnostem přizpůsobení). V prostředí Qt Desinger jsou postupně vytvářeny jednotlivé funkční prvky nástroje pomocí přetahování položek z menu.

Qt Desinger umožnuje nejen tvorbu, ale i nastavení jednotlivých funkčních prvků (název, možné hodnoty, propojení prvků, atd.). Dále lze grafické rozhraní nastylovat pomocí qss sylování, jedná se o nastavení vzhledu prvků ve způsobu přirovnatelném k css (kaskádové webové styly).

3.2. Data

3.2.1. Data Nálezové databáze ochrany přírody

Druhová nálezová data jsou pro účel nástroje získávána z Nálezové databáze ochrany přírody. Databáze je spravována Agenturou ochrany přírody a krajiny (AOPK) a je součástí Informačního Systému Ochrany Přírody (ISOP). Databáze byla vytvořena v roce 2006 na základě potřeby sběru dat a tvorbu výstupů pro evropské směrnice o stanovištích (92/43/EHS) a o ptácích (79/409/EHS) vzhledem ke vstupu ČR do Evropského společenství (Hošek a kol. 2008). Obsahem databáze jsou záznamy časoprostorových lokalizací druhů hub, rostlin a živočichů (Chobot a kol. 2011). Databáze celkem obsahuje přes 27 milionů záznamů, které pocházejí z různých zdrojů (inventarizační průzkumy, digitalizace historických dat, záznamy z vědeckého výzkumu, ale i občanská věda). Většina záznamů pochází od odborníků na daný druh, kteří plní zadání AOPK ČR (Chobot a kol. 2018). Různorodost zdrojů dat lze považovat za předpoklad pro nepřesnosti.

Nálezová databáze ochrany přírody prošla od svého vytvoření v roce 2006 dlouhým vývojem. Významným mezníkem byl přechod ze smluvního poskytování expertům a úřadům na poskytování dat veřejnosti formou otevřených dat (open data). Zásadním krokem pro otevřenost dat bylo umožnění veřejného prohlížení dat přes webové rozhraní aplikace BioLog (Chobot a kol. 2018).

AOPK ČR využívá Nálezovou databázi ochrany přírody pro různé účely příkladem je tvorba analytických podkladů, koncepce druhové ochrany (Hošek a kol. 2008) nebo aktuální projekt zaměřený na monitoring nepůvodních druhů (Chobot a kol. 2021).

Nálezová data neobsahují pouze informaci o poloze nálezu, ale řadu dalších atributů. Atributy, které nástroj využívá jsou (i) časová informace o datu (*DATUM_OD*), (ii) informace o polohové spolehlivosti (*CXPRESNOST*), která v metrech udává přesnost nálezu. Atribut přesnosti může být zavádějící, protože nejnižší hodnota (0) u nálezů NDOP často znamená neznámou hodnotu a velmi nízké hodnoty často neodpovídají realitě, vzhledem k přesnosti GNSS přijímačů.

Data lze z Nálezové databáze ochrany přírody exportovat v textových formátech nebo ve vektorovém formátu shapefile (shp). Stahování nálezových dat z webu (https://portal.nature.cz/nd/) je pro uživatele omezené, protože v textovém formátu je umožneno stahování omezeného počtu záznamů (maximálně 1000) a lokalizace ve

vektorovém formátu jsou pouhými lokalizacemi bez ostatních atributů. Řešení tohoto problému je QGISový plugin/Python balík *ndop-downloader*, který tento limit obchází pomocí algoritmizace stahování dat.

3.2.2. Multispektrální družicové snímky Landsat

Landsat je družicový systém zřizovaný NASA (National Aeronautics and Space Administration) a USGS (United States Geological Survey). První družice Landsat 1 byla spuštěna v roce 1972, postupem času bylo vypuštěno dalších 8 družic Landsat. Nejnovější je družice Landsat 9, spuštěná v září 2021.

Pro získání spektrálních charakteristik jsou aplikací využívány multispektrální družicové snímky misí Landsat 4, Landsat 5, Landsat 7 a Landsat 8. Důležitou vlastností programu Landsat je zpětná návaznost dat, díky stálosti parametrů snímání mezi jednotlivými misemi (rozlišení a rozložení pásem). Landsat poskytuje dlouhou časovou řadu multispektrálních družicových snímků s dostatečným rozlišením pro mapování přírodních habitatů. Výhodou délky časové řady snímků Landsat je možnost sledování reakce druhů na změnu prostředí (pokud jsou dostupná i nálezová data) (Oeser a kol. 2019).

Snímače Thematic Mapper (TM) na Lansatu 4 a Landsatu 5, Enhanced Thematic Mapper Plus (ETM+) na Landsatu 7 a Operational Land Imager (OLI) na Landsatu 8 snímají v srovnatelných základních optických pásmech (modré, zelené, červené, blízké-infračervené a dvě krátkovlnné-infračervené) v prostorovém rozlišení 30 metrů. Časové rozlišení systémů Landsat je 16 dní. Spektrální a radiometrické rozlišení je mezi Landsaty rozdílné. Novější senzory snímají ve větším spektrálním rozsahu (více pásem) a mají vyšší radiometrické rozlišení.

3.3. Návrh aplikace

Praktická část práce se zaměřuje na tvorbu nástroje sloužícího k získání, vizualizaci a vyhodnocení charakteristiky druhového habitatu, libovolného uživatelem zvoleného druhu z Nálezové databáze ochrany přírody. Základem nástroje jsou (i) uživatelké grafické rozhrání obstarávající uživatelské vstupy a zobrazení vypočtených hodnot, (ii) výpočetní algoritmus pro získání a zpracování dat a (iii) skriptu pro propojení grafického rozhraní aplikace a algoritmu pro výpočet spektrálních charakteristik, tvorbu vizualizací a export dat.

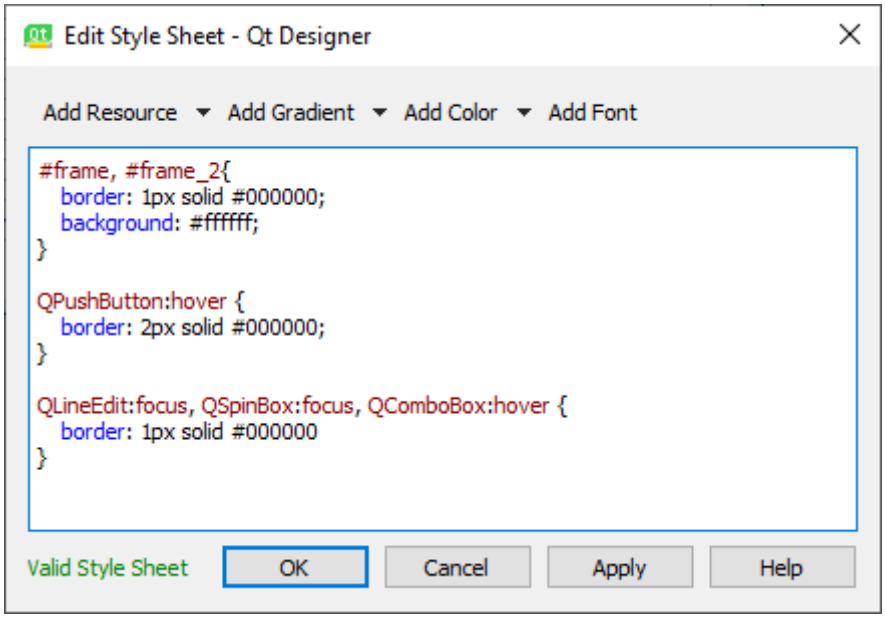
Uživatel nástroje nejprve má možnost zvolit druh, který chce analyzovat pomocí textového pole, do kterého zadá název druhu. Následně proběhne vytěžení dat z nálezové databáze, poté uživatel zvolí parametry pro (i) filtrování nálezů a (ii) výpočet spektrální charakteristiky. Nálezy se vyfiltrují dle atributu polohové přesnosti. Uživatel vybere z možných filtrů, následně zadá hodnotu, nebo hodnoty podle, kterých proběhne filtr. Druhým nastavením je volba časového období. Uživatel aplikace zadává dvě hodnoty roků (počátek a konec) a hodnotu kroku, ze kterých je následně vytvořena posloupnost počátečních roků jednotlivých období pro srovnání. Tyto parametry vstupují nejen do filtru nálezů, ale i do algoritmu pro výpočet spektrální charakteristiky v cloudovém prostředí Google Eearth Engine. Po zadání parametrů je spuštěn výpočet, následně jsou získané hodnoty vizualizovány. Uživatel aplikace má možnost exportovat (i) vizualizace vývoje spektrální charakteristiky druhového habitatu a (ii) vypočtené spektrální hodnoty jednotlivých nálezů.

Nástroj je tvořen třemi základními pythonovými skripty (i) skript definující grafické uživatelské rozhraní – *gui.py* (příloha č. 2), (ii) skript s hlavními funkcemi – *methods.py* (příloha č. 3) a (iii) spouštecí skript propojící celou nástroj – *main.py* (příloha č. 1).

3.3.1. Grafické uživatelské rozhraní (GUI)

Grafické uživatelské rozhraní nástroje jsem vytvořil v prostředí Qt Desinger (viz kapitola 3.1.3). Grafické rozhraní je složeno z hlavního okna (*QMainWindow*), ve kterém jsou záložky (*QStackedWidget*). Přes záložky uživatel postupně prochází a zadává jednotlivé parametry. Na vytvořené záložky jsem vložil jednotlivé funkční prvky (tlačítka, textová pole a další). V prostředí Qt Desinger jsem provedl základní nastavení prvků (nastavení názvů, povolených hodnot).

Po vytvoření základních funkčních prvků jsem upravil vzhled uživatelského grafického rozhraní pomocí rozložení (*QLayout*), které umožňuje ukotvení prvků (důležité při změně velikosti okna) a qss stylování (Obr. 2).



Obr. 2: Vytvořené stylování hlavního okna pomocí qss

Vytvořený .ui soubor jsem následně převedl do formátu .py. Převod je založen na knihovně *PyQt5* a je spouštěn v příkazovém řádku (Obr. 3). Po tomto převodu je možné propojení (pomocí importu) s dalšími python skripty.

```
python -m PyQt5.uic.pyuic -x gui.ui -o gui.py
```

Obr. 3: Příkaz pro převod .ui souboru na .py soubor

Následně jsem vytvořil další .ui soubor, který slouží jako vyskakovací okno (*QDialog*) chybových hlášek. Postup pro vytvoření souboru je obdobný jako pro tvorbu hlavního okna.

3.3.2. Algoritmus pro získání spektrálních charakteristik (methods)

Algoritmus pro výpočet spektrálních charakteristik druhové niky (*methods.py* – příloha č. 3) je založen na několika python knihovnách a na parametrech specifikovaných uživatelem. V zásadě se skládá z dvou funkcí: (i) funkce pro získání nálezových dat z Nálezové databáze ochrany přírody (*getOccurrenceData*) a (ii) funkce pro výpočet charakteristiky spektrální niky (*getSpectralValues*).

Funkce getOccurrenceData

Funkce *getOccurrenceData* je založená na Python knihovně *ndop-downloader* (verze 0.1.8). Na základě vstupního řetězce druhového názvu, získaného z textového pole (*QLineEdit*), funkce vrací dataset nálezů z Nálezové databáze ochrany přírody ve formě *GeoDataFrame* (objekt knihovny *geopandas*), jedná se o objekt reprezentující tabulková data (pandas znamená "panel dataset"), která mají geometrickou informaci (geometrii). Pro implemetaci této funkcionality jsem upravil funkci *get_ndop_csv_data* (knihovna *ndop-downloader*), která ve své původní formě dataset nálezů ukládá ve formátu .csv na lokální disk. Vytvořil jsem funkci *getNdopData,* která nálezový dataset vrací jako pole, naplněné polemi – první obsahuje názvy atributů, následují pole s nálezy (po 500 nálezech).

Po zavolání funkce *getNdopData* jsou vybrány potřebné sloupce (*DATUM*_OD, *CXPRESNOST*, *X* a *Y*), které jsou následně převedeny do číselných datových typů (*float* nebo *int*). Poté je vytvořen *GeoDataFrame,* který je transformován do projekce WGS84 (pomocí metody objektu typu *GeoDataFrame* "*.to_csrs('epsg:4326')*"). Vytvořený objekt je funkcí vrácen (return).

Funkce getSpectralValues

Funkce *getSpectralValues* slouží k výpočtu spektrální charakteristiky druhového habitatu. Pro výpočet využívá funkcionality cloudového nástroje Google Earth Engine, ke kterému přistupuje pomocí knihovny *earthengine-api* (*ee*). Další využívanou knihovnou je *GeoPandas*. Tato knihovna je potřebná, protože jedním ze vstupů funkce *getSpectralValues* je *GeoDataFrame* (vytvořený pomocí funkce *getOccurrenceData*). Dalšími vstupy jsou parametry pro filtrování (typ filtru, hodnota, nebo hodnoty pro filtrování) a parametry časového období (počáteční rok, konečný rok, krok). Vstupní hodnoty jsou získány z prvků grafického rozhraní typu *QComboBox* a *QSpinBox*.

Nejprve je vytvořena kolekce (cloudový objekt Google Eearth Engine *ee.ImageCollection*) atmosféricky korigováných snímků Landsat, které jsou filtrovány dle měsíců od května do srpna. Atmosféricky korigované snímky s hodnotou odrazivosti na zemském povrchu jsou využity z důvodu časoprostorového charakteru prováděné analýzy. Optické satelitní snímky jsou v našich podmínkách (zeměpisných šířkách) často z velké části pokryty oblačností, která způsobuje rozptyl elektromagnetického záření (viz kapitola 2.3.1.). Pro účel odstranění oblačnosti jsem vytvořil funkci pro maskování oblačnosti *maskClouds*, která je aplikována na všechny

snímky pomocí cyklu na straně serveru (*map*). Maska pro odstanění oblačnosti je založena na pásmu pixel_qa, které obsahuje informaci o kvalitě pixelu (například zda v daném pixelu je oblačnost nebo její stín). Jednotlivá pásma jsou přeřazena a přejmenována pro jednotnost mezi snímky z ruzných misí Landsat.

Následuje filtr nálezových dat podle parametrů filtru (uživatel aplikace má take možnost data nefiltrovat). Uživatel filtruje dle atributu polohové přesnosti (*CXPRESNOST*). Na základě parametrů časového období je vytvářena řada počátečních let pro každé období. Řada je ukládána jako pole, které je následně procházeno lokalním cyklem (*for*). V cyklu probíhá (i) filtrace nálezových dat dle časového období, (ii) tvorba satelitní mozajky pro dané období a (iii) získání hodnot jednotlivých pásem.

Satelitní mozaiky jsou také tvořeny na základě sekvence počátečních let. Podle let je kolekce snímků Landsat filtrována pomocí metody *ee.FilterDate*, jejíž výstupem je nová kolekce, na kterou je zavolána metoda *median*. Metoda *median* z kolekce vytvoří jeden snímek (*ee.Image*) s hodnotami pixelů vypočtenými jako medián prostorově se překrývajích pixelů.

Nálezy jsou ukládány v *GeoDataFramu*, ten lze filtrovat pomocí řezů a metody *isin*, která postupně prochází jednotlivé prvky a rozhoduje, zda hodnota daného atributu (*DATUM_OD*) je obsažena v zadaném poli (pole je parametrem metody *isin*). Z důvodu omezení maximálního počtu prvků (5000), které lze nahrát do GEE, je potřeba prvky nahrávat postupně pomocí lokálního cyklu (*for*). Nálezy jsou v cyklu řezány do podmnožin obsahujících maximálně 5000 prvků a jsou nahrávány do cloudové proměnné typu (*ee.FeatureCollection*).

Vytvořené kolekci prvků se přiřazují hodnoty snímku pomocí metody *sampleRegions*. Výstupem je nová kolekce s přiřazenými atributy hodnot jednotlivých pásem. Na kolekci se spektrálními hodnotami je zavolána funkce *getDownloadURL*, která vrací odkaz pro stažení kolekce. Odkaz je využit jako parameter funkce *read_csv* (knihovna *pandas*). Výstupy se postupně (během cyklu) ukládají do knihovny (*dictonary*), která je konečným výstupem funkce *getSpectralValues*. Vzhledem k tvorbě mozaiky pomocí masky může dojít k situaci, že pro daný bod nebudou dostupné hodnoty pixelu. V takovém případě daný bod není použit. Tato situace nastává především v období mezi lety 1980 a 1985, protože kolekce Landsat v tomto období obsahuje snímky

Landsat 4 od roku 1982 a Landsat 5 od roku 1984. Problém může nastat hlavně při nastavení nižších hodnot kroku, například při počátečním roku 1980 a kroku 1 nebo 2 nebude vytvořena první, popřípadě první dvě mozaiky.

3.3.3. Hlavní spouštecí skript (main)

Spouštěcí skript (*main*.py – příloha č. 1) má na starosti: (i) propojení grafického rozhraní a hlavních výpočetních funkcí (*getOccurrenceData* a *getSpectralValues*), (ii) vizualizaci a možnost exportu dat.

Grafickému rozhraní je ve spouštěcím sktiptu vytvořena funkcionalita pomocí spouštění funkcí na základě signálů, které emitují funkční prvky. Základní funkce jsou vytvořeny přímo ve spouštecím skriptu. Hlavní výpočetní funkce jsou ve vedlejším souboru (*methods.py* – příloha č. 3), který je do spouštícího skriptu importován. Vzhledem k časové náročnosti hlavních výpočetních funkcí je implementováno spouštění těchto funkcí na druhém vlákně pomocí objektu *QThread* (účel této implementace je nezamrzání nástroje při spouštění funkcí). Pro větší uživatelskou přívětivost je po dokončení hlavních výpočetních funkcí spuštěno upozornění na hlavním panelu (problikávání ikony). Pro účel spuštění funkce na druhém vlákně je vytvořen objekt (třídy *WorkerOccurrences* a *WorkerSpectral*), který při vytvoření (*__init__*) jako parametry přijímá potřebné hodnoty pro funkci, a následně je zavolána metoda *run*, která spustí danou funkci.

Do hlavního skriptu je importován soubor pro okno dialogu (*dialog.py*), které je zobrazeno při chybách v zadaných hodnotách nebo neproběhnutých funkcích. Zobrazení probíhá pomocí vytvoření objektu třídy *Dialog* a zavoláním jeho metody *show*.

Vizualizace je spuštěna po dokončení druhé hlavní výpočetní funkce. Vizualizace jsou vytvářeny pomocí knihovny *matplotlib*. Vytvářejí se tři grafické výstupy. Každý výstup má vlastní funkci pro svou tvorbu (z důvodu lepší přehlednosti): (i) krabicový graf s možností volby vizualizovaného pásma (*visualizeData*), (ii) polární graf (*visualizeData2*) a (iii) mřížka obsahující krabicové grafy všech šesti pásem (*visualizeData3*). Každý graf má své ovládací prvky (funkcionalita *NavigationToolbar*), součástí ovládacích prvků je možnost export grafického výstupu. U krabicových grafů jsou na ose x vyneseny počáteční roky jednotlivých období a na

ose y hodnoty odrazivosti. Polární graf má na obvodu jednotlivá spektrální pásma a jako faktory počáteční roky.

Export je zajišťován pomocí *Qt* objektu *QFileDialog*. Při zavolání otevře okno prohlížeče souborů, ve kterém uživatel prochází adresářem, následně zadá název souboru. Cesta k souboru je vrácena pomocí metody *getSaveFileName*. Slovník se spektrálními hodnotami je následně pomocí metody *to_excel* (knihovna *pandas*) zapsán do excelového souboru. Na listech vytvořeného excelového souboru jsou jednotlivá období.

Finální spustitelný soubor nástroje (.exe) jsem vytvořil pomocí Python knihovny pyinstaller, spuštením příkazu v příkazovém řádku (Obr. 4).

```
!pyinstaller main.py -F --distpath . --debug=all -n"NSNA" -i"iconEdit.ico"
```

Obr. 4: Příkaz pro vytvoření .exe souboru z .py souboru

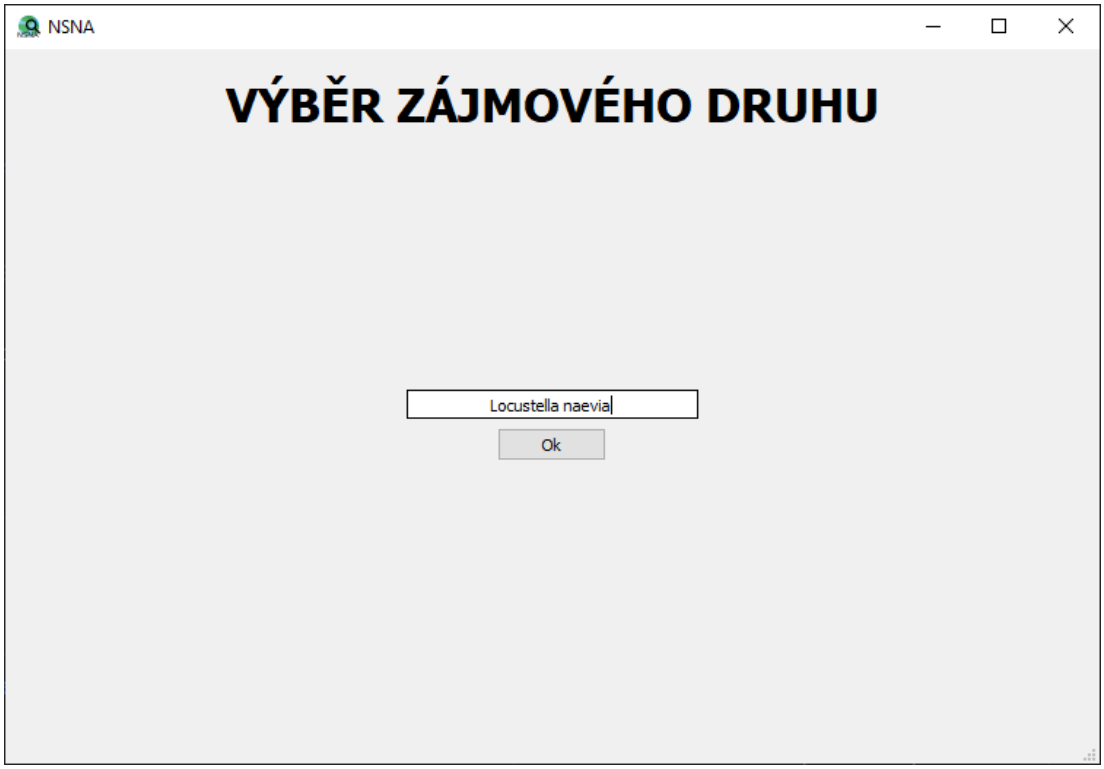
4. Výsledky

4.1. Nástroj pro výzkum spektrální charakteristiky

V praktické části jsem vytvořil nástroj pro výzkum spektrálních charakteristiky druhového habitat. Nástroj je navržen pro analýzu nálezových dat z Nálezové databáze ochrany přírody (NSNA – NDOP spectral niche analysis). Pro aplikaci jsem vytvořil ikonu (Obr. 5) ve vektorovém grafickém editoru InkScape.

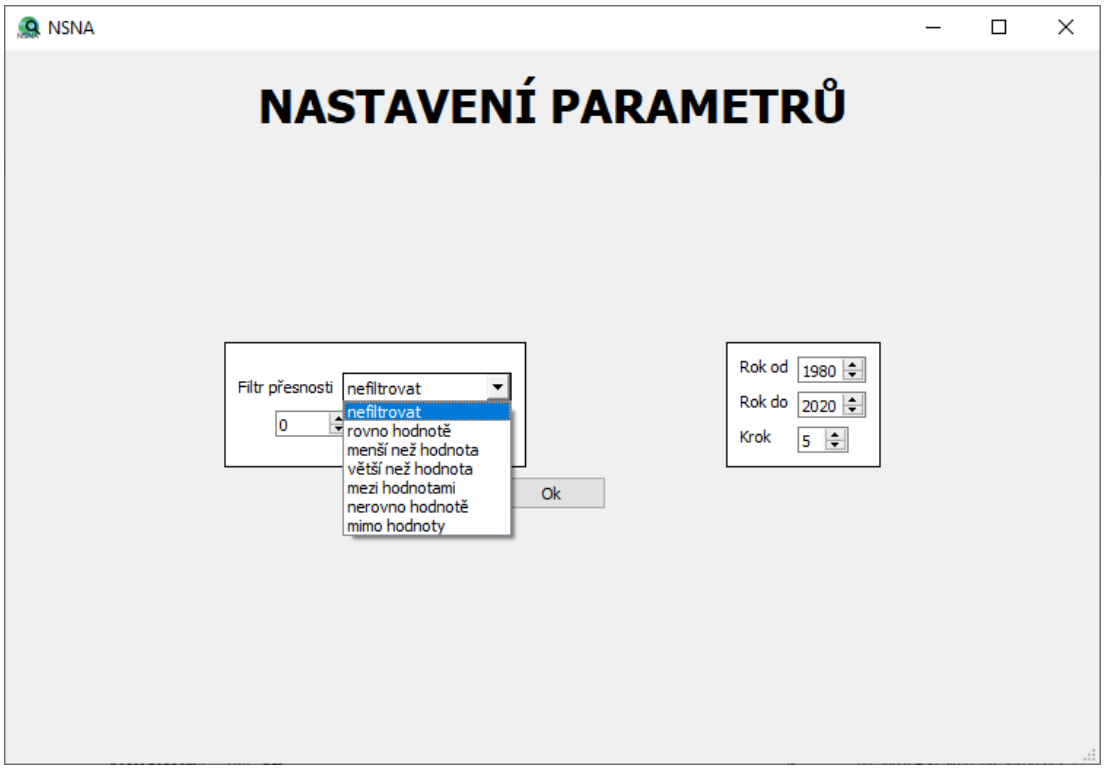
Obr. 5: Ikona aplikace NSNA

Po spuštění nástroje se otevře konzole a grafické rozhraní. Na první straně grafického rozhraní (Obr. 6) nástroje uživatel zadá latinský název druhu. Po potvrzení zadané hodnoty je spuštěna funkce pro získání nálezových dat.



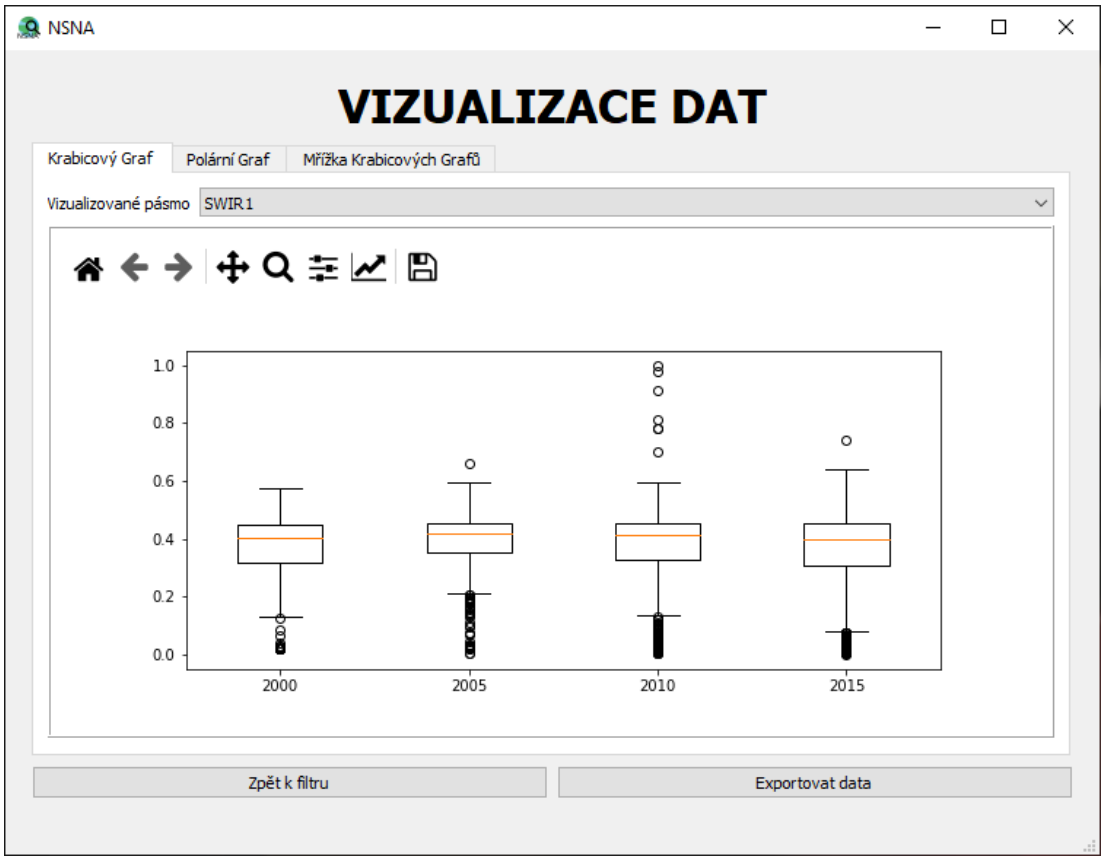
Obr. 6: První strana grafického rozhraní

Po získání dat proběhne posun na další stánku grafického rozhraní (Obr. 7), na které uživatel zadává parametry filtru a časového období. Po potvrzení hodnot je spušten výpočet spektrální charakteristiky.

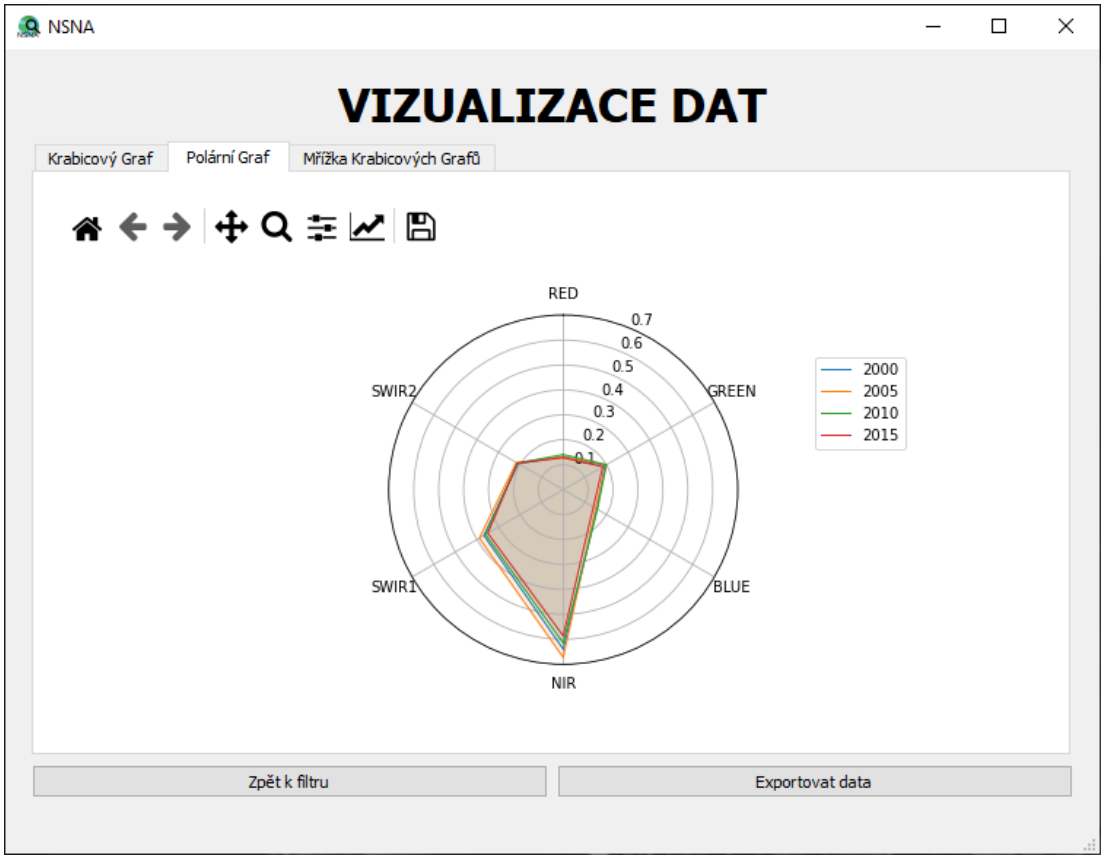


Obr. 7: Druhá strana grafického rozhraní

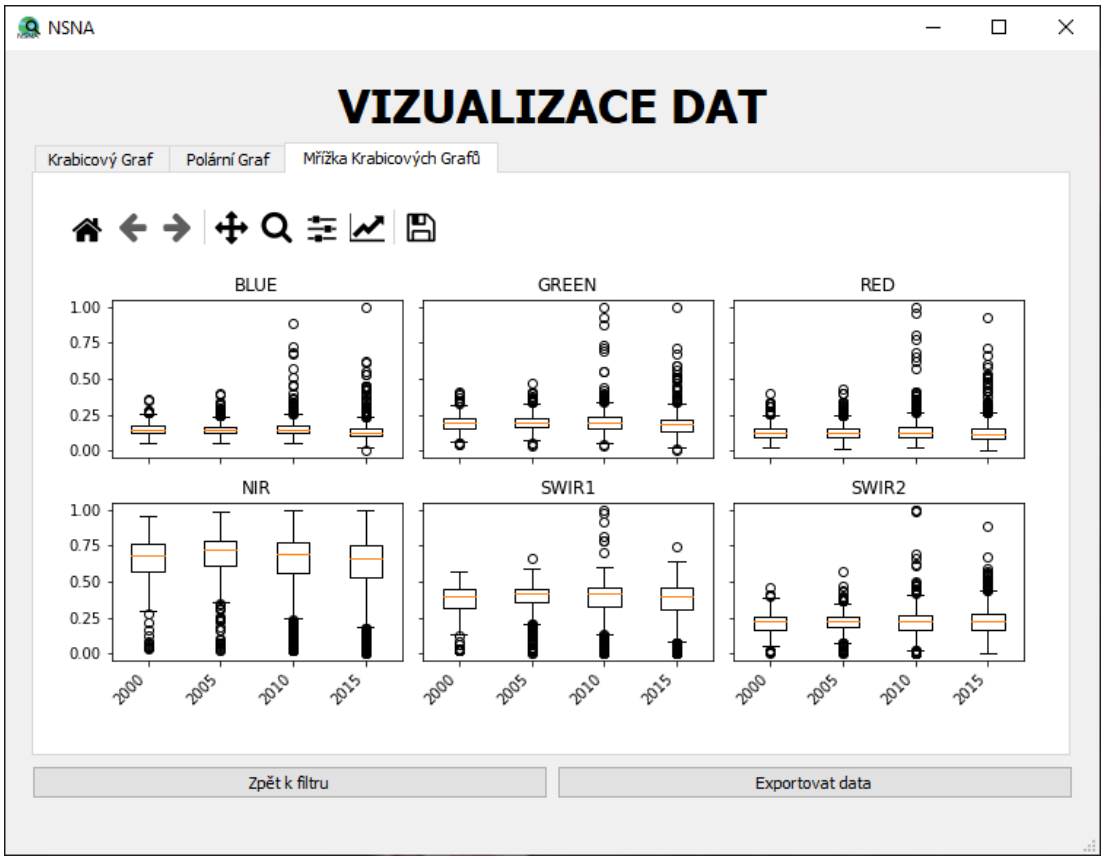
Po průběhu funkce je grafické rozhraní posunuto na poslední stranu (Obr. 8 - 10). Na poslední straně jsou vytvořeny tři vizualizace, mezi kterými lze přepínat pomocí záložek, a dvě tlačítka: (i) tlačítko pro návrat k filtru umožnuje aplikovat nový filtr bez potřeby opakovaného stahování nálezových dat a (ii) tlačítko pro export umožnuje uložení dat do excelové tabulky.



Obr. 8: Třetí strana grafického rozhraní, záložka se zobrazením krabicového grafu pro dané pásmo



Obr. 9: Třetí strana grafického rozhraní, záložka se zobrazením polárního grafu

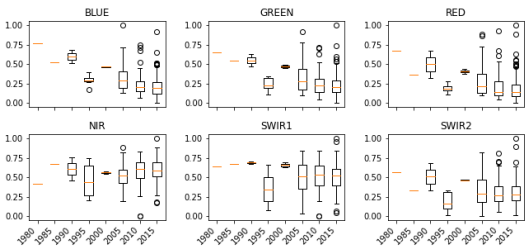


Obr. 10: Třetí strana grafického rozhraní, záložka se zobrazením krabicových grafů všech šesti využitých pásem

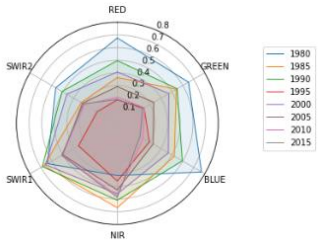
4.2 Ukázka výstupů pro konkrétním druhu

Jako ukázkový druh jsem zvolil plcha velkého (*Glis glis*). Pro ukázku srovnám grafické výstupy pro nálezy s hodnotou přesnoti rovnou 0 a nálezy s hodnotou přesnosti nerovnou 0 (hodnoty 1 až nekonečno).

Na krabicovém grafu spektrální charakteristiky nálezů s hodnotou přesnosti rovou 0 (Obr. 11) je zřejmá vyskoká kolísavost charakteristiky spektrální charakteristiky habitatu plcha. Polární graf nálezů s přesností rovnou 0 (Obr. 12) poukazuje na rozmanitost spektrální charakteristiky habitatu v jednotlivých obdobích.

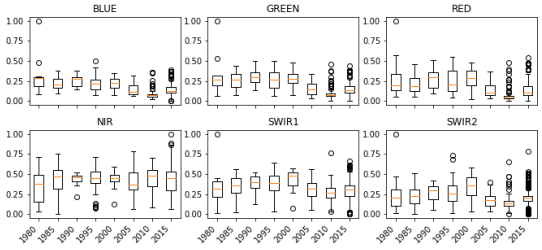


Obr. 11: Krabicové grafy spektrální charakteristiky habitatu plcha velkého s přesností nálezu rovnou 0

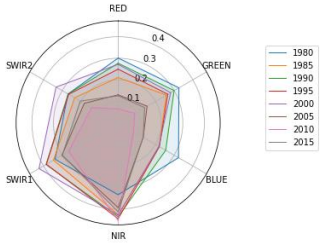


Obr. 12: Polární graf spektrální charakteristiky habitatu plcha velkého s přesností nálezu rovnou 0

Z krabicového grafu znázorňujícího nálezy s přesností nerovnou 0 (Obr. 13) lze vyčíst větší stabilitu v průměrech spektrálních hodnot. Za zmíňku stojí i množství odlehlých hodnot v posledních dvou obdobích. Otázkou může být čím je trend v posledních letech způsoben, rozmachem občanské vědy nebo například kvadrátovým mapováním v předchozích letech? Polární graf spektrální charakteristiky habitatu plcha velkého v jednotlivých obdobích (Obr. 14) je relativně jednotný.



Obr. 13: Krabicové grafy spektrální charakteristiky habitatu plcha velkého s přesností nálezu větší než 0



Obr. 14: Polární graf spektrální charakteristiky habitatu plcha velkého s přesností nálezu větší než 0

Z grafických výstupů aplikace interpretuji větší heterogenitu spektrální charakteristiky nálezů plcha velkého s přesností rovnou 0. Otázkou je zda výstup není ovlivněn nižším počtem nálezů plcha velkého s touto přesností (285). Především v několika počátečních obdobích je minimální počet nálezů.

5. Diskuse

Teoretickým základem jsou předpoklady o konzervatismu ekologické niky (Peterson 1999, Wiens a kol. 2010) a aplikovatelnosti spektrálních hodnot multispektrálních satelitních snímků pro popis druhového habitatu (Laurent a kol. 2005). Z těchto předpokladů vychází koncept kvantifikace sampling bias druhových nálezových dat: pokud dochází k změnám spektrální charakteristiky habitatu daného druhu lze předpokládat nedůvěryhodnost, stabilita spektrální charakteristiky naopak značí větší kvalitu dat. Na základě těchto předpokladů jsem vytvořil výzkumný nástroj pro analýzu nálezových dat z Nálezové databáze ochrany Přírody. Interpretace výsledků vyžaduje kritické hodnocení vycházející z ekologických charakteristik zvoleného druhu, například u druhu s velkou mobilitou lze předpokládat větší variabilitu.

Detekce sampling bias nálezových dat je aktuálním tématem hlavně v oblasti modelování druhových distribucí. Vzhledem k vlivu kvality dat na přesnost modelu (Støa a kol. 2018). Informace o kvalitě nálezových dat, ale není běžným atributem. Cosentino & Maiorano (2021) kvantifikovali sampling bias pomocí funkce závislosti množství nálezů na vzdálenosti různých faktorů dostupnosti. Bylo využito faktorů dostupnosti: silic, měst, vodních ploch a pobřeží. Byl prokázán vliv vzdálenosti silnic a měst na množství nálezů, faktory dostupnosti vodních ploch a pobřeží byly vyhodnoceny jako zanedbatelné.

Vytvořený nástroj poskytuje základ pro analýzu (ne)kvality nálezových dat z Nálezové databáze ochrany přírody na základě vizualizací a možnost exportu dat pro možnost následných analýz. Hlavní výhodou nástroje je schopnost automatického získání všech potřebných dat z webových zdrojů.

Do nástroje lze v budoucnosti implementovat novou funkcionalitu dle potřeb koncových uživatelů. Mezi případné rozšíření může patřit: (i) možnost využití vlastních nálezových dat a jejich nahrání z disku, (ii) rozšíření možností filtrování (více atributů, přesnější nastavení časových období nebo filtr nálezů z konkrétních měsíců), (iii) možnost nastavení získání spektrální informace (nastavení měsíců snímků, získání spektrální informace z okolí nálezu – pomocí bufferu nebo rozlišení využívaného pro extrakci hodnot pixelů, výpočet indexů, výpočet více hodnot – například před začátkem vegetační sezóny a během vegetační sezóny, nastavení tvorby mozaiek), (iv) implementace statistického vyhodnocení dat, (v) rozšíření možností

vizualizace dat (například seskupování podle kategorie přesnosti lokalizace) nebo (vi) možnost automatického dávkového zpracování na základě seznamu zvolených druhů a přednastavených parametrů. Nástroj bude nejspíš potřeba aktualizovat a upravit pro potřeby skutečného využití a potřeb koncových uživatelů. Pro lepší interperetaci grafických výstupů by jejich součástí mohla být informace o počtu záznamů v jednotlivých obdobích.

Analýza ukázkového druhu (plcha velkého) ukázala výstupy nástroje a možnosti interpretace výsledků. U zvoleného druhu je otázkou zda jsem zvolil optimální parametry a zda není výstup zkreslující. Vzhledem k těmto skutečnostem považuji za zásadní krok pro další verzi nástroje (která bude vyvýjena v rámci projektu nebo diplomové práce), vytvoření parametru pro nastavení minimálního počtu záznamů pro období. Tento parameter by mohl zabránit zkreslení výsledných výstupů.

6. Závěr a přínos práce

Práce v literární rešerši schrnuje teoretický základ pro kvantifikaci sampling bias nálezových dat pomocí spektrální charakteristiky druhového habitat na základě neklasifikovaných multispektrálních družicových snímků.

V rámci praktické části práce jsem vytvořil nástroj pro výzkum spekrální charakteristiky druhových habitatů definovaných lokalizacemi druhů z Nálezové databáze ochrany přírody. Primárním účelem aplikace je hodnocení kvality dat, konkrétně kvantifikace sampling bias, pro kterou je potřeba interpretace výsledků uživatele. Rozhodně nelze říct, že by aplikace řešila problém nálezových dat v praktickém a vědeckém využití, ale má potenciál jako nástroj pro výzkum charakteristik nálezových dat.

Na ukázce vybraného druhu (plcha velkého) byla demonstrována funkcionalita nástroje a byly vyhodnoceny zjevné a nezpochybnitelné nedostatky. Řešení těchto nedostatků je prvním budoucím krokem pro další vývoj nástroje. Další rozšíření funkcionality nástroje (zmiňované v diskusi) poskytují řadu námětů pro další řešení problematiky.

7. Seznam zdrojů

1. Allouche, O., Tsoar, A. & Kadmon, R. (2006). Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology, 43(6), 1223-1232. https://doi.org/10.1111/j.1365-2664.2006.01214.x.
2. Araújo, M. B. & Guisan, A. (2006). Five (or so) challenges for species distribution modelling. Journal of Biogeography, 33(10), 1677-1688. https://doi.org/10.1111/j.1365-2699.2006.01584.x.
3. Boenigk J., Wodniok S., Glücksman E., 2015: Biodiversity and earth history, Springer, London.
4. Conner, L. M. (2002). A Technique to Locate Isolated Populations Using Satellite Imagery. Wildlife Society Bulletin.
5. Cosentino, F. & Maiorano, L. (2021). Is geographic sampling bias representative of environmental space?. Ecological Informatics, 64. http://doi.org/10.1016/j.ecoinf.2021.101369.
6. Degbelo, A. & Kuhn, W. (2018). Spatial and temporal resolution of geographic information: an observation-based theory. Open geospatial data, Software and Standards, 3. https://doi.org/10.1186/s40965-018-0053-8.
7. Elith, J. (2013). Species Distribution Modeling. In: Levin, S. A (ed.): Encyclopedia of Biodiversity. Elsevier Science Publishing Co Inc. San Diego. 692–705.
8. Elton, C. (1927). Animal Ecology. Sedgwick and Jackson, London.
9. Estes, L. D., Okin, G. S., Mwangi, A. G. & Shugart, H. H. (2008). Habitat selection by a rare forest antelope: A multi-scale approach combining field data and imagery from three sensors. Remote Sensing of Environment, 112(5), 2033-2050. https://doi.org/10.1016/J.RSE.2008.01.004.
10. Fourcade, Y., Engler, J. O., Rödder, D. & Secondi, J. (2014). Mapping Species Distributions with MAXENT Using a Geographically Biased Sample of Presence Data: A Performance Assessment of Methods for Correcting Sampling Bias. PLoS ONE 9(5). https://doi.org/10.1371/journal.pone.0097122.
11. Gábor, L., Moudrý, V., Barták, V., & Lecours, V. (2019). How do species and data characteristics affect species distribution models and when to use

environmental filtering? International Journal of Geographical Information Science, 34(8), 1567-1584. https://doi.org/10.1080/13658816.2019.1615070.

12. Gillespie, T., Foody, G., Rocchini, D., Giorgi, A. & Saatchi, S. (2008). Measuring and modeling biodiversity from space. Progress in Physical Geography: Earth and Environment, 32(2), 203-221. https://doi.org/10.1177/0309133308093606.
13. Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D. & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone, Remote Sensing of Environment, 202, 18-27. https://doi.org/10.1016/j.rse.2017.06.031.
14. Guillera-Arroita, G., Lahoz-Monfort, J. J., Elith, J., Gordon, A., Kujala, H., Lentini, P. E., McCarthy, M.A., Tingley, R. & Wintle, B.A. (2015). Matching distribution models to applications. Global Ecology and Biogeography, 24(3), 276-292. https://doi.org/10.1111/geb.12268.
15. Guisan, A. & Zimmermann, N. (2000). Predictive habitat distribution models in ecology. Ecological Modelling, 135(2-3), 147-186. https://doi.org/10.1016/S0304-3800(00)00354-9.
16. Grinnell, J. (1917). The niche-relationships of the California Thrasher. The Auk, 34(4), 427– 433.
17. Hansen, M. C., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S. A., Tyukavina, A., Thau, D., Stheman, S. V., Goetz, S. J., Loveland, T. R., Kommareddy, A., Egorov, A., Chini, L., Justice, C. O. & Townshend, J. R. G. (2013). High-Resolution Global Maps of 21-st Century Forest Cover Change. Science, 342(6160), 850-853. https://doi.org/10.1126/science.1244693.
18. Hošek, M., Zárybnický, J., Škapec, L., Chobot, K. & Zohorna, J. (2008). Koncepce zpřístupnění nálezových dat ochrany přírody (online) [cit. 2022.03.24], dostupné z <https://www.casopis.ochranaprirody.cz/vyzkum-a-dokumentace/koncepce-zpristupneni-nalezovych-dat-ochrany-prirody/>.
19. Hutchinson G. (1992). Population studies: Animal ecology and demography, Bulletin od Mathematical Biology, 53, 193-213.
20. Chandler, M., See, L., Copas, K., Schmidt, A., Claramunt, B., Danielsen, F., Legrind, J., Masinde, S., Miller-Rushing, A., Greg, N. & Turak, E. (2016). Contribution of citizen science towards international biodiversity monitoring.

Biological Conservation, 213(B), 280-294. https://doi.org/10.1016/j.biocon.2016.09.004.

21. Chapman, A. D. (2005). Uses of Primary Species-Occurrence Data. Global Biodiversity Information Facility, Copenhagen.
22. Chauvier, Y., Zimmermann, N. E., Poggiato, G., Bystrova, D., Brun, P. & Thuiller, W. (2021). Novel methods to correct for observer and sampling bias in presence-only species distribution models. Global Ecology and Biogeography, 30(11), 2312-2325. https://doi.org/10.1111/geb.13383.
23. Chobot K., Kučera Z., Zárybnický J. & Hošek M. (20110. Nálezová databáze ochrany přírody (online) [cit. 2022.03.24], dostupné z <https://vesmir.cz/cz/casopis/archiv-casopisu/2011/cislo-7/nalezova-databaze-ochrany-prirody.html>.
24. Chobot, K., Kučera, Z., Duda, P. & Zárybnický, J. (2018). Nálezová databáze ochrany přírody otevřena veřejnosti (online) [cit. 2022.03.24], dostupné z <https://www.casopis.ochranaprirody.cz/z-nasi-prirody/nalezova-databaze-ochrany-prirody-otevrena-verejnosti/>
25. Chobot, K., Pergl, J. & Görner, T. (2021). Sledování nepůvodních a invazních druhů (online) [cit. 2022.03.24], dostupné z <https://www.casopis.ochranaprirody.cz/vyzkum-a-dokumentace/sledovani-nepuvodnich-a-invaznich-druhu/>
26. Kropáček J., Moravec D. & Komárek J. (2020). Dálkový průzkum Země I, ČZU v Praze, Praha.
27. Kovář, P. (2014). Ekosystémová a krajinná ekologie. Karolinum, Praha.
28. Laurent, E., Shi, H., Gatziolis, D., LeBouton, J., Walters, M. & Liu, J. (2005). Using the spatial and spectral precision of satellite imagery to predict wildlife occurrence patterns. Remote Sensing of Environment, 97(2), 249-262. https:/doi.org/10.1016/j.rse.2005.04.015.
29. Liu, C., White, M., Newell, G. & Griffioen, P. (2013). Species distribution modelling for
conservation planning in Victoria, Australia, Ecological Modelling, 249, 68-74. https://doi.org/10.1016/j.ecolmodel.2012.07.003.
30. Longley P., Goodchild M. F., Maguire D. J. & Rhind D. (2005). Geographic information science & systems. Wiley, Hoboken.

31. Loveland, T. R. & Dwyer, J. L. (2012). Landsat: Building a strong future. Remote Sensing of Environment, 122, 22-29. https://doi.org/10.1016/j.rse.2011.09.022.
32. Madani N., Kimball J., Nazeri M., Kumar L. & Affleck D. (2016). Remote Sensing Derived Fire Frequency, Soil Moisture and Ecosystem Productivity Explain Regional Movements in Emu over Australia. PloS one, 11(1). https://doi.org/10.1371/journal.pone.0147285.
33. Michener, W. K. & Jones, M. B. (2012). Ecoinformatics: supporting ecology as a data-intensive science. Trends in ecology & evolution, 27(2), 85-93. https://doi.org/10.1016/j.tree.2011.11.016.
34. Miller, J. (2010). Species Distribution Modeling. Geography Compass, 4(6), 490-509. https://doi.org/10.1111/j.1749-8198.2010.00351.x.
35. Moudrý, V. (2015). Modelling species distributions with simulated virtual species. Journal of Biogeography, 42(8), 1365-1366. https://doi.org/10.1111/jbi.12552.
36. Nagendra, H. (2001) Using remote sensing to assess biodiversity. International Journal of Remote Sensing, 22(12), 2377-2400. https://doi.org/10.1080/01431160117096.
37. Oeser, J., Heurich, M., Senf, C., Pflugmacher, D., Belotti, E. & Kuemmerle, T. (2020). Habitat metrics based on multi-temporal Landsat imagery for mapping large mammal habitat. Remote Sensing in Ecology and Conservation., 6(1), 52-69. https://doi.org/10.1002/rse2.122.
38. Pasher, J., King, D. & Lindsay, K. (2007). Modelling and mapping potential hooded warbler (Wilsonia citrina) habitat using remotely sensed imagery. Remote Sensing of Environment, 107(3), 471-483. https://doi.org/doi:10.1016/j.rse.2006.09.022.
39. Pavelka, K. (1999). Zpracování obrazových záznamů DPZ. České vysoké učení technické, Praha.
40. Pearman, P. B., Guisan, A., Broennimann, O. & Randin, Ch. F. (2008). Niche dynamics in space and time. Trends in Ecology & Evolution, 23(3), 149-158. https://doi.org/doi:10.1016/j.tree.2007.11.005.
41. Pearson, R. G. (2008). Species' Distribution Modeling for Conservation Educators and Practitioners (online) [cit. 2022.03.24], dostupné z

<http://ncep.amnh.org. https://academic.uprm.edu/~jchinea/UIP-MAPR/refs/modelos/pearson2008.pdf>.

42. Pekel, J., Cottam, A., Gorelick, N. & Belward, A. S. (2016). High-resolution mapping of global surface water and its long-term changes. Nature, 540, 418-422. https://doi.org/10.1038/nature20584.
43. Petersen, T. K., Speed, J. D. M., Grøtan, V. & Austrheim, G. (2021). Species data for understanding biodiversity dynamics: The what, where and when of species occurrence data collection. Ecological Solutions and Evidence, 2(1). https://doi.org/10.1002/2688-8319.12048.
44. Peterson, A. T. (1999). Conservatism of Ecological Niches in Evolutionary Time. Science, 285(5431), 1265-1267. https://doi.org/10.1126/science.285.5431.1265.
45. Peterson, A. T. & Holt, R. D. (2003). Niche differentiation in Mexican birds: using point occurrences to detect ecological innovation. Ecology Letters, 6(8), 774-782. https://doi.org/10.1046/j.1461-0248.2003.00502.x.
46. Peterson, A. (2004). Predicting the Geography of Species' Invasions via Ecological Niche Modeling. The Quarterly review of biology, 78(4), 419-433. https://doi.org/10.1086/378926.
47. Phillips, S. J., Dudík, M., Elith, J., Graham, C. H., Lehmann, A., Leathwick, J. & Ferrier, S. (2009). Sample selection bias and presence-only distribution models: implications for background and pseudo-absence data. Ecological Applications, 19(1), 181-197. https://doi.org/10.1890/07-2153.1.
48. Robertson, M. P., Cumming G. S. & Erasmus, B. F. N. (2010). Getting the most out of atlas data. Diversity and Distributions, 16(3), 363-375. https://doi.org/10.1111/j.1472-4642.2010.00639.x.
49. Rocchini, D., Salvatori, N., Beierkuhnlein, C., Chiarucci, A., De Boissieu, F., Förster, M., Garzón López, C. X., Gillespie, T., Hauffe, H., He, K., Kleinschmit, B., Lenoir, J., Malavasi, M., Moudrý, V., Nagendra, H., Payne, D., Šímová, P., Torresani, M., Wegmann, M. & Féret, J. (2021). From local spectral species to global spectral communities: A benchmark for ecosystem diversity estimate by remote sensing. Ecological Informatics, 61. https://doi.org/10.1016/j.ecoinf.2020.101195.
50. Rondinini, C., Wilson, K. A., Boitani, L., Grantham, H. & Possingham, H. P. (2006). Tradeoffs of different types of species occurrence data for use in

systematic conservation planning. Ecology Letters, 9, 1136-1145. https://doi.org/10.1111/j.1461-0248.2006.00970.x.

51. Rödder, D. & Lötters, S. (2009). Niche shift versus niche conservatism? Climatic characteristics of the native and invasive ranges of the Mediterranean house gecko (Hemidactylus turcicus). Global Ecology and Biogeography, 18(6), 674-687. https://doi.org/10.1111/j.1466-8238.2009.00477.x.
52. Sillero, N., Arenas-Castro, S., Enriquez-Urzelai, U., Vale, C., Guedes, D., Martínez-Freiría, F., Real, R. & Barbosa, A. M. (2021). Want to model a species niche? A step-by-step guideline on correlative ecological niche modelling. Ecological Modelling, 456. https://doi.org/10.1016/j.ecolmodel.2021.109671.
53. Soberón, J. & Peterson, A. (2005). Interpretation of Models of Fundamental Ecological Niches and Species' Distributional Areas. Biodiversity Informatics, 2. https://doi.org/10.17161/bi.v2i0.4.
54. Stickler, C. M. & Southworth, J. (2008). Application of multi-scale spatial and spectral analysis for predicting primate occurrence and habitat associations in Kibale National Park, Uganda, Remote Sensing of Environment, 112(5), 2170-2186. https://doi.org/10.1016/j.rse.2007.10.013.
55. Støa, B., Halvorsen, R., Mazzoni, S. & Gusarov,V. (2018). Sampling bias in presence-only data used for species distribution modelling: theory and methods for detecting sample bias and its effects on models. Sommerfeltia, 1, 1-53. https://doi.org/10.2478/som-2018-0001.
56. Storch, D. & Mihulka, S. (2007). Úvod do současné ekologie. Portál, Praha.
57. Townsend C. R., Begon M. & Harper J. L. (2010). Základy ekologie. Univerzita Palackého v Olomouci, Olomouc.
58. Thomas, C., Cameron, A., Green, R., Bakkenes, M., Beaumont, J. L., Collingham, C. Y., Erasmus, F. N. B., Ferreira de Siqueira, M., Grainger, A., Hannah, L., Hughes, L., Huntley, B., van Jaarsveld, S. A., Midgley F. G., Miles, L., Ortega-Huerta, A. M., Townsend Peterson, A., Phillips L. O. & Williams, E. S. (2004). Extinction risk from climate change. Nature, 427, 145-148. https://doi.org/10.1038/nature02121.
59. Vysoudil M. (1993). Dálkový průzkum Země 3. Univerzita Palackého, Olomouc.

60. Wiens, J. J., Ackerly, D. D., Allen, A. P., Anacker, B. L., Buckley, L. B., Cornell, H. V., Damschen, E. I., Jonathan Davies, T., Grytnes, J. A., Harrison, S. P., Hawkins, B. A., Holt, R. D., McCain, C. M. & Stephens, P. R. (2010). Niche conservatism as an emerging principle in ecology and conservation biology. Ecology Letters, 13, 1310-1324. https://doi.org/10.1111/j.1461-0248.2010.01515.x.
61. Whittaker, R. J., Araújo, M. B, Jepson, P., Ladle, R. J., Watson, J. E. M. & Willis, K. J. (2005). Conservation Biogeography: assessment and prospect. Diversity and Distributions, 11(1), 3-23. https://doi.org/10.1111/j.1366-9516.2005.00143.x

8. Přílohy

Příloha č. 1: main.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 18 11:40:14 2022

@author: michael

!python -m PyQt5.uic.pyuic -x gui.ui -o gui.py
!python -m PyQt5.uic.pyuic -x dialog.ui -o dialog.py
!pyinstaller main.py -F --distpath . --debug=all -n"NSNA" -i"iconEdit.ico"
"""

from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QFileDialog
from PyQt5.QtCore import QObject, QThread, pyqtSignal, Qt
from PyQt5.Qt import QDialog, QApplication
from PyQt5 import QtWidgets
from PyQt5 import QtGui
from gui import Ui_MainWindow
from dialog import Ui_Dialog
import sys
import ee, pandas as pd
import methods
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as NavigationToolbar
from matplotlib.figure import Figure

class WorkerOccurrences(QObject):
    finished = pyqtSignal()

    def __init__(self, taxonName):
        super().__init__()
        self.taxonName = taxonName

    def run(self):
        try:
            self.ex = False
            self.occurrenceData = methods.getOccurrences(self.taxonName)
        except Exception as e:
            self.ex = True
            self.e = e

        self.finished.emit()


class WorkerSpectral(QObject):
    finished = pyqtSignal()

    def __init__(self, yearParams, filterParams):
        super().__init__()
        self.yearParams = yearParams
        self.filterParams = filterParams

    def run(self):
        try:
            self.ex = False
```

```
            self.spectralDict = methods.getSpectralValues(self.yearParams, self.filterParams)
        except Exception as e:
            self.ex = True
            self.e = e

        self.finished.emit()


class Dialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.dialog = Ui_Dialog()
        self.dialog.setupUi(self)


class MainWindow(Ui_MainWindow):
    def init(self, MainWindow):
        self.MainWindow = MainWindow
        self.MainWindow.setWindowIcon(QtGui.QIcon('iconEdit.ico'))
        self.plotShowed = False
        self.plot2Showed = False
        self.plot3Showed = False
        self.bands = ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']

    def interactions(self):
        self.taxonPushButton.clicked.connect(self.getOccurrences)
        self.filterPushButton.clicked.connect(self.getSpectralValues)
        self.bandComboBox.currentIndexChanged.connect(self.visualizeData)
        self.returnFilterPushButton.clicked.connect(lambda: self.stackedWidget.setCurrentIndex(1))
        self.exportDataPushButton.clicked.connect(self.exportData)

    def showDialog(self, e):
        self.show = Dialog(self.MainWindow)
        self.show.dialog.errorLabel.setText(str(e))
        self.show.setWindowFlags(self.show.windowFlags() & ~Qt.WindowContextHelpButtonHint)
        self.show.dialog.okPushButton.clicked.connect(lambda: self.show.close())
        app.alert(self.MainWindow, 0)
        self.show.exec()

    def getOccurrences(self):
        self.taxonName = self.taxonLineEdit.text()
        if self.taxonName != '':
            self.thread = QThread()
            self.worker = WorkerOccurrences(self.taxonName)
            self.worker.moveToThread(self.thread)
            self.thread.started.connect(self.workerOccurrencesStart)
            self.thread.started.connect(self.worker.run)
            self.worker.finished.connect(self.thread.quit)
            self.worker.finished.connect(self.worker.deleteLater)
            self.thread.finished.connect(self.thread.deleteLater)
            self.thread.finished.connect(self.workerOccurrencesEnd)
            self.thread.start()
        else:
            self.showDialog('zadej latinský název druhu!')

    def workerOccurrencesStart(self):
        self.MainWindow.setCursor(Qt.BusyCursor)
        self.taxonPushButton.setEnabled(False)
        self.taxonLineEdit.setEnabled(False)
```

```
def workerOccurrencesEnd(self):
    self.MainWindow.unsetCursor()
    self.taxonPushButton.setEnabled(True)
    self.taxonLineEdit.setEnabled(True)
    self.occurrenceData = self.worker.occurrenceData

    if self.worker.ex:
        self.showDialog(str(self.worker.e))
    else:
        app.alert(self.MainWindow, 0)
        self.stackedWidget.setCurrentIndex(1)

def getSpectralValues(self):
    filterParams = [self.occurrenceData,
                    self.filterSpinBox.value(),
                    self.filterSpinBox2.value(),
                    self.filterComboBox.currentIndex()
                    ]

    yearParams = [self.startYearSpinBox.value(),
                  self.endYearSpinBox.value(),
                  self.stepSpinBox.value()
                  ]

    if yearParams[0] > yearParams [1] or yearParams[2] > yearParams[1] - yearParams[0]:
        self.showDialog('chybně zadané hodnoty časového období!')
    else:
        if filterParams[3] in [4,6]:
            if filterParams[1] > filterParams[2]:
                self.showDialog('druhá hodnota filtru přesnosti musí být větší než první!')
            else:
                self.thread = QThread()
                self.worker = WorkerSpectral(yearParams, filterParams)
                self.worker.moveToThread(self.thread)
                self.thread.started.connect(self.workerSpectralStart)
                self.thread.started.connect(self.worker.run)
                self.worker.finished.connect(self.thread.quit)
                self.worker.finished.connect(self.worker.deleteLater)
                self.thread.finished.connect(self.thread.deleteLater)
                self.thread.finished.connect(self.workerSpectralEnd)
                self.thread.start()

        else:
            self.thread = QThread()
            self.worker = WorkerSpectral(yearParams, filterParams)
            self.worker.moveToThread(self.thread)
            self.thread.started.connect(self.workerSpectralStart)
            self.thread.started.connect(self.worker.run)
            self.worker.finished.connect(self.thread.quit)
            self.worker.finished.connect(self.worker.deleteLater)
            self.thread.finished.connect(self.thread.deleteLater)
            self.thread.finished.connect(self.workerSpectralEnd)
            self.thread.start()


def workerSpectralStart(self):
    self.MainWindow.setCursor(Qt.BusyCursor)
    self.filterPushButton.setEnabled(False)
    self.startYearSpinBox.setEnabled(False)
    self.endYearSpinBox.setEnabled(False)
```

```
        self.stepSpinBox.setEnabled(False)
        self.filterComboBox.setEnabled(False)
        self.filterSpinBox.setEnabled(False)
        self.filterSpinBox2.setEnabled(False)

    def workerSpectralEnd(self):
        self.MainWindow.unsetCursor()
        self.spectralDict = self.worker.spectralDict
        self.filterPushButton.setEnabled(True)
        self.startYearSpinBox.setEnabled(True)
        self.endYearSpinBox.setEnabled(True)
        self.stepSpinBox.setEnabled(True)
        self.filterComboBox.setEnabled(True)
        self.filterSpinBox.setEnabled(True)
        self.filterSpinBox2.setEnabled(True)

        if self.worker.ex:
            self.showDialog(str(self.worker.e))
        else:
            app.alert(self.MainWindow, 0)
            self.visualizeData()
            self.visualizeData2()
            self.visualizeData3()
            self.stackedWidget.setCurrentIndex(2)


    def exportData(self):
        fileName = QFileDialog.getSaveFileName(caption='Save File', filter='*.xlsx')[0]
        if fileName != '':
            with pd.ExcelWriter(fileName, engine='xlsxwriter') as writer:

                for i in self.spectralDict:
                    pd.DataFrame(self.spectralDict[i]).to_excel(writer,
                                                        sheet_name=str(i)
                                                        )

    def visualizeData(self):
        dictonary = dict()
        for i in self.spectralDict:
            dictonary[i] = {band: self.spectralDict[i][band] for band in self.bands}
        band = self.bandComboBox.currentText()
        if len(dictonary) != 0:
            if not self.plotShowed:
                self.figure = plt.figure()
                self.canvas = FigureCanvas(self.figure)
                self.toolbar = NavigationToolbar(self.canvas, self.boxPlotFrame)
                self.layout = QtWidgets.QVBoxLayout()
                self.layout.addWidget(self.toolbar)
                self.layout.addWidget(self.canvas)
                self.boxPlotFrame.setLayout(self.layout)
                self.ax = self.figure.add_subplot(111)

            self.ax.clear()
            self.ax.boxplot([dictonary[i][band] for i in dictonary],
                            labels=[i for i in dictonary])

            self.canvas.draw()
            self.plotShowed = True
        else:
            self.showDialog('nodata')
```

```
def visualizeData2(self):
    from math import pi
    dictonary = dict()
    for i in self.spectralDict:
        dictonary[i] = {band: self.spectralDict[i][band] for band in self.bands}

    radarDict = {'group': [],
                 'RED': [],
                 'GREEN': [],
                 'BLUE': [],
                 'NIR': [],
                 'SWIR1': [],
                 'SWIR2': []}

    if not self.plot2Showed:
        self.figure2 = plt.figure()
        self.canvas2 = FigureCanvas(self.figure2)
        self.toolbar2 = NavigationToolbar(self.canvas2, self.radarPlotFrame)
        self.layout2 = QtWidgets.QVBoxLayout()
        self.layout2.addWidget(self.toolbar2)
        self.layout2.addWidget(self.canvas2)
        self.radarPlotFrame.setLayout(self.layout2)
        self.ax2 = self.figure2.add_subplot(111, polar=True)

    self.ax2.clear()

    for year in dictonary:
        radarDict['group'].append(year)
        for band in dictonary[year]:
            valueList = dictonary[year][band]
            valueListMean = sum(valueList)/len(valueList)
            radarDict[band].append(valueListMean)

    df = pd.DataFrame(radarDict)

    categories=list(df)[1:]
    N = len(categories)

    angles = [n / float(N) * 2 * pi for n in range(N)]
    angles += angles[:1]

    self.ax2.set_theta_offset(pi / 2)
    self.ax2.set_theta_direction(-1)

    self.ax2.set_xticks(angles[:-1], categories)

    for i in range(len(df.group.to_list())):
        values=df.loc[i].drop('group').values.flatten().tolist()
        values += values[:1]
        self.ax2.plot(angles, values, linewidth=1, linestyle='solid', label=f"{df.group.to_list()[i]}")
        self.ax2.fill(angles, values, alpha=0.1)

    self.ax2.legend(bbox_to_anchor=(1.5, 0.9))

    self.canvas2.draw()
    self.plot2Showed = True

def visualizeData3(self):
     dictonary = dict()
```

```
        for i in self.spectralDict:
            dictonary[i] = {band: self.spectralDict[i][band] for band in self.bands}
        if len(dictonary) != 0:
            if not self.plot3Showed:
                self.figure3, self.axs = plt.subplots(2, 3, sharex=True, sharey=True)
                self.canvas3 = FigureCanvas(self.figure3)
                self.toolbar3 = NavigationToolbar(self.canvas3, self.boxPlotFrame2)
                self.layout3 = QtWidgets.QVBoxLayout()
                self.layout3.addWidget(self.toolbar3)
                self.layout3.addWidget(self.canvas3)
                self.boxPlotFrame2.setLayout(self.layout3)

            c = 0
            x = 0
            y = 0
            for b in self.bands:
                self.axs[x, y].clear()
                self.axs[x, y].boxplot([dictonary[i][b] for i in dictonary],
                                       labels=[i for i in dictonary])
                self.axs[x, y].set_title(b)

                if c < 2:
                    c+=1
                    y+=1
                else:
                    y=0
                    x=1
                    c=0

            self.figure3.autofmt_xdate(rotation=45)
            self.figure3.tight_layout()
            self.canvas3.draw()
            self.plot3Showed = True
        else:
            self.showDialog('nodata')


class AppWindow(QMainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.ui = MainWindow()
        self.ui.setupUi(self)
        self.ui.init(self)
        self.ui.interactions()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.processEvents()
    w = AppWindow()
    w.show()
    sys.exit(app.exec_())
```

příloha č. 2: gui.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'gui.ui'
#
# Created by: PyQt5 UI code generator 5.12.3
#
```

```
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(750, 489)
        MainWindow.setStyleSheet("#frame, #frame_2{\n"
"    border: 1px solid #000000;\n"
"    background: #ffffff;\n"
"}\n"
"\n"
"QPushButton:hover {\n"
"    border: 2px solid #000000;\n"
"}\n"
"\n"
"QLineEdit:focus, QSpinBox:focus, QComboBox:hover {\n"
"    border: 1px solid #000000\n"
"}")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget)
        self.stackedWidget.setObjectName("stackedWidget")
        self.page = QtWidgets.QWidget()
        self.page.setObjectName("page")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.page)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.verticalLayout_8 = QtWidgets.QVBoxLayout()
        self.verticalLayout_8.setObjectName("verticalLayout_8")
        self.label = QtWidgets.QLabel(self.page)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
        self.label.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setPointSize(24)
        font.setBold(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.verticalLayout_8.addWidget(self.label, 0, QtCore.Qt.AlignHCenter)
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_8.addItem(spacerItem)
        self.verticalLayout_6 = QtWidgets.QVBoxLayout()
        self.verticalLayout_6.setObjectName("verticalLayout_6")
        self.taxonLineEdit = QtWidgets.QLineEdit(self.page)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.taxonLineEdit.sizePolicy().hasHeightForWidth())
        self.taxonLineEdit.setSizePolicy(sizePolicy)
```

```
        self.taxonLineEdit.setMinimumSize(QtCore.QSize(200, 0))
        self.taxonLineEdit.setCursor(QtGui.QCursor(QtCore.Qt.IBeamCursor))
        self.taxonLineEdit.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.taxonLineEdit.setToolTip("")
        self.taxonLineEdit.setText("")
        self.taxonLineEdit.setAlignment(QtCore.Qt.AlignCenter)
        self.taxonLineEdit.setClearButtonEnabled(False)
        self.taxonLineEdit.setObjectName("taxonLineEdit")
        self.verticalLayout_6.addWidget(self.taxonLineEdit, 0,
QtCore.Qt.AlignHCenter|QtCore.Qt.AlignVCenter)
        self.taxonPushButton = QtWidgets.QPushButton(self.page)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.taxonPushButton.sizePolicy().hasHeightForWidth())
        self.taxonPushButton.setSizePolicy(sizePolicy)
        self.taxonPushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.taxonPushButton.setFocusPolicy(QtCore.Qt.NoFocus)
        self.taxonPushButton.setToolTip("")
        self.taxonPushButton.setStatusTip("")
        self.taxonPushButton.setWhatsThis("")
        self.taxonPushButton.setObjectName("taxonPushButton")
        self.verticalLayout_6.addWidget(self.taxonPushButton, 0,
QtCore.Qt.AlignHCenter|QtCore.Qt.AlignVCenter)
        self.verticalLayout_8.addLayout(self.verticalLayout_6)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_8.addItem(spacerItem1)
        self.gridLayout_2.addLayout(self.verticalLayout_8, 0, 0, 1, 1)
        self.stackedWidget.addWidget(self.page)
        self.page_2 = QtWidgets.QWidget()
        self.page_2.setObjectName("page_2")
        self.gridLayout_9 = QtWidgets.QGridLayout(self.page_2)
        self.gridLayout_9.setObjectName("gridLayout_9")
        self.verticalLayout_7 = QtWidgets.QVBoxLayout()
        self.verticalLayout_7.setObjectName("verticalLayout_7")
        self.label_2 = QtWidgets.QLabel(self.page_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth())
        self.label_2.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setPointSize(24)
        font.setBold(True)
        font.setWeight(75)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.verticalLayout_7.addWidget(self.label_2, 0, QtCore.Qt.AlignHCenter)
        spacerItem2 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_7.addItem(spacerItem2)
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem3)
        self.frame = QtWidgets.QFrame(self.page_2)
```

```
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.gridLayout_3 = QtWidgets.QGridLayout(self.frame)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_5.setObjectName("horizontalLayout_5")
        self.label_7 = QtWidgets.QLabel(self.frame)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_7.sizePolicy().hasHeightForWidth())
        self.label_7.setSizePolicy(sizePolicy)
        self.label_7.setObjectName("label_7")
        self.horizontalLayout_5.addWidget(self.label_7)
        self.filterComboBox = QtWidgets.QComboBox(self.frame)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.filterComboBox.sizePolicy().hasHeightForWidth())
        self.filterComboBox.setSizePolicy(sizePolicy)
        self.filterComboBox.setFocusPolicy(QtCore.Qt.NoFocus)
        self.filterComboBox.setObjectName("filterComboBox")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.filterComboBox.addItem("")
        self.horizontalLayout_5.addWidget(self.filterComboBox)
        self.verticalLayout_2.addLayout(self.horizontalLayout_5)
        self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_6.setObjectName("horizontalLayout_6")
        self.filterSpinBox = QtWidgets.QSpinBox(self.frame)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.filterSpinBox.sizePolicy().hasHeightForWidth())
        self.filterSpinBox.setSizePolicy(sizePolicy)
        self.filterSpinBox.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.filterSpinBox.setMaximum(99999)
        self.filterSpinBox.setObjectName("filterSpinBox")
        self.horizontalLayout_6.addWidget(self.filterSpinBox)
        self.filterSpinBox2 = QtWidgets.QSpinBox(self.frame)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.filterSpinBox2.sizePolicy().hasHeightForWidth())
        self.filterSpinBox2.setSizePolicy(sizePolicy)
        self.filterSpinBox2.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.filterSpinBox2.setMaximum(99999)
        self.filterSpinBox2.setObjectName("filterSpinBox2")
        self.horizontalLayout_6.addWidget(self.filterSpinBox2)
```

```
        self.verticalLayout_2.addLayout(self.horizontalLayout_6)
        self.gridLayout_3.addLayout(self.verticalLayout_2, 0, 0, 1, 1)
        self.horizontalLayout_3.addWidget(self.frame)
        spacerItem4 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem4)
        self.frame_2 = QtWidgets.QFrame(self.page_2)
        self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_2.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame_2.setObjectName("frame_2")
        self.gridLayout_8 = QtWidgets.QGridLayout(self.frame_2)
        self.gridLayout_8.setObjectName("gridLayout_8")
        self.formLayout_2 = QtWidgets.QFormLayout()
        self.formLayout_2.setObjectName("formLayout_2")
        self.label_4 = QtWidgets.QLabel(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_4.sizePolicy().hasHeightForWidth())
        self.label_4.setSizePolicy(sizePolicy)
        self.label_4.setObjectName("label_4")
        self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.label_4)
        self.label_5 = QtWidgets.QLabel(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_5.sizePolicy().hasHeightForWidth())
        self.label_5.setSizePolicy(sizePolicy)
        self.label_5.setObjectName("label_5")
        self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.LabelRole, self.label_5)
        self.label_6 = QtWidgets.QLabel(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_6.sizePolicy().hasHeightForWidth())
        self.label_6.setSizePolicy(sizePolicy)
        self.label_6.setObjectName("label_6")
        self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.LabelRole, self.label_6)
        self.endYearSpinBox = QtWidgets.QSpinBox(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.endYearSpinBox.sizePolicy().hasHeightForWidth())
        self.endYearSpinBox.setSizePolicy(sizePolicy)
        self.endYearSpinBox.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.endYearSpinBox.setMinimum(1982)
        self.endYearSpinBox.setMaximum(2030)
        self.endYearSpinBox.setProperty("value", 2020)
        self.endYearSpinBox.setObjectName("endYearSpinBox")
        self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.FieldRole, self.endYearSpinBox)
        self.startYearSpinBox = QtWidgets.QSpinBox(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.startYearSpinBox.sizePolicy().hasHeightForWidth())
```

```
        self.startYearSpinBox.setSizePolicy(sizePolicy)
        self.startYearSpinBox.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.startYearSpinBox.setMinimum(1980)
        self.startYearSpinBox.setMaximum(2030)
        self.startYearSpinBox.setObjectName("startYearSpinBox")
        self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.startYearSpinBox)
        self.stepSpinBox = QtWidgets.QSpinBox(self.frame_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.stepSpinBox.sizePolicy().hasHeightForWidth())
        self.stepSpinBox.setSizePolicy(sizePolicy)
        self.stepSpinBox.setFocusPolicy(QtCore.Qt.ClickFocus)
        self.stepSpinBox.setMinimum(1)
        self.stepSpinBox.setMaximum(20)
        self.stepSpinBox.setProperty("value", 5)
        self.stepSpinBox.setObjectName("stepSpinBox")
        self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.FieldRole, self.stepSpinBox)
        self.gridLayout_8.addLayout(self.formLayout_2, 1, 0, 1, 1)
        self.horizontalLayout_3.addWidget(self.frame_2)
        spacerItem5 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem5)
        self.verticalLayout_7.addLayout(self.horizontalLayout_3)
        self.verticalLayout_5 = QtWidgets.QVBoxLayout()
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.filterPushButton = QtWidgets.QPushButton(self.page_2)
        self.filterPushButton.setEnabled(True)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.filterPushButton.sizePolicy().hasHeightForWidth())
        self.filterPushButton.setSizePolicy(sizePolicy)
        self.filterPushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.filterPushButton.setFocusPolicy(QtCore.Qt.NoFocus)
        self.filterPushButton.setObjectName("filterPushButton")
        self.verticalLayout_5.addWidget(self.filterPushButton, 0, QtCore.Qt.AlignHCenter)
        self.verticalLayout_7.addLayout(self.verticalLayout_5)
        spacerItem6 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_7.addItem(spacerItem6)
        self.gridLayout_9.addLayout(self.verticalLayout_7, 0, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_2)
        self.page_3 = QtWidgets.QWidget()
        self.page_3.setObjectName("page_3")
        self.gridLayout_5 = QtWidgets.QGridLayout(self.page_3)
        self.gridLayout_5.setObjectName("gridLayout_5")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_3 = QtWidgets.QLabel(self.page_3)
        font = QtGui.QFont()
        font.setPointSize(24)
        font.setBold(True)
        font.setWeight(75)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.verticalLayout.addWidget(self.label_3, 0, QtCore.Qt.AlignHCenter)
        self.gridLayout_5.addLayout(self.verticalLayout, 0, 0, 1, 1)
```

```
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.returnFilterPushButton = QtWidgets.QPushButton(self.page_3)
        self.returnFilterPushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.returnFilterPushButton.setFocusPolicy(QtCore.Qt.NoFocus)
        self.returnFilterPushButton.setObjectName("returnFilterPushButton")
        self.horizontalLayout.addWidget(self.returnFilterPushButton)
        self.exportDataPushButton = QtWidgets.QPushButton(self.page_3)
        self.exportDataPushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.exportDataPushButton.setFocusPolicy(QtCore.Qt.NoFocus)
        self.exportDataPushButton.setObjectName("exportDataPushButton")
        self.horizontalLayout.addWidget(self.exportDataPushButton)
        self.gridLayout_5.addLayout(self.horizontalLayout, 2, 0, 1, 1)
        self.tabWidget_2 = QtWidgets.QTabWidget(self.page_3)
        self.tabWidget_2.setFocusPolicy(QtCore.Qt.NoFocus)
        self.tabWidget_2.setObjectName("tabWidget_2")
        self.tab_3 = QtWidgets.QWidget()
        self.tab_3.setObjectName("tab_3")
        self.gridLayout_4 = QtWidgets.QGridLayout(self.tab_3)
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout()
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
        self.label_8 = QtWidgets.QLabel(self.tab_3)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_8.sizePolicy().hasHeightForWidth())
        self.label_8.setSizePolicy(sizePolicy)
        self.label_8.setObjectName("label_8")
        self.horizontalLayout_4.addWidget(self.label_8)
        self.bandComboBox = QtWidgets.QComboBox(self.tab_3)
        self.bandComboBox.setFocusPolicy(QtCore.Qt.NoFocus)
        self.bandComboBox.setObjectName("bandComboBox")
        self.bandComboBox.addItem("")
        self.bandComboBox.addItem("")
        self.bandComboBox.addItem("")
        self.bandComboBox.addItem("")
        self.bandComboBox.addItem("")
        self.bandComboBox.addItem("")
        self.horizontalLayout_4.addWidget(self.bandComboBox)
        self.verticalLayout_3.addLayout(self.horizontalLayout_4)
        self.boxPlotFrame = QtWidgets.QFrame(self.tab_3)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.MinimumExpanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.boxPlotFrame.sizePolicy().hasHeightForWidth())
        self.boxPlotFrame.setSizePolicy(sizePolicy)
        self.boxPlotFrame.setFrameShape(QtWidgets.QFrame.Box)
        self.boxPlotFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.boxPlotFrame.setObjectName("boxPlotFrame")
        self.verticalLayout_3.addWidget(self.boxPlotFrame)
        self.gridLayout_4.addLayout(self.verticalLayout_3, 0, 0, 1, 1)
        self.tabWidget_2.addTab(self.tab_3, "")
        self.tab_4 = QtWidgets.QWidget()
        self.tab_4.setObjectName("tab_4")
        self.gridLayout_6 = QtWidgets.QGridLayout(self.tab_4)
```

```
        self.gridLayout_6.setObjectName("gridLayout_6")
        self.radarPlotFrame = QtWidgets.QFrame(self.tab_4)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.MinimumExpanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.radarPlotFrame.sizePolicy().hasHeightForWidth())
        self.radarPlotFrame.setSizePolicy(sizePolicy)
        self.radarPlotFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.radarPlotFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.radarPlotFrame.setObjectName("radarPlotFrame")
        self.gridLayout_6.addWidget(self.radarPlotFrame, 0, 0, 1, 1)
        self.tabWidget_2.addTab(self.tab_4, "")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.gridLayout_7 = QtWidgets.QGridLayout(self.tab)
        self.gridLayout_7.setObjectName("gridLayout_7")
        self.boxPlotFrame2 = QtWidgets.QFrame(self.tab)
        self.boxPlotFrame2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.boxPlotFrame2.setFrameShadow(QtWidgets.QFrame.Raised)
        self.boxPlotFrame2.setObjectName("boxPlotFrame2")
        self.gridLayout_7.addWidget(self.boxPlotFrame2, 0, 0, 1, 1)
        self.tabWidget_2.addTab(self.tab, "")
        self.gridLayout_5.addWidget(self.tabWidget_2, 1, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_3)
        self.gridLayout.addWidget(self.stackedWidget, 4, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.stackedWidget.setCurrentIndex(0)
        self.tabWidget_2.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "NSNA"))
        self.label.setText(_translate("MainWindow", "VÝBĚR ZÁJMOVÉHO DRUHU"))
        self.taxonLineEdit.setPlaceholderText(_translate("MainWindow", "zadej latinský název druhu"))
        self.taxonPushButton.setText(_translate("MainWindow", "Ok"))
        self.label_2.setText(_translate("MainWindow", "NASTAVENÍ PARAMETRŮ"))
        self.label_7.setText(_translate("MainWindow", "Filtr přesnosti"))
        self.filterComboBox.setItemText(0, _translate("MainWindow", "nefiltrovat"))
        self.filterComboBox.setItemText(1, _translate("MainWindow", "rovno hodnotě"))
        self.filterComboBox.setItemText(2, _translate("MainWindow", "menší než hodnota"))
        self.filterComboBox.setItemText(3, _translate("MainWindow", "větší než hodnota"))
        self.filterComboBox.setItemText(4, _translate("MainWindow", "mezi hodnotami"))
        self.filterComboBox.setItemText(5, _translate("MainWindow", "nerovno hodnotě"))
        self.filterComboBox.setItemText(6, _translate("MainWindow", "mimo hodnoty"))
        self.label_4.setText(_translate("MainWindow", "Rok od"))
        self.label_5.setText(_translate("MainWindow", "Rok do"))
        self.label_6.setText(_translate("MainWindow", "Krok"))
        self.filterPushButton.setText(_translate("MainWindow", "Ok"))
        self.label_3.setText(_translate("MainWindow", "VIZUALIZACE DAT"))
        self.returnFilterPushButton.setText(_translate("MainWindow", "Zpět k filtru"))
        self.exportDataPushButton.setText(_translate("MainWindow", "Exportovat data"))
        self.label_8.setText(_translate("MainWindow", "Vizualizované pásmo"))
        self.bandComboBox.setItemText(0, _translate("MainWindow", "BLUE"))
```

```
        self.bandComboBox.setItemText(1, _translate("MainWindow", "GREEN"))
        self.bandComboBox.setItemText(2, _translate("MainWindow", "RED"))
        self.bandComboBox.setItemText(3, _translate("MainWindow", "NIR"))
        self.bandComboBox.setItemText(4, _translate("MainWindow", "SWIR1"))
        self.bandComboBox.setItemText(5, _translate("MainWindow", "SWIR2"))
        self.tabWidget_2.setTabText(self.tabWidget_2.indexOf(self.tab_3), _translate("MainWindow",
"Krabicový Graf"))
        self.tabWidget_2.setTabText(self.tabWidget_2.indexOf(self.tab_4), _translate("MainWindow",
"Polární Graf"))
        self.tabWidget_2.setTabText(self.tabWidget_2.indexOf(self.tab), _translate("MainWindow",
"Mřížka Krabicových Grafů"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

příloha č. 3: methods.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 18 16:07:51 2022

@author: michael
"""

import ee, geopandas as gpd, pandas as pd
import ndop as nd
from io import StringIO
import csv

def getOccurrences(taxonName):
    def getNdopData(s,num_rec,table_payload):
        csv_table = []
        counter = 0

        for i in range(0, num_rec, 500):
            counter+=1
            table_url = (
                'https://portal.nature.cz/nd/find.php?'
                'akce=seznam&opener=&vztazne_id=0&order=ID_ND_NALEZ'
                '&orderhow=DESC&frompage={frompage}&pagesize=500&'
                'filtering=&searching=&export=1&ndtoken={ndtoken}'
            ).format(frompage=str(i), ndtoken=table_payload['ndtokenexport'])

            req = s.post(url=table_url, data=table_payload)
            req.encoding = 'cp1250'
            f = StringIO(req.text)
            reader = csv.reader(f, delimiter=';')
            if counter == 1:
                csv_table.append(list(reader))
            else:
                csv_table.append(list(reader)[1:])

        ndopData = sum(csv_table,[])
        return ndopData
```

```
    search_payload = nd.get_search_pars()
    search_payload.update({'rfTaxon':f'{taxonName}'})

    username = "michael.kuceraaa@gmail.com"
    password = "Ei1qM1aPi"


    s = nd.login(username, password)
    table_payload, num_rec = nd.search_filter(s,search_payload)

    ndopData = getNdopData(s,
                num_rec,
                table_payload
                )

    ndopDf = pd.DataFrame(data=ndopData[1:], columns=ndopData[0])

    souradniceX = ndopDf.X.to_list()
    souradniceY = ndopDf.Y.to_list()
    presnost = ndopDf.CXPRESNOST.to_list()
    datum = ndopDf.DATUM_OD.to_list()

    dfDict = {'X':[],
              'Y':[],
              'presnost':[],
              'datum':[]
              }

    for i in range(len(souradniceY)):
        try:
            dfDict['X'].append(float(souradniceX[i].replace(',', '.')))
            dfDict['Y'].append(float(souradniceY[i].replace(',', '.')))
            dfDict['presnost'].append(int(presnost[i]))
            dfDict['datum'].append(int(datum[i]))
        except:
            None

    df = pd.DataFrame(columns=['X', 'Y', 'CXPRESNOST', 'DATUM_OD'])
    df.X = dfDict['X']
    df.Y = dfDict['Y']
    df.CXPRESNOST = dfDict['presnost']
    df.DATUM_OD = dfDict['datum']

    gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.X,
df.Y)).set_crs('epsg:5514').to_crs('epsg:4326')

    return gdf

def getSpectralValues(yearParams, filterParams):
    spectralDict = dict()
    try:
        ee.Initialize()
    except:
        ee.Authenticate()
        ee.Initialize()

    def maskClouds(image):
        qa = image.select('pixel_qa')
```

```
        maskedImage = image.updateMask(qa.bitwiseAnd(1 << 3).eq(0)).updateMask(qa.bitwiseAnd(1
<< 5).eq(0))
        return maskedImage

    def getLandsat():
        l8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR') \
              .filter(ee.Filter.calendarRange(5, 8, 'month')) \
              .map(maskClouds) \
              .select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7'],
                    ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']
                    )

        l7 = ee.ImageCollection('LANDSAT/LE07/C01/T1_SR') \
              .filter(ee.Filter.calendarRange(5, 8, 'month')) \
              .map(maskClouds) \
              .select(['B1', 'B2', 'B3', 'B4', 'B5', 'B7'],
                    ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']
                    )

        l5 = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR') \
              .filter(ee.Filter.calendarRange(5, 8, 'month')) \
              .map(maskClouds) \
              .select(['B1', 'B2', 'B3', 'B4', 'B5', 'B7'],
                    ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']
                    )

        l4 = ee.ImageCollection('LANDSAT/LT04/C01/T1_SR') \
              .filter(ee.Filter.calendarRange(5, 8, 'month')) \
              .map(maskClouds) \
              .select(['B1', 'B2', 'B3', 'B4', 'B5', 'B7'],
                    ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']
                    )

        landsatCollection = l4.merge(l5).merge(l7).merge(l8)

        return landsatCollection

    def filterOccurrences():
        filterType = [filterParams[0],
                      filterParams[0][filterParams[0]['CXPRESNOST'] == filterParams[1]],
                      filterParams[0][filterParams[0]['CXPRESNOST'] < filterParams[1]],
                      filterParams[0][filterParams[0]['CXPRESNOST'] > filterParams[1]],
                      filterParams[0][filterParams[0]['CXPRESNOST'].isin(list(range(filterParams[1],
filterParams[2]+1)))],
                      filterParams[0][filterParams[0]['CXPRESNOST'] != filterParams[1]],
                      filterParams[0][~filterParams[0]['CXPRESNOST'].isin(list(range(filterParams[1],
filterParams[2]+1)))]
                      ]

        return filterType[filterParams[3]]

    landsatCollection = getLandsat()
    gdf = filterOccurrences()
    output = list()
    yearList = list()
    years = list(range(yearParams[0],
                       yearParams[1]+1,
                       yearParams[2]
                       )
                 )
```

```
    yearsF = [int(str(j)+('0101')) for j in years]

    for yearInx in range(len(years)-1):
        gdfYear = gdf[gdf.DATUM_OD.isin(list(range(yearsF[yearInx],
                                        yearsF[yearInx+1]
                                        )
                                    )
                                )
                    ]

        image = landsatCollection.filterDate(ee.Date.fromYMD(years[yearInx],1,1),
                                    ee.Date.fromYMD(years[yearInx+1],1,1)
                                    ).median()

        chunks = len(gdfYear) // 5000 + 1
        results = list()

        for i in range(chunks):
            gdfYearChunk = gdfYear[i*5000:(i+1)*5000]
            eeYearChunk = ee.FeatureCollection([ee.Feature(feature) for feature in
gdfYearChunk.iterfeatures()])

            try:
                result = pd.read_csv(image.sampleRegions(collection=eeYearChunk,
                                            scale=30,
                                            tileScale=4
                                            ).getDownloadURL())[['DATUM_OD',
                                                    'CXPRESNOST',
                                                    'X',
                                                    'Y',
                                                    'BLUE',
                                                    'GREEN',
                                                    'RED',
                                                    'NIR',
                                                    'SWIR1',
                                                    'SWIR2'
                                                    ]
                                                    ]

                results.append(result)
            except:
                None

        if len(results) != 0:
            df = pd.concat([i for i in results])
            output.append(df)
            yearList.append(years[yearInx])

    spectralDict = dict()
    for i in range(len(output)):
        spectralDict[yearList[i]] = dict()
        for band in ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2', 'CXPRESNOST',
'DATUM_OD']:
            spectralDict[yearList[i]][band] =  output[i][band].to_list()

    for band in ['BLUE', 'GREEN', 'RED', 'NIR', 'SWIR1', 'SWIR2']:
        normList = list()
        for i in spectralDict:
            normList.extend(spectralDict[i][band])
```

```
        bandMin = min(normList)
        bandMax = max(normList)
        for i in spectralDict:
            spectralDict[i][band] = [((x - bandMin)/(bandMax - bandMin)) for x in spectralDict[i][band]]

    return spectralDict
```

příloha č. 4: dialog.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'dialog.ui'
#
# Created by: PyQt5 UI code generator 5.12.3
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(400, 300)
        self.gridLayout_2 = QtWidgets.QGridLayout(Dialog)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.gridLayout_3 = QtWidgets.QGridLayout()
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.errorLabel = QtWidgets.QLabel(Dialog)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.errorLabel.sizePolicy().hasHeightForWidth())
        self.errorLabel.setSizePolicy(sizePolicy)
        self.errorLabel.setMaximumSize(QtCore.QSize(400, 16777215))
        self.errorLabel.setTextFormat(QtCore.Qt.PlainText)
        self.errorLabel.setScaledContents(False)
        self.errorLabel.setWordWrap(True)
        self.errorLabel.setObjectName("errorLabel")
        self.gridLayout_3.addWidget(self.errorLabel, 0, 0, 1, 1,
QtCore.Qt.AlignHCenter|QtCore.Qt.AlignVCenter)
        self.verticalLayout.addLayout(self.gridLayout_3)
        self.okPushButton = QtWidgets.QPushButton(Dialog)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.okPushButton.sizePolicy().hasHeightForWidth())
        self.okPushButton.setSizePolicy(sizePolicy)
        self.okPushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.okPushButton.setFocusPolicy(QtCore.Qt.NoFocus)
        self.okPushButton.setObjectName("okPushButton")
        self.verticalLayout.addWidget(self.okPushButton, 0,
QtCore.Qt.AlignHCenter|QtCore.Qt.AlignBottom)
        self.gridLayout_2.addLayout(self.verticalLayout, 0, 0, 1, 1)

        self.retranslateUi(Dialog)
```

```
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "chyba!"))
        self.errorLabel.setText(_translate("Dialog", "chyba"))
        self.okPushButton.setText(_translate("Dialog", "ok"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

příloha č. 5: requirements.txt

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: win-64
alabaster=0.7.12=pypi_0
altgraph=0.17.2=pypi_0
anyio=3.5.0=pypi_0
argon2-cffi=21.3.0=pypi_0
argon2-cffi-bindings=21.2.0=pypi_0
arrow=1.2.2=pypi_0
astroid=2.9.3=pypi_0
atomicwrites=1.4.0=pypi_0
attrs=21.4.0=pyhd3eb1b0_0
autopep8=1.6.0=pypi_0
babel=2.9.1=pypi_0
backcall=0.2.0=pypi_0
bcrypt=3.2.0=pypi_0
beautifulsoup4=4.10.0=pypi_0
binaryornot=0.4.4=pypi_0
black=22.1.0=pypi_0
blas=1.0=mkl
bleach=4.1.0=pypi_0
bqplot=0.12.33=pypi_0
branca=0.4.2=pypi_0
bzip2=1.0.8=he774522_0
ca-certificates=2022.2.1=haa95532_0
cachetools=5.0.0=pypi_0
certifi=2021.10.8=py38haa95532_2
cffi=1.15.0=pypi_0
chardet=3.0.4=pypi_0
charset-normalizer=2.0.12=pypi_0
click=7.1.2=pyhd3eb1b0_0
click-plugins=1.1.1=pyhd3eb1b0_0
cligj=0.7.2=py38haa95532_0
cloudpickle=2.0.0=pypi_0
colorama=0.4.4=pypi_0
colour=0.1.5=pypi_0
cookiecutter=1.7.3=pypi_0
cryptography=36.0.2=pypi_0
curl=7.80.0=h2bbff1b_0
cycler=0.11.0=pypi_0
debugpy=1.5.1=pypi_0
```

decorator=5.1.1=pypi_0
defusedxml=0.7.1=pypi_0
diff-match-patch=20200713=pypi_0
docutils=0.17.1=pypi_0
earthengine-api=0.1.303=pypi_0
ee-extra=0.0.12=pypi_0
entrypoints=0.4=pypi_0
expat=2.4.4=h6c2663c_0
ffmpeg-python=0.2.0=pypi_0
filelock=3.6.0=pypi_0
fiona=1.8.4=py38h22081e2_0
flake8=4.0.1=pypi_0
folium=0.12.1.post1=pypi_0
fonttools=4.31.1=pypi_0
freexl=1.0.6=h2bbff1b_0
future=0.18.2=pypi_0
gdal=2.3.3=py38hdf43c64_0
gdown=4.4.0=pypi_0
geeadd=0.5.5=pypi_0
geemap=0.12.1=pypi_0
geocoder=1.38.1=pypi_0
geojson=2.5.0=pypi_0
geopandas=0.10.2=pypi_0
geos=3.7.1=h33f27b4_0
google-api-core=2.7.1=pypi_0
google-api-python-client=1.12.11=pypi_0
google-auth=2.6.2=pypi_0
google-auth-httplib2=0.1.0=pypi_0
google-cloud-core=2.2.3=pypi_0
google-cloud-storage=2.2.1=pypi_0
google-crc32c=1.3.0=pypi_0
google-resumable-media=2.3.2=pypi_0
googleapis-common-protos=1.56.0=pypi_0
googledrivedownloader=0.4=pypi_0
hdf4=4.2.13=h712560f_2
hdf5=1.10.4=h7ebc959_0
here-map-widget-for-jupyter=1.1.3=pypi_0
httplib2=0.20.4=pypi_0
httplib2shim=0.0.3=pypi_0
icc_rt=2019.0.0=h0cc432a_1
icu=58.2=ha925a31_3
idna=2.8=pypi_0
imagesize=1.3.0=pypi_0
importlib-metadata=4.11.3=pypi_0
importlib-resources=5.4.0=pypi_0
inflection=0.5.1=pypi_0
intel-openmp=2021.4.0=haa95532_3556
intervaltree=3.1.0=pypi_0
ipyevents=2.0.1=pypi_0
ipyfilechooser=0.6.0=pypi_0
ipykernel=6.9.2=pypi_0
ipyleaflet=0.15.0=pypi_0
ipynb-py-convert=0.4.6=pypi_0
ipython=7.32.0=pypi_0
ipython-genutils=0.2.0=pypi_0
ipytree=0.2.1=pypi_0
ipywidgets=7.7.0=pypi_0
isort=5.10.1=pypi_0
jedi=0.18.1=pypi_0
jellyfish=0.9.0=pypi_0

jinja2=3.0.3=pypi_0
jinja2-time=0.2.0=pypi_0
jpeg=9d=h2bbff1b_0
json5=0.9.6=pypi_0
jsonschema=4.4.0=pypi_0
jupyter=1.0.0=pypi_0
jupyter-client=6.1.12=pypi_0
jupyter-console=6.4.3=pypi_0
jupyter-core=4.9.2=pypi_0
jupyter-server=1.15.6=pypi_0
jupyterlab=3.3.2=pypi_0
jupyterlab-pygments=0.1.2=pypi_0
jupyterlab-server=2.11.2=pypi_0
jupyterlab-widgets=1.1.0=pypi_0
kealib=1.4.7=h07cbb95_6
keyring=23.5.0=pypi_0
kiwisolver=1.4.0=pypi_0
krb5=1.16.4=hc04afaa_0
lazy-object-proxy=1.7.1=pypi_0
libboost=1.73.0=h6c2663c_11
libcurl=7.80.0=h86230a5_0
libgdal=2.3.3=h10f50ba_0
libiconv=1.15=h1df5818_7
libkml=1.3.0=hd124aa8_5
libnetcdf=4.6.1=h411e497_2
libpng=1.6.37=h2a8f88b_0
libpq=11.2=h3235a2c_0
libspatialite=4.3.0a=hc36aec2_19
libssh2=1.9.0=h7a1dbc1_1
libtiff=4.2.0=hd0e1b90_0
libxml2=2.9.12=h0ad7f3c_0
logzero=1.7.0=pypi_0
lz4-c=1.9.3=h2bbff1b_1
markupsafe=2.1.1=pypi_0
matplotlib=3.5.1=pypi_0
matplotlib-inline=0.1.3=pypi_0
mccabe=0.6.1=pypi_0
mistune=0.8.4=pypi_0
mkl=2021.4.0=haa95532_640
mkl-service=2.4.0=py38h2bbff1b_0
mkl_fft=1.3.1=py38h277e83a_0
mkl_random=1.2.2=py38hf11a4ad_0
mss=6.1.0=pypi_0
munch=2.5.0=pyhd3eb1b0_0
mypy-extensions=0.4.3=pypi_0
nbclassic=0.3.7=pypi_0
nbclient=0.5.13=pypi_0
nbconvert=6.4.4=pypi_0
nbformat=5.2.0=pypi_0
ndop-downloader=0.1.6=pypi_0
nest-asyncio=1.5.4=pypi_0
notebook=6.4.10=pypi_0
notebook-shim=0.1.0=pypi_0
numpy=1.21.5=py38ha4e8547_0
numpy-base=1.21.5=py38hc2deb75_0
numpydoc=1.2=pypi_0
openssl=1.1.1m=h2bbff1b_0
owslib=0.25.0=pypi_0
packaging=21.3=pypi_0
palettable=3.3.0=pypi_0

pandas=1.4.1=pypi_0
pandocfilters=1.5.0=pypi_0
paramiko=2.10.3=pypi_0
parso=0.8.3=pypi_0
pathspec=0.9.0=pypi_0
pcre=8.45=hd77b12b_0
pefile=2021.9.3=pypi_0
pexpect=4.8.0=pypi_0
pickleshare=0.7.5=pypi_0
pillow=9.0.1=pypi_0
pip=21.2.2=py38haa95532_0
platformdirs=2.5.1=pypi_0
plotly=5.6.0=pypi_0
pluggy=1.0.0=pypi_0
poyo=0.5.0=pypi_0
proj4=5.2.0=ha925a31_1
prometheus-client=0.13.1=pypi_0
prompt-toolkit=3.0.28=pypi_0
protobuf=3.19.4=pypi_0
psutil=5.9.0=pypi_0
ptyprocess=0.7.0=pypi_0
pyasn1=0.4.8=pypi_0
pyasn1-modules=0.2.8=pypi_0
pycodestyle=2.8.0=pypi_0
pycparser=2.21=pypi_0
pycrs=1.0.2=pypi_0
pydocstyle=6.1.1=pypi_0
pyflakes=2.4.0=pypi_0
pygments=2.11.2=pypi_0
pyinstaller=4.10=pypi_0
pyinstaller-hooks-contrib=2022.2=pypi_0
pylint=2.12.2=pypi_0
pyls-spyder=0.4.0=pypi_0
pynacl=1.5.0=pypi_0
pyparsing=3.0.7=pypi_0
pyproj=3.3.0=pypi_0
pyqt5=5.12.3=pypi_0
pyqt5-sip=12.9.1=pypi_0
pyqtwebengine=5.12.1=pypi_0
pyrsistent=0.18.1=pypi_0
pyshp=2.2.0=pypi_0
pysocks=1.7.1=pypi_0
python=3.8.12=h6244533_0
python-box=5.4.1=pypi_0
python-dateutil=2.8.2=pypi_0
python-lsp-black=1.1.0=pypi_0
python-lsp-jsonrpc=1.0.0=pypi_0
python-lsp-server=1.3.3=pypi_0
python-slugify=6.1.1=pypi_0
pytz=2022.1=pypi_0
pywin32=303=pypi_0
pywin32-ctypes=0.2.0=pypi_0
pywinpty=2.0.5=pypi_0
pyyaml=6.0=pypi_0
pyzmq=22.3.0=pypi_0
qdarkstyle=3.0.2=pypi_0
qstylizer=0.2.1=pypi_0
qtawesome=1.1.1=pypi_0
qtconsole=5.2.2=pypi_0
qtpy=2.0.1=pypi_0

ratelim=0.1.6=pypi_0
requests=2.21.0=pypi_0
rope=0.23.0=pypi_0
rsa=4.8=pypi_0
rtree=0.9.7=pypi_0
sankee=0.0.7=pypi_0
send2trash=1.8.0=pypi_0
setuptools=58.0.4=py38haa95532_0
shapely=1.6.4=py38h222a598_0
six=1.16.0=pyhd3eb1b0_1
sniffio=1.2.0=pypi_0
snowballstemmer=2.2.0=pypi_0
sortedcontainers=2.4.0=pypi_0
soupsieve=2.3.1=pypi_0
sphinx=4.4.0=pypi_0
sphinxcontrib-applehelp=1.0.2=pypi_0
sphinxcontrib-devhelp=1.0.2=pypi_0
sphinxcontrib-htmlhelp=2.0.0=pypi_0
sphinxcontrib-jsmath=1.0.1=pypi_0
sphinxcontrib-qthelp=1.0.3=pypi_0
sphinxcontrib-serializinghtml=1.1.5=pypi_0
spyder=5.2.2=pypi_0
spyder-kernels=2.1.3=pypi_0
sqlite=3.38.0=h2bbff1b_0
tenacity=8.0.1=pypi_0
terminado=0.13.3=pypi_0
testpath=0.6.0=pypi_0
text-unidecode=1.3=pypi_0
textdistance=4.2.2=pypi_0
three-merge=0.1.1=pypi_0
tinycss2=1.1.1=pypi_0
tk=8.6.11=h2bbff1b_0
toml=0.10.2=pypi_0
tomli=2.0.1=pypi_0
tornado=6.1=pypi_0
tqdm=4.63.0=pypi_0
traitlets=5.1.1=pypi_0
traittypes=0.2.1=pypi_0
typing-extensions=4.1.1=pypi_0
ujson=5.1.0=pypi_0
uritemplate=3.0.1=pypi_0
urllib3=1.24.3=pypi_0
vc=14.2=h21ff451_1
voila=0.3.4=pypi_0
vs2015_runtime=14.27.29016=h5e58377_2
watchdog=2.1.6=pypi_0
wcwidth=0.2.5=pypi_0
webencodings=0.5.1=pypi_0
websocket-client=1.3.1=pypi_0
websockets=10.2=pypi_0
wheel=0.37.1=pyhd3eb1b0_0
whitebox=2.1.2=pypi_0
whiteboxgui=0.7.0=pypi_0
widgetsnbextension=3.6.0=pypi_0
wincertstore=0.2=py38haa95532_2
wrapt=1.13.3=pypi_0
xerces-c=3.2.3=ha925a31_0
xlsxwriter=3.0.3=pypi_0
xyzservices=2022.3.0=pypi_0
xz=5.2.5=h62dcd97_0

yapf=0.32.0=pypi_0
zipp=3.7.0=pypi_0
zlib=1.2.11=h8cc25b3_4
zstd=1.4.9=h19a0ad4_0

příloha č. 6: online odkaz na složku se zdrojovými soubory a .exe soubor nástroje

https://czuvpraze-my.sharepoint.com/:f:/g/personal/xkucm053_studenti_czu_cz/Ep4EwTH9TLVChUd5k4z4aqMBQhqCPpdZIobQHAr8zPDoDA?e=GAxIli