Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

AUTOMATIZACE TVORBY HODNOTOVÝCH MĚŘÍTEK KARTODIAGRAMŮ V PROSTŘEDÍ ARCGIS FOR DESKTOP

Magisterská práce

Bc. Radka NOVÁKOVÁ

Vedoucí práce RNDr. Alena VONDRÁKOVÁ, Ph.D.

Olomouc 2016
Geoinformatika

ANOTACE

Hlavním cílem magisterské práce je automatizace tvorby hodnotových měřítek kartodiagramů v prostředí ArcGIS for Desktop. Kartodiagramy jsou nejčastěji používány pro prezentování statistických údajů a řadí se do skupiny tzv. statistických map. Podle vztažného bodu se kartodiagramy dělí na bodové, liniové a plošné. Na rozdíl od kartogramů vyjadřují data výhradně v absolutních hodnotách.

V současné době existuje řada programů, ve kterých lze vytvořit kartodiagram. Často však není proces tvorby kvalitní (nedodržuje kartografické zásady), nebo je omezený (nabízí jen část z mnoha druhů kartodiagramů). Obecně velkým problémem je sestrojení vhodného a odpovídajícího hodnotového měřítka. Obvykle se v programech při tvorbě kartodiagramu nevyskytuje a uživatel musí použít pro jeho tvorbu nějaký grafický program.

V této magisterské práci bude snaha tento problém vyřešit a nabídnout prostřednictvím svého nástroje s podrobným návodem automatickou tvorbu hodnotových měřítek pro několik typů kartodiagramů.

KLÍČOVÁ SLOVA

kartodiagram; automatizace; hodnotové měřítko; Tkinter; Python

Počet stran práce: 72

Počet příloh: 13 (12 vázaných, 1 volná)

ANOTATION

The main aim of the thesis is an Automation of cartodiagram scales creation in ArcGIS Desktop. Cartodiagrams are most often used for presenting statistical data and belong to the group - Statistical maps. According to the reference point we can divide cartodiagrams into the point, line and area. Unlike choropleth maps data are exclusively expressed in absolute values.

Currently there are a number of programs which can create the cartodiagrams. The process of creating is not often in a high-quality (not following the cartographic principles) or the process is limited (offering only part of many types of the cartodiagrams). Generally, a major problem is to create a suitable and corresponding diagram scale. Usually we can not find the diagram scales in programs and user must use for this creation some graphic programs.

In this thesis there will be an attempt to solve this problem and offer an Automation of cartodiagram scales creation for several types of cartodiagrams through its own tools with detailed instructions.

KEYWORDS

cartodiagram; automation; diagram scale; Tkinter; Python

Number of pages 72
Number of appendixes 13

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracovala samostatně a uvedla jsem všechny použité podklady a literaturu.

- jsem si vědoma, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, aby jeden výtisk diplomové práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí,

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

Olomouc dne

Bc. Radka Nováková

podpis autora

Děkuji vedoucí práce RNDr. Aleně VONDRÁKOVÉ, Ph.D. za podněty a připomínky při vypracování práce.

OBSAH

ÚVOD

Kartodiagramy jsou nejčastěji používány pro prezentování statistických údajů a řadí se do skupiny tzv. statistických map. Podle vztažného bodu se kartodiagramy dělí na bodové, liniové a plošné. Na rozdíl od kartogramů vyjadřují data výhradně v absolutních hodnotách. Kartodiagramy se vyskytují v odborných publikacích kartografie, ale také v oborech, které se pracují se statistickými daty např. demografie, ekonomie.

V současné době existuje řada programů, ve kterých lze vytvořit kartodiagram. Největší zastoupení mají GIS softwary, ve kterých se vytváří mapy a uživateli se nabízí doplnit mapu nějakým vhodným kartodiagramem. Sestrojit jej lze i v několika statistických či grafických programech. Často však není proces tvorby kvalitní (nedodržuje kartografické zásady), nebo je omezený (nabízí jen část z mnoha druhů kartodiagramů). Obecně velkým problémem je sestrojení vhodného a odpovídajícího hodnotového měřítka. Tvorba hodnotového měřítka obvykle není možná v programech při tvorbě kartodiagramu a uživatel musí použít pro jeho tvorbu nějaký grafický program. Nutno podoktnout, že tato tvorba je časově náročná a uživatelsky nekonformní.

Autorka se v této magisterské práci snaží tento problém vyřešit a nabídnout prostřednictvím svého nástroje s podrobným návodem automatickou tvorbu hodnotových měřítek pro několik typů kartodiagramů.

1 CÍLE PRÁCE

Hlavním cílem magisterské práce byla **automatizace tvorby hodnotových měřítek kartodiagramů v prostředí ArcGIS for Desktop**. Dílčí cíle se dělí na teoretické a praktické. V **teoretické** části práce byly stanoveny následující cíle:

- podrobná **rešerše literatury** (české i zahraniční) věnující se problematice kartodiagramů se zaměřením na tvorbu příslušných hodnotových měřítek pro různé typy kartodiagramů.
- **ověření funkcionality** dostupných nástrojů na tvorbu různých typů kartodiagramů a k tomu odpovídajících hodnotových měřítek (v různých softwarových nástrojích).

V **praktické** části byl výstupem práce **návod a aplikace** (toolbox nebo jiný vhodný nástroj), jakým bude možné **vytvářet hodnotová měřítka** pro různé typy kartodiagramů realizovaných v softwaru ArcGIS for Desktop, a to s takovým výstupem, který bude dále využitelný (volba velikosti měřítka, fontu popisu, export do křivkového formátu apod.). Po naprogramování aplikace byla ověřena její funkcionalita na **ukázkových příkladech**.

V závěru práce byly vyhotoveny webové stránky, DVD s aplikací přiložené k textu práce a další formální náležitosti.

2 METODY A POSTUPY ZPRACOVÁNÍ

Na počátku samotného zpracování diplomové práce bylo nezbytně nutné si detailněji nastudovat literaturu zabývající se tematickou kartografií, vyjadřovacími metodami a konkrétně diagramy a kartodiagramy. Dále bylo zapotřebí najít a otestovat programy, které by mohly kartodiagramy vytvářet a jestli dokáží vytvořit hodnotové měřítko pro tyto kartodiagramy. A na základě těchto znalostí naprogramovat vhodné vlastní prostředí pro tvorbu hodnotových měřítek kartodiagramů.

2.1 Postup zpracování

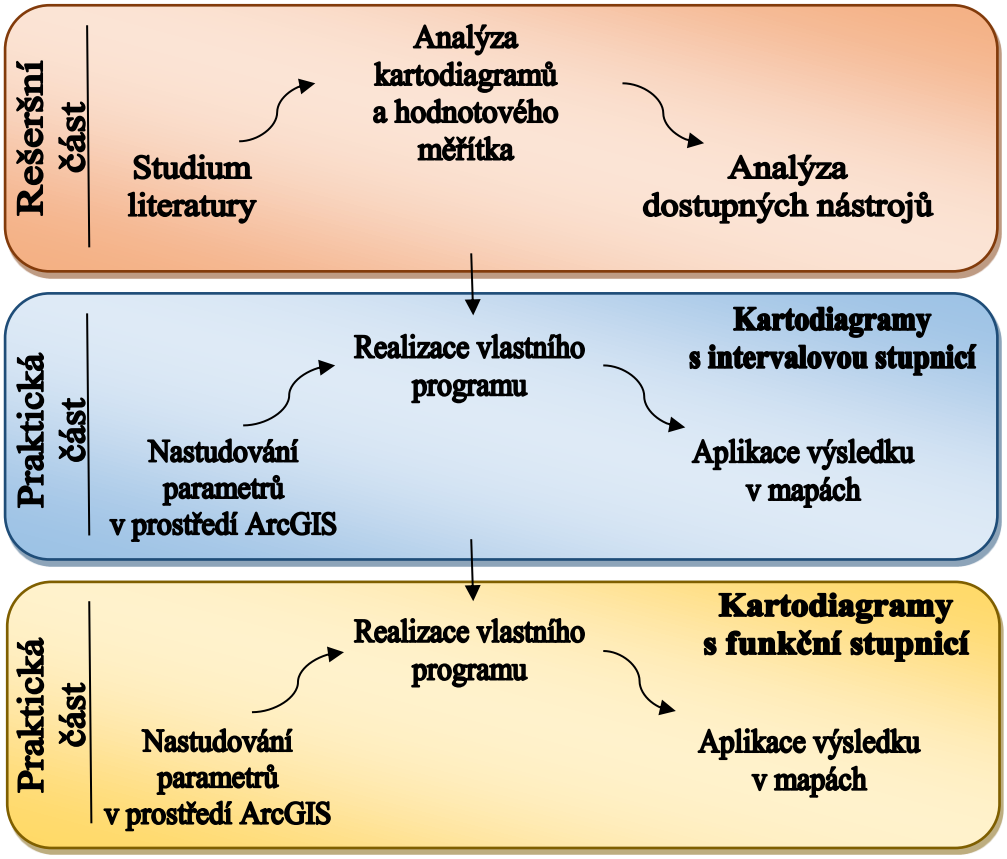


Schéma č. 1 – Postup zpracování diplomové práce.

2.2 Použité metody

Jelikož byl program navrhnut pro prostředí ArcGIS, byly zde nejprve nastudovány všechny typy kartodiagramů a jejich parametrů. Následovala analýza těchto všech parametrů a bylo zhodnoceno, které parametry jsou pro tvorbu hodnotových měřítek nezbytné, ty byly převzaty do vlastního programu nezměněny, a ty, které budou do programu nasazeny v určité obměně. Některé parametry byly ve vlastním programu vynechány.

Následovala samotná programátorská činnost. Pro každý příkaz byla dohledána dokumentace, aby byl příkaz správně napsán a interpretován. První byly definovány

všechny potřebné parametry pro tvorbu hodnotových měřítek, dále následovala výpočetní část neboli jádro programu. Ta byla pro každý program trochu jiná. Pro kartodiagramy s využitím funkční stupnice bylo nutné nastudovat dvě metody, které prostředí ArcGIS nabízí pro výpočet jejich velikostí a to metodu tzv. Normal case a Flannery metodu. Vzorce a výpočet byl popsán v kapitole 5.4.2 Vysvětlení parametrů. Poslední částí každého programu byla část věnovaná exportu hodnotového měřítka – volba formátu a místa uložení na disku.

2.3 Použitá data

K naprogramování vlastního programu nebylo nutné použít žádná data. Data byla potřebná až při testování funkčnosti vytvořeného programu, zda program správně pracuje a aby se případně odhalily nedostatky. Pro testovací účely byla použita data z ArcČR 500 od společnosti ARCDATA PRAHA. Byly využity vrstvy okresů, krajů a obcí. Tematickou vrstvu pro vytvoření ukázek použití hodnotových měřítek v mapách zajišťovala stažená data ze stránek Českého statistického úřadu.

2.4 Použité programy

Nejvíce využívaným programem byl produkt ArcGIS for Desktop verze 10.2, ve kterém probíhalo nastudování implementované nabídky pro tvorbu kartodiagramů, jejich parametrů a v konečné fázi zde byly vytvořeny ukázky vytvořených hodnotových měřítek.

V produktu ArcGIS for Deskop verze 10.2 probíhala i prvotní realizace automatické tvorby hodnotových měřítek kartodiagramů, konkrétně v Python Toolboxu.

Konečná realizace se uskutečnila ve volně dostupném modulu Tkinter, založeném na jazyce Python – v tomto případě Python 2.7. Nutností bylo naimportovat několik knihoven, aby program dokázal bez problémů fungovat. Nejvyužívanějšími knihovnami byly samotný Tkinter, Matplotlib, Os, Math, Sys, Arcpy a PIL. Všechny knihovny jsou blíže popsány a vysvětleny v kapitolách jednotlivých programů, u kterých byly použity.

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Tato kapitola je věnována seznámení s metodou kartodiagramu a na jaké druhy a typy člení kartodiagramy domácí či zahraniční autoři. Dále zde budou rozebrány a zhodnoceny dostupné nástroje pro tvorbu kartodiagramů, jejich měřítek a legend.

3.1 Kartodiagramy v kartografii

Kartodiagram je mapa s dílčími územními celky, do kterých jsou diagramy znázorněna statistická data (absolutní hodnoty), většinou geografického charakteru (Kaňok, 1999). Podle Voženílka (2004) patří kartodiagramy do skupiny tzv. statistických map. Nejčastěji se používají k prezentaci statistických údajů. V novější publikaci Voženílka, Kaňoka a kol. (2011) jsou za kartodiagramy považována mapová díla pro znázorňování kvantity, především pro znázorňování absolutních hodnot jevu. Jsou vhodné především pro srovnání konkrétních hodnot v dílčích územních jednotkách na mapě, např. počet obyvatel, objem výroby, poměr vývozu a dovozu apod. K jejich tvorbě se používají výhradně proporcionálně a gradované (vzestupně) sestavené stupnice diagramů.

Co se týče srovnání terminologie v zahraničí a v České republice, může dojít k určitým nesrovnalostem. Český termín „kartogram“ neznamená v zahraničí „cartogram“, což označuje anamorfózu a tedy i pojem „cartodiagram“ v podstatě jako termín neexistuje. Pro jejich označení se používají spíše pojmy jako „proportional symbol map“ nebo „graduated symbol map“. Podle Indiemapper (2010) jsou kartodiagramy velmi flexibilní, protože zde mohou být použity buď číselné údaje (např. import/export, věk), nebo odstupňované kategorické údaje (např. nízké, střední a vysoké riziko úpadku). Jsou také flexibilní, protože mohou být použity pro údaje vztahující se ke geografickým bodům (např. přesná lokace místa), nebo údaji vztahujícími se k zeměpisným oblastem (např. státu). Dále uvádí, že výhodou kartodiagramů oproti tečkové metodě je, že je obecně snazší pro čtenáře mapy extrahovat data z mapy, protože odhad velikosti symbolu je méně pracný než počítání mnoho malých teček. Výhodou kartodiagramů oproti kartogramům je volba možnosti použití buď surových dat (součty, počty) nebo standardizovaných dat (procenta, ceny, poměrů).

3.1.1 Dělení kartodiagramů podle reference

Podle Voženílka, Kaňoka a kol. (2011) se podle reference k typu bodu, linii nebo ploše volí způsob umisťování diagramů a celková kompozice kartodiagramu. Kaňok (1999) rozlišuje kartodiagramy podle vztažných prvků na **bodové, liniové a plošné.**

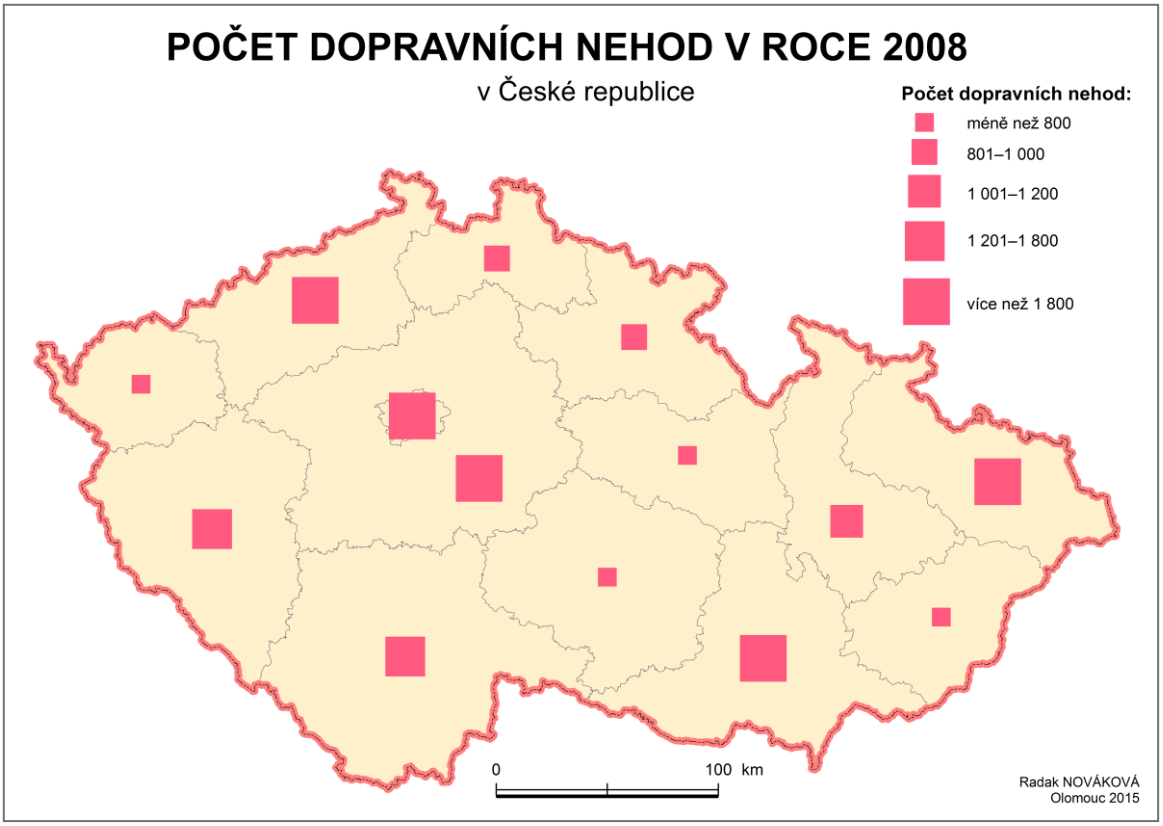
Bodové kartodiagramy se od plošných liší tím, že jejich data nejsou vztažena k celé ploše, nýbrž k určitému místu/bodu (např. městu, meteorologické stanici). **Plošné** diagramy znázorňují charakteristiky jevů pomocí diagramů vztažených k ploše, čili k světadílu, státu, přírodním jednotkám – povodím apod. **Liniové** kartodiagramy se nejčastěji vztahují k linii – silnici, řece, železnici. Lze jimi vyjádřit velikost jevu, ale i směr pohybu.

3.1.2 Dělení podle způsobu konstrukce a počtu znázorněných jevů

Dělení kartodiagramů podle způsobu konstrukce a počtu znázorněných jevů popsal Kaňok (1999) ve své publikaci a kartodiagramy rozdělil na: jednoduché, složené, strukturní, součtové, srovnávací, dynamické, segmentové, vektorové, stuhové, anamorfózní a jiné.

Jednoduchý kartodiagram

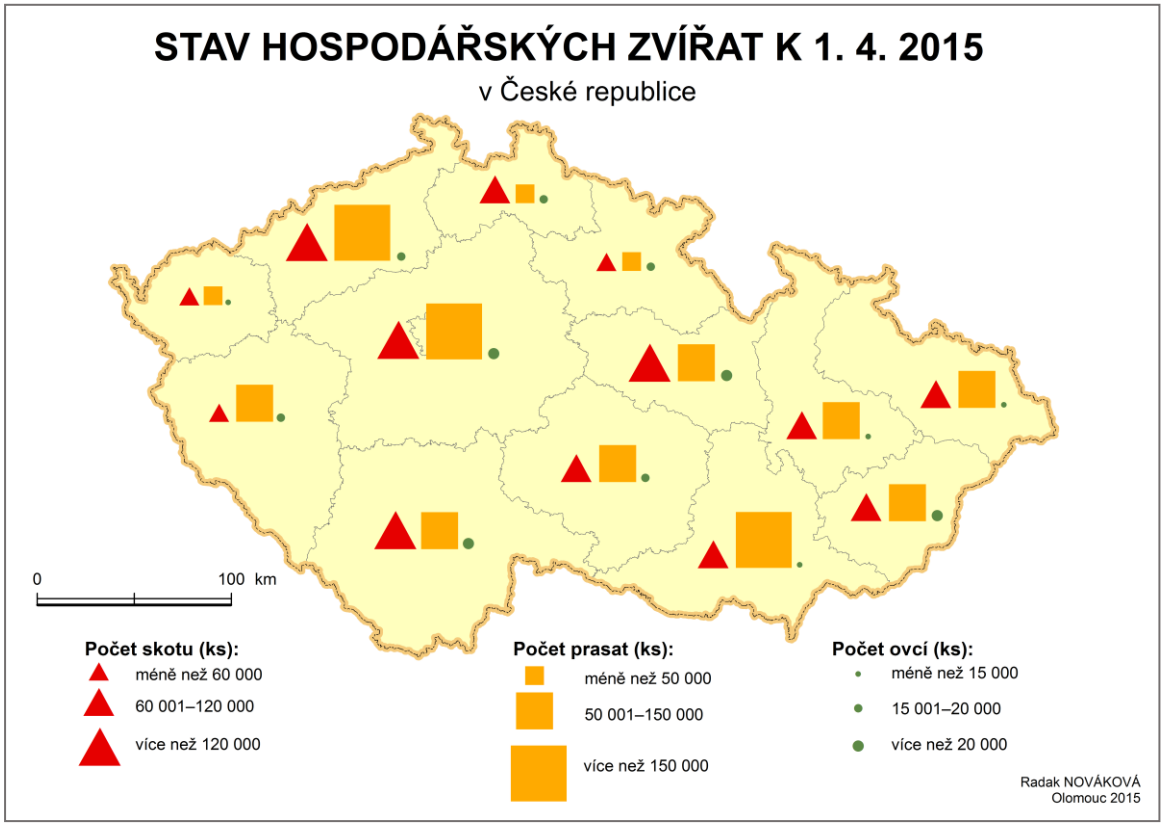
Jednoduchý kartodiagram zobrazuje jen jeden jev, tedy pomocí jednoho geometrického tvaru (Kaňok, 1999).



Obr. 1 Jednoduchý kartodiagram.

Složený kartodiagram

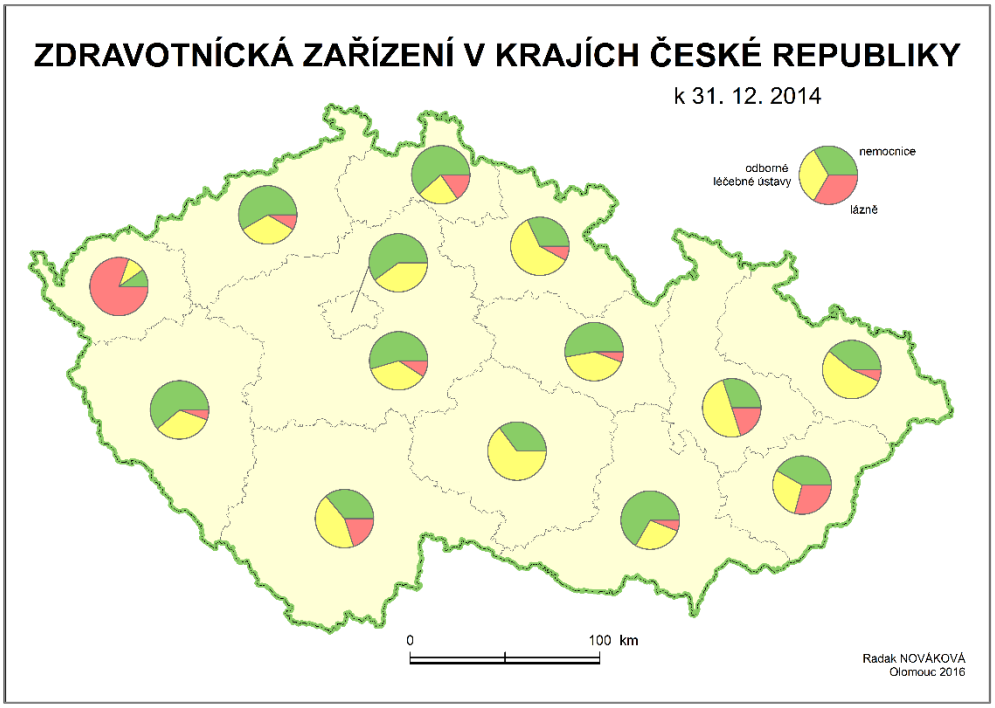
Na rozdíl od jednoduchého kartodiagramu znázorňuje více jevů současně a to buď pomocí diagramu jiných typů nebo stejným typem, ovšem rozlišeným barvou či rastrem. Všechny mohou být zpracovány buď v jedné, nebo více jednotkách (Voženílek, Kaňok a kol. 2011).



Obr. 2 Složený kartodiagram (Kaňok, 1999).

Strukturní kartodiagram

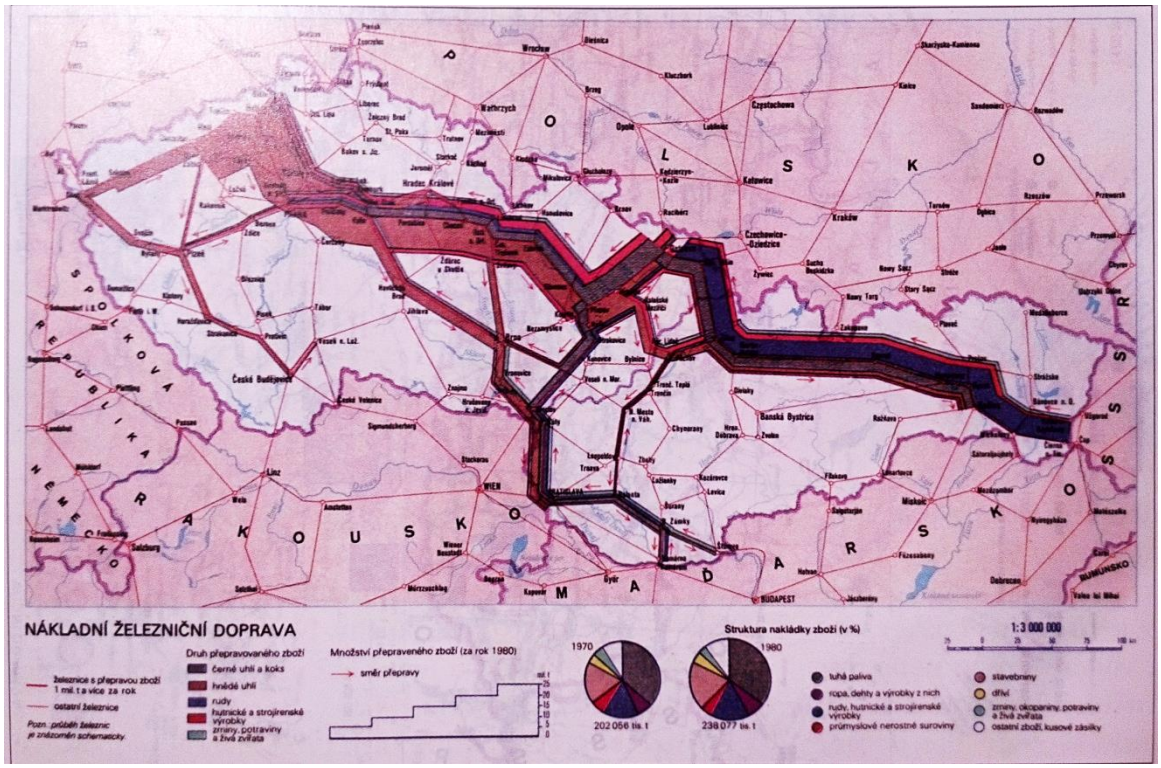
Tento druh kartodiagramu obsahuje diagramy stejné velikosti strukturně dělené pro znázornění jevů v bodech, liniích nebo plochách, nelze však zjistit absolutní hodnotu jevu, součet všech dílčích částí dává součet 100% (Voženílek, Kaňok a kol. 2011).



Obr. 3 Strukturní kartodiagram.

Součtový kartodiagram

Součtový kartodiagram obsahuje soubor diagramů vztažených k bodu, linii nebo ploše, kde každý z nich zobrazuje velikost sledovaného jevu v absolutních hodnotách a zároveň znázorňuje vnitřní strukturu jevu. Celkové velikosti diagramů narůstají, jsou vyjádřeny součtem jednotlivých dílčích částí (Voženílek, Kaňok a kol. 2011).

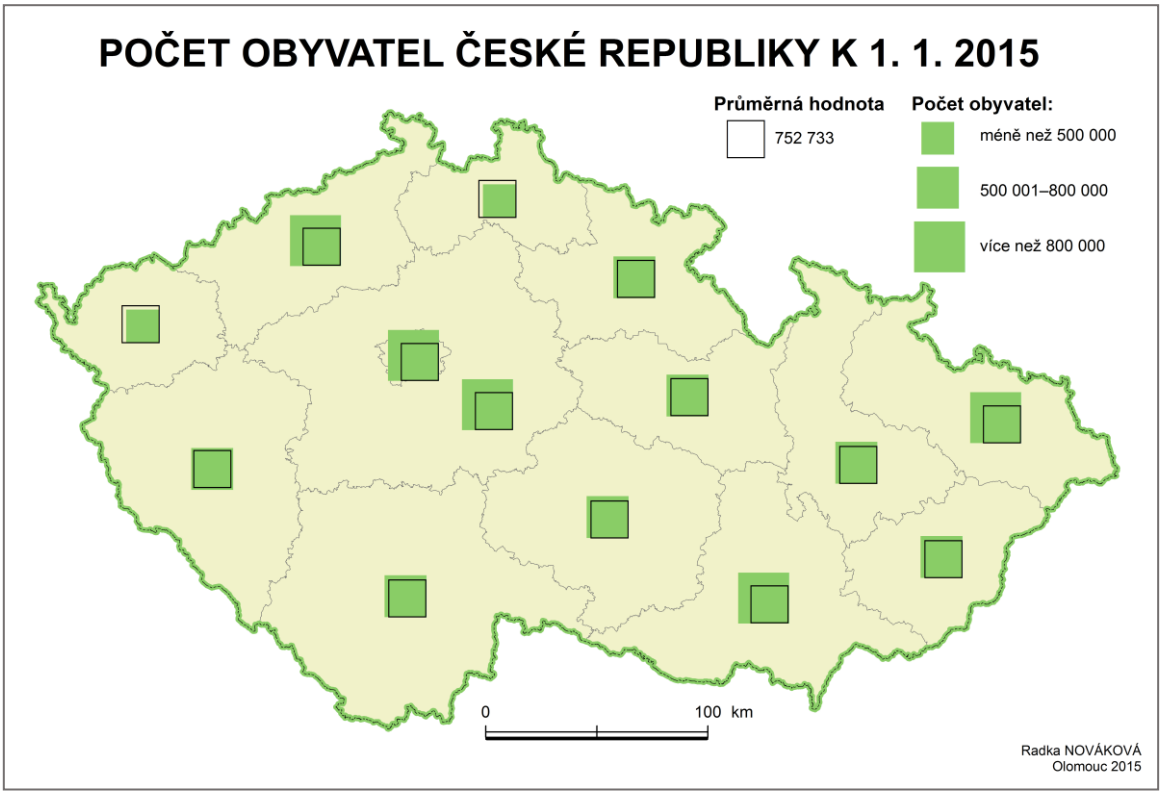


Obr. 4 Součtový kartodiagram (Voženílek, Kaňok a kol. 2011).

Srovnávací kartodiagram

Podle Kaňoka (1999), je srovnávací kartodiagram soubor diagramů v mapě, kde každý z nich je složen ze dvou diagramů. Jeden z nich má stálou velikost a je obyčejně zaznamenán do map jen v podobě obrysů. Prezentuje se jím většinou střední hodnota jevu

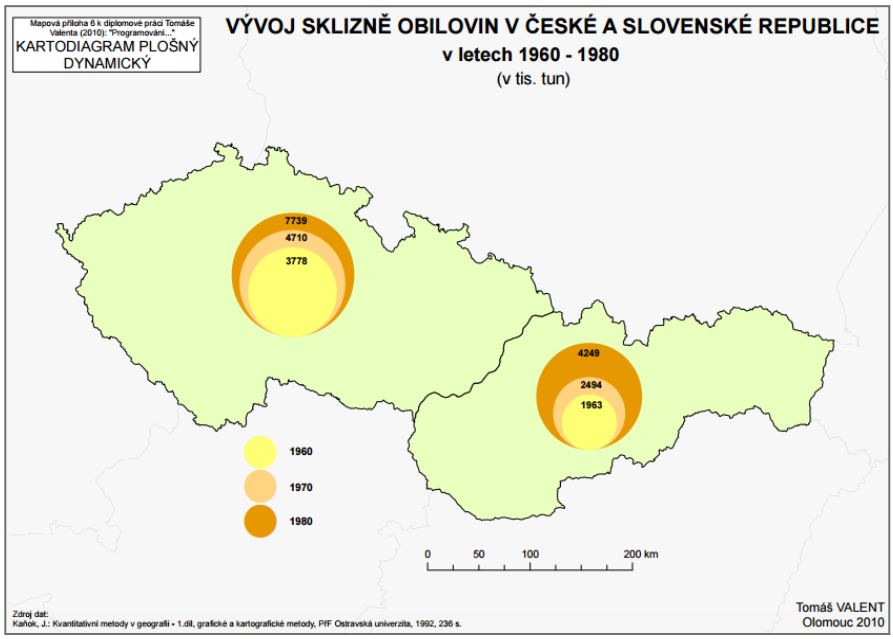
na sledovaném území, nebo velikost optimální, výchozí, nebo perspektivní. Velikost druhé části srovnávacího diagramu závisí na velikost jevu v daném bodě, nebo v dílčí části území.



Obr. 5 Srovnávací kartodiagram.

Dynamický kartodiagram

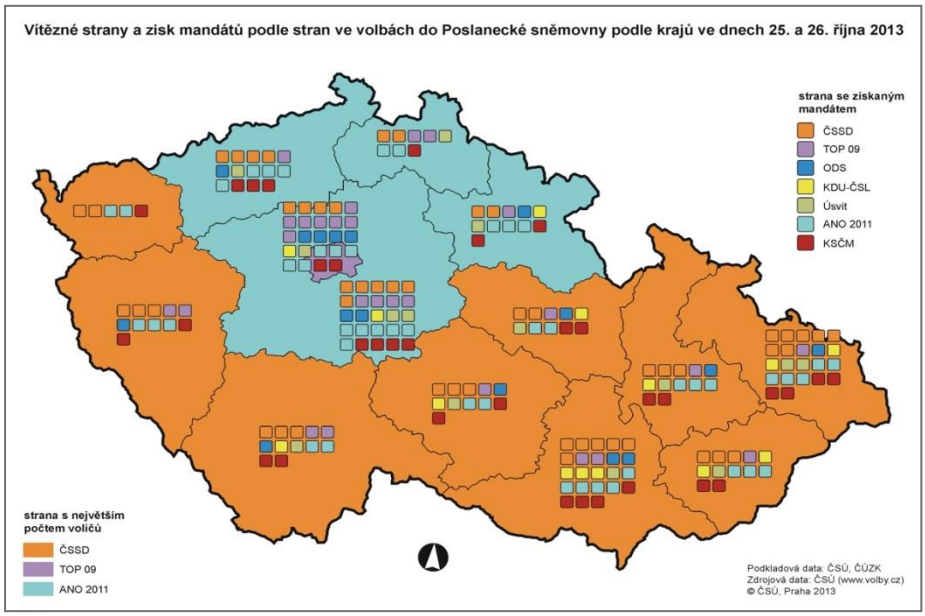
Podle Voženílka, Kaňoka a kol. (2011) vyjadřuje dynamický kartodiagram časově proměnlivé jevy. Ke znázornění se používají dynamické diagramy, které zobrazují minimálně tři časové údaje o jevu.



Obr. 6 Dynamický kartodiagram [4].

Segmentový kartodiagram

Voženílek, Kaňok a kol. (2011) definují segmentový kartodiagram jako geometrický obrazec s celkovou hodnotou jevu danou součtem jednotlivých segmentů o různých hodnotách, kde za segmenty se často používají geometrické, výjimečně symbolické znaky. Tyto segmenty jsou uspořádány do větších pravidelných obrazců, např. řádků, čtverců, obdélníků apod. a čtenář mapy jejich součtem odhaduje celkovou hodnotu jevu.



Obr. 7 Segmentový kartodiagram [5].

Liniový kartodiagram

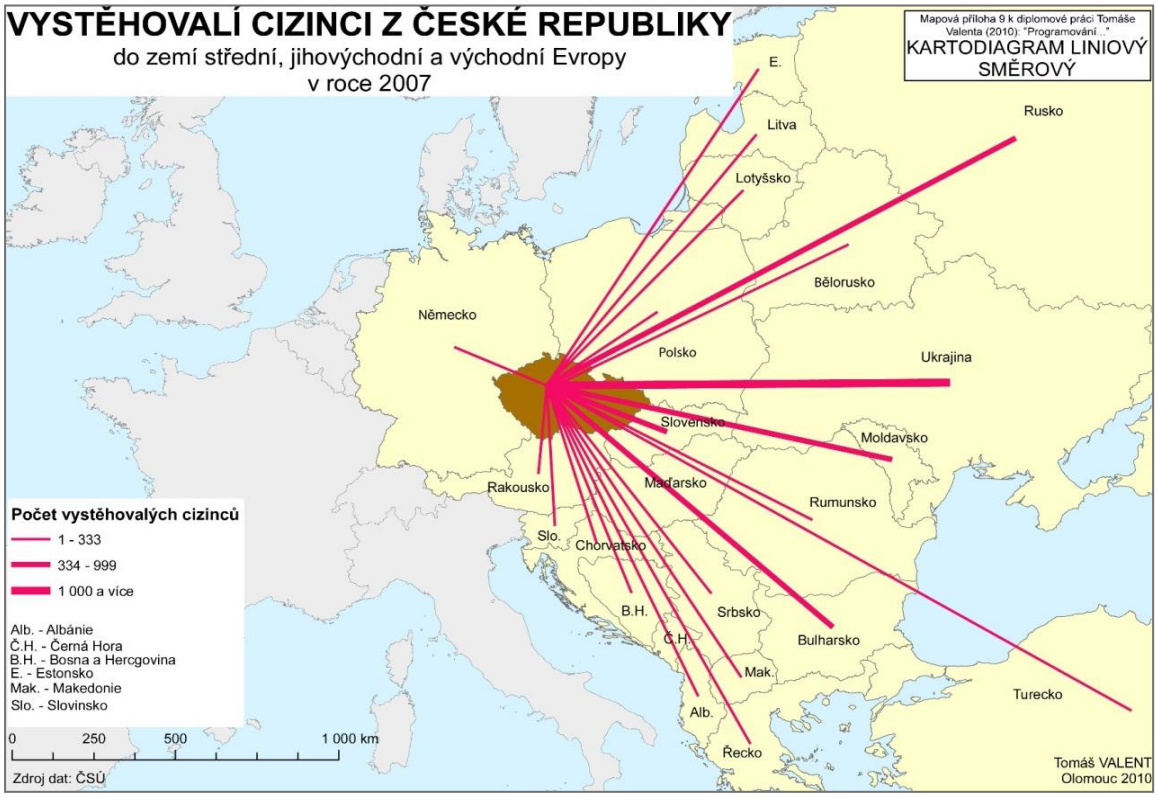
Jak již bylo výše zmíněno, liniový kartodiagram podává dvě informace o jevu a to jeho velikost a směr. Díky tomu se také liniové kartodiagramy rozlišují na vektorové a stuhové.

Vektorový diagram je definován svým počátečním bodem, směrem a délkou vektoru. Typickými vektorovými diagramy jsou tzv. větrné růžice používající se v meteorologii, konkrétně u meteorologických stanic. Jsou charakteristické svým středem, směry světových stran a délkami vektorů reprezentujících intenzitu jevu. Vložením vektorového diagramu do mapy vzniká vektorový kartodiagram. Tento druh kartodiagramu se dá dále rozlišit na kartodiagram vektorový proudový a dosahový (Voženílek, Kaňok a kol. 2011).

Vektorový proudový kartodiagram nemá podle Kaňoka (1999) jeden centrální bod, skládá se z proudu nebo trsu šipek vhodně lokalizovaných a podle sledovaného jevu správně směrově orientovaných kartodiagramů se používá často v mapách mořských proudů a mapách převažujících směrů větrů v určitém období. Někdy se délka a šířka šipek využívá nejen k vyjádření kvality (studený, teplý mořský proud), ale i kvantity (rychlost proudění).

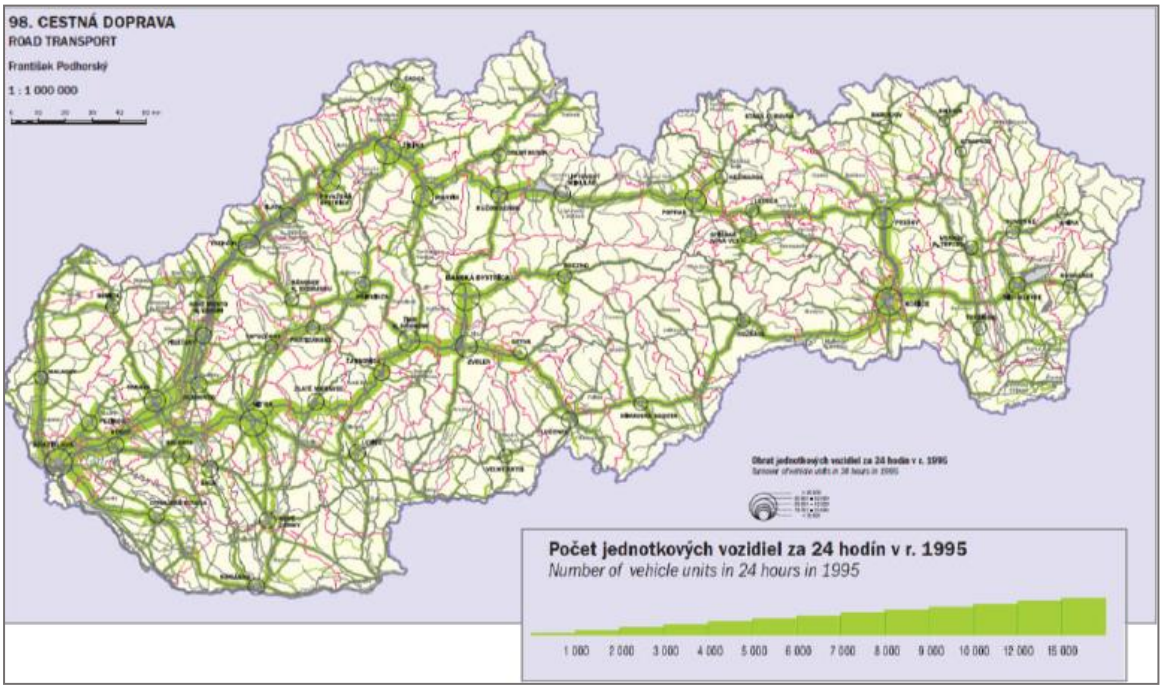
Obr. 8 Vektorový proudový kartodiagram [6].

Vektorový dosahový kartodiagram na rozdíl od předchozího typu nemá jeden centrální bod a vytváří se tak, že z určitého centrálního bodu se rýsují rovné nebo zaoblené čáry k jiným bodům či plochám, se kterými je centrální bod v nějaké souvislosti. Tímto centrálním bodem může být i plocha (stát, povodí, apod.). Při konstrukci jsou preferovány tři vlastnosti vektorů – centrální bod, směr, dosahový bod (Voženílek, Kaňok a kol. 2011)



Obr. 9 Vektorový dosahový kartodiagram [4].

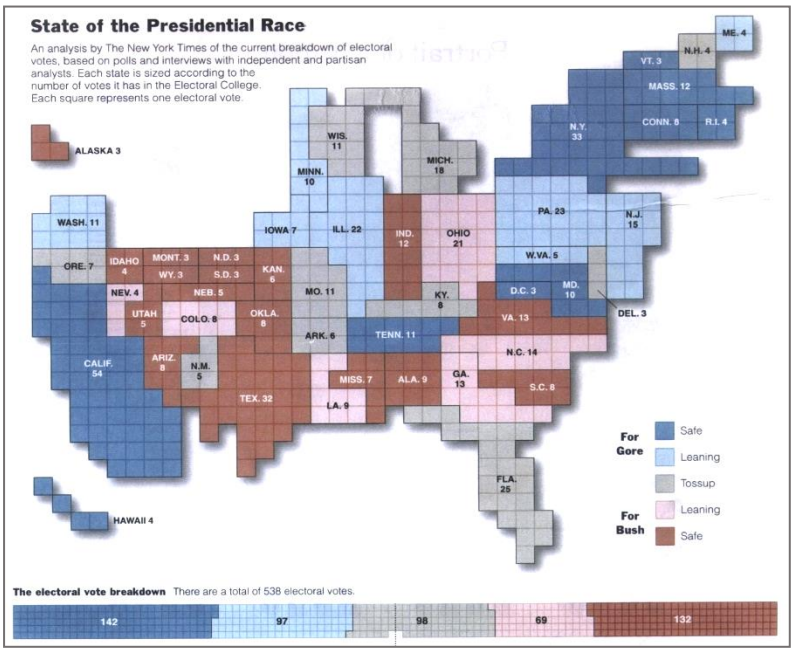
Stuhový kartodiagram zachovává reálný průběh čar a ukazuje i podíly přemísťované kvantity jevu. Číselná hodnota je vyjádřena celkovou šířkou stuhy. Od předcházejících typů liniových kartodiagramů se liší tím, že směr linie znázorňující průběh přemísťování jevu se mění (Kaňok, 1999).



Obr. 10 Stuhový kartodiagram [7].

Anamorfózní kartodiagram

Je to plošný kartodiagram, který svou sestavou diagramů v ploše naznačuje tvar sledovaného území. Jednotlivé diagramy jsou přiřazeny k dílčím plochám tvořící celek. Nejčastěji to jsou vyšší administrativní celky ve státě (spolkové republiky, provincie, kraje, obvody). Důležité je, že celkový dojem plošně uspořádaných diagramů tvoří vyšší administrativní celek. K předvedení sledovaného jevu se používá různých diagramů, které zároveň určují druh anamorfózního kartodiagramu (Kaňok, 1999).



Obr. 11 Anamorfózní kartodiagram [8].

3.1.3 Hodnotové měřítko

Na tematických mapách jsou jevy a objekty znázorněny diagramy, plošnými znaky a areály, kde plocha vyjadřuje skutečné kvantitativní hodnoty prvků obsahu mapy. Pro odvozování a porovnávání hodnot mezi sebou, musejí být k dispozici srovnávací poměry a k tomu slouží hodnotová (diagramová) měřítka. Toto měřítko slouží ke zjištění velikosti objektu nebo jevu v mapě. Má podobu srovnávacího obrazce, symbolu, diagramu, velikostní stupnice a jiné.

Diagramové měřítko je vztah mezi zobrazením množiny daných kvantit a množinou výšek diagramových znaků. Celková velikost diagramového znaku je vypočítána z konkrétní rovnice a jeho výšky (Voženílek, 2002). Všechny následující diagramy a jejich vzorce jsou čerpány z publikace Kaňoka (1999).

Diagram sloupcový vyjadřuje číselnou hodnotu jevu pomocí své výšky. Šířka sloupce, jeho poloha či tvar zde nehrají roli. Diagram může být reprezentován čarou, válcem, hranolem nebo jiným tvarem.

Rovnice pro výpočet skutečné hodnoty jevu je následující:

$$H = h * v$$

kde:

H …… skutečná hodnota velikosti jevu

h ……. jednotková délková míra použitá v diagramu (např. měřítko, koeficient převodu hodnoty naměřené v diagramu)

v ……. výška sloupce vyjádřená ve stejných jednotkách délky jako *h*

Prakticky může legenda vypadat následovně: 1 mm ~ 100 t

Diagram čtvercový se vytváří podle vztahu $S = a^2$, přičemž *S* vyjadřuje plochu čtverce, *a* je strana čtverce. Plocha čtverce je tedy výsledná skutečná hodnota jevu.

Výpočet plochy čtvercového diagramu:

$$H = h * a^2$$

kde:

H …… skutečná číselná hodnota velikosti jevu

h ……. jednotková plošná míra použitá v diagramu

a ……. velikost strany čtverce

Praktický příklad legendy: 1 cm^2 ~ 500tis. t

Diagram kruhový se velmi jednoduše rýsuje. Identifikace tohoto diagramu na mapě nezávidí na jeho otočení a jeho konstrukční rovnice vypadá takto:

$$r = \sqrt{\frac{H}{h\pi}} \quad \rightarrow \quad H = \pi * h * r^2$$

kde:

r …… poloměr kruhu

H …….. skutečná číselná hodnota velikosti jevu

h …… jednotková plošná míra použitá v diagramu

Diagram polokruhový je obměnou předešlého typu a nejčastěji se používá při srovnání dvou jevů (např. vývoz a dovoz). Poloměr *r* se vypočítá:

$$r = \sqrt{\frac{2H}{h * \pi}}$$

$$H = \frac{\pi * h * r^2}{2}$$

Diagram trojúhelníkový je na konstrukci problematičtější díky optickému zkreslení, které může být vyvoláno neúměrným proporcionálním rozdělením trojúhelníku vůči podílu velikosti rozdělení plochy.

Rovnice pro výpočet plochy trojúhelníku je: *S = (a * v) / 2*, kde je potřeba znát dvě proměnné a to stranu *a* a výšku *v*. V kartografii se nejčastěji využívá rovnostranný a rovnoramenný trojúhelník.

Výpočet pro rovnostranný trojúhelník:

$$S = \frac{a^2 * \sqrt{3}}{4} = 0{,}433a^2$$

$$H = \frac{h * a^2\sqrt{3}}{4} = 0{,}433a^2h$$

$$a = \frac{\sqrt{H}}{0{,}7211h}$$

Pro trojúhelník rovnoramenný:

$$S = \frac{a * v}{2}$$

$$H = \frac{h * a^2 * x}{2}$$

$$a = \sqrt{\frac{2H}{h * x}}$$

kde:

S …… plocha trojúhelníku
a …… strana trojúhelníku
v …… výška trojúhelníku
H …… skutečná číselná hodnota velikosti jevu
h …… jednotková míra plochy použitá v diagramu
x …… konstanta *x = v/a*

Diagramy mnohoúhelníkové, které se používají v mapách, jsou zásadně pravidelné. Do mnohoúhelníkových diagramů jsou řazeny pětiúhelníky, šestiúhelníky, sedmiúhelníky a osmiúhelníky. Ve všech pravidelných mnohoúhelnících stačí pouze jeden parametr k výpočtu jejich plochy.

Vzorce k jednotlivých mnohoúhelníkům:

- Pětiúhelník

$$S = \frac{5 * a^2 * \cot 36°}{4}$$

$$H = \frac{5 * h * a^2 * \cot 36°}{4} = 1{,}66ha^2$$

$$a = \sqrt{\frac{4H}{5 * h * \cot 36°}} = \sqrt{\frac{H}{1{,}66h}}$$

- Šestiúhelník – využívá se nejčastěji kvůli své snadné konstrukci a snadnému měření skutečných hodnot jevu

$$S = \frac{3\sqrt{3} * a^2}{2}$$

$$H = \frac{3\sqrt{3} * h * a^2}{2} = 2{,}6ha^2$$

$$a = \sqrt{\frac{2H}{3\sqrt{3} * h}} = \sqrt{\frac{H}{2{,}6h}}$$

- Sedmiúhelník

$$S = \frac{7 * a^2 * \cot 25°43'}{4}$$

$$H = \frac{7 * h * a^2 * \cot 25°43'}{4} = 3{,}7ha^2$$

$$a = \sqrt{\frac{4H}{7 * h * \cot 25°43'}} = \sqrt{\frac{H}{3{,}7h}}$$

- Osmiúhelník

$$S = 2 * a^2 * \cot 22°30'$$

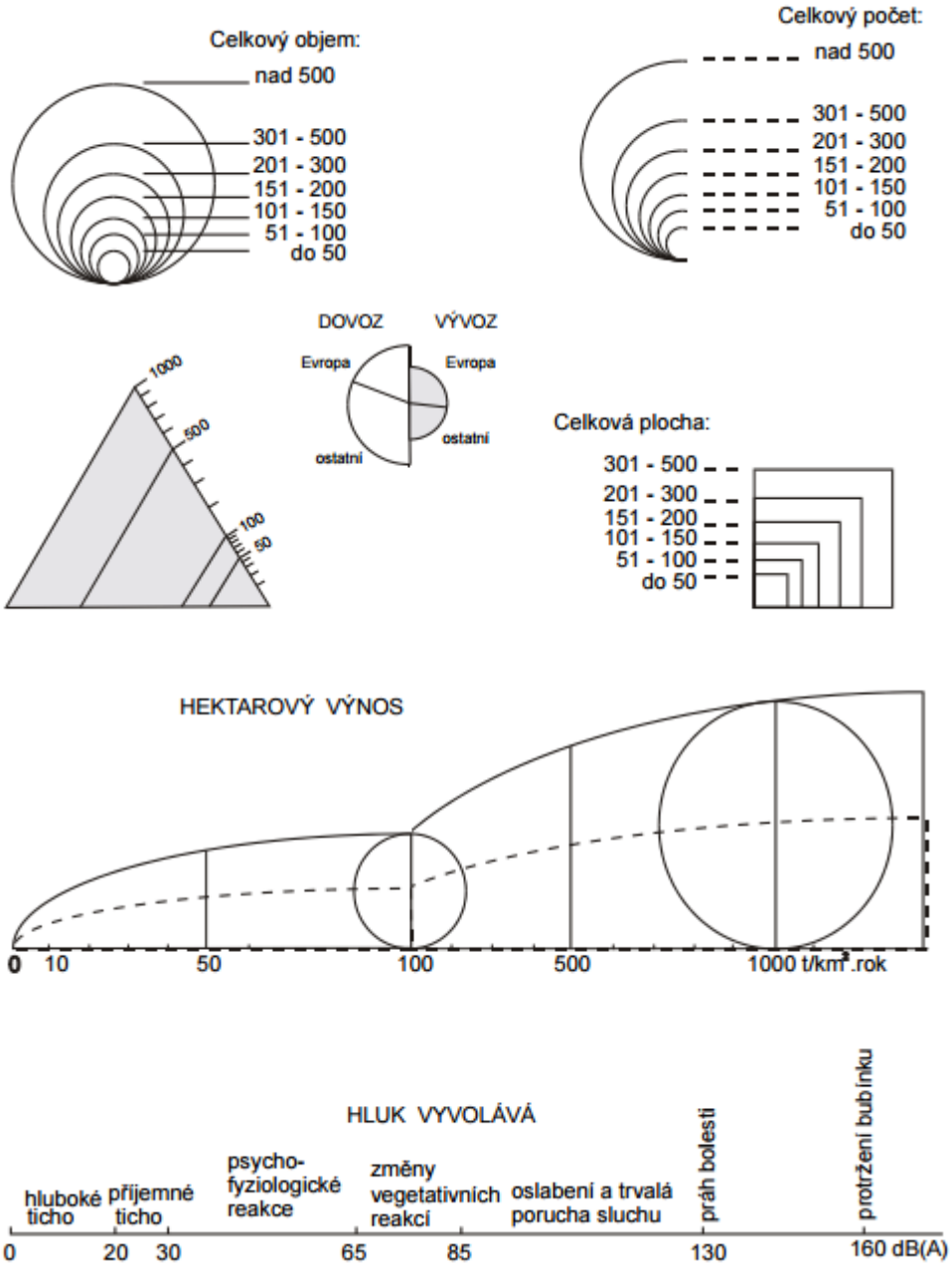
$$H = 2 * h * a^2 * \cot 22°30' = 2{,}83ha^2$$

$$a = \sqrt{\frac{H}{2 * h * \cot 22°30'}} = \sqrt{\frac{H}{2{,}83h}}$$

kde:

S …… plocha pravidelného mnohoúhelníku
a …… strana pravidelného mnohoúhelníku
H …… skutečná číselná hodnota velikosti jevu
h …… jednotková plošná míra

Ukázky hodnotových měřítek podle Voženílka (2002):



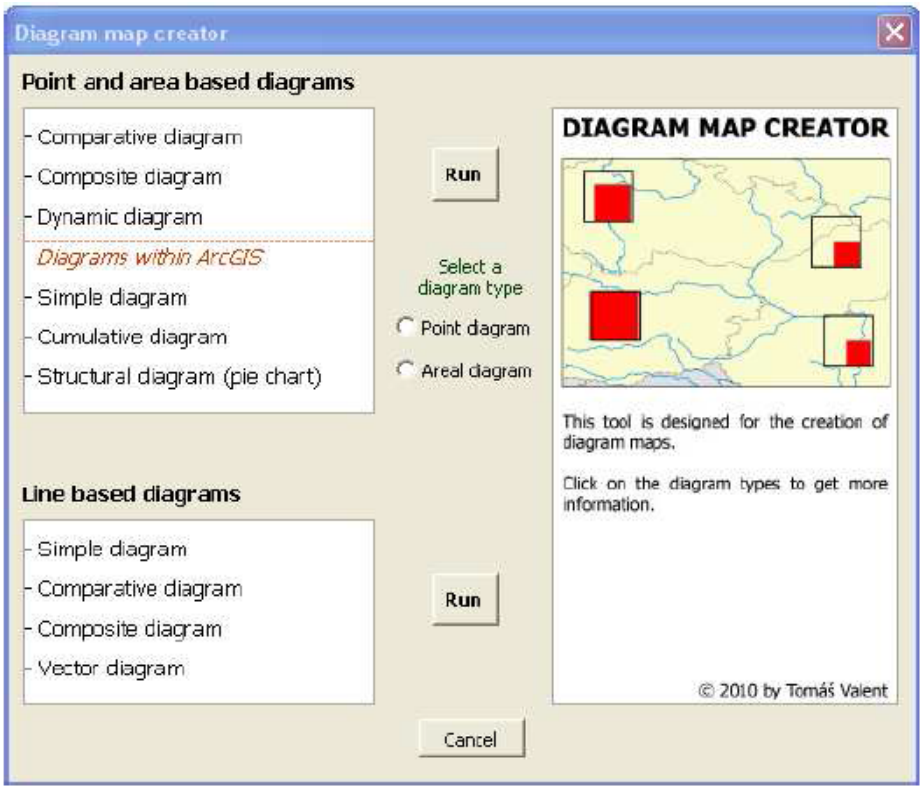
3.2 Nástroje pro tvorbu kartodiagramů a hodnotových měřítek

Nástroje pro tvorbu kartodiagramů a jejich hodnotových měřítek v softwarech lze rozdělit na geoinformatické a statistické. Pro účely této diplomové práce byly vybrány tyto nástroje: ArcGIS for Desktop 10.2, AutoCAD Map 3D 2014, GRASS GIS 7.0.1, Quantum GIS Desktop 2.10.1, Janitor 2.7.34b, Geomedia Viewer 2013 a Kristýna GIS 4.01 a R Studio.

ArcGIS for Desktop 10.2 dopadl z výčtu zkoumaných softwarů na tvorbu kartodiagramů nejlépe. Z těchto programů disponuje největší funkcionalitou potřebnou pro tuto

problematiku. V softwaru ArcGIS for Desktop lze vytvořit mnoho typů kartodiagramů a lze s nimi jednoduše pracovat – měnit jejich parametry, měnit parametry diagramového znaku i pracovat s hodnotovým měřítkem (volba počtu tříd klasifikace, metody klasifikace, ruční úprava intervalů apod.).

Ovšem ani tento robustní nástroj není na tvorbu kartodiagramů nejlepší volbou, jelikož nedokáže sestrojit všechny druhy kartodiagramů, které jsou výše zmíněny a ani výstup hodnotového měřítka v legendě není správně prezentován. Pro větší rozsah tvorby kartodiagramů byla sestrojena aplikace Diagram map creator. Tato nadstavba dokáže automatizovaně vytvořit vybrané druhy kartodiagramů na základě zadání uživatele. Základním rozhraním nadstavby Diagram map creator je hlavní formulář, kde si uživatel vybírá typ kartodiagramu, který chce vytvořit. Na levé straně je seznam s výběrem kartodiagramů a na pravé je informativní okno (Valent, 2010).



Obr. 12 Hlavní formulář aplikace Diagram map creator.

Ale ani tato aplikace se nezaměřuje na automatickou tvorbu hodnotového měřítka v podobě funkce opisující průběh skutečných hodnot jevu objektu.

AutoCAD Map 3D 2014 je základním editačním nástrojem v geografických řešeních (GIS, mapové aplikace) postavených nad technologiemi Autodesk. Slouží primárně ke správě prostorových a popisných údajů v geografických databázích – zejména Oracle a Microsoft SQL. Představuje kompletní řešení pro tvorbu map a topologickou analýzu, urbanismus a plánování, základní mapy závodu, projektování inženýrských sítí, pro GIS a FM (správu majetku), založené na objektové bázi AutoCADu (Cadstudio, 2015).

Tento nástroj není přizpůsoben pro tvorbu kartodiagramů. Pokud bude uživatel chtít v tomto prostředí vytvořit kartodiagram, žádný nástroj zde nenajde. Tvorba je možná pouze ručně pomocí kreslících nástrojů. Tímto způsobem by šel vytvořit pouze jednoduchý kartodiagram bodový nebo liniový. Z toho vyplývá, že automatická tvorba hodnotového měřítka zde také není možná.

GRASS GIS 7.0.1 je open source software využívaný pro správu geoprostorových dat a jejich analýzu, zpracování obrazu, grafiky a mapové výroby, prostorové modelování a vizualizaci. GRASS GIS je v současné době používán v akademických a komerčních prostředích po celém světě, stejně jako v mnoha vládních agenturách a poradenských společností v oblasti životního prostředí. Je zakládajícím členem Open Source Geospatial Foundation (OSGeo) (GRASS GIS, 1998).

Jedna z nabídek v tomto programu se nazývá „Přidat různé vektorové vrstvy (tematickou, graf,…)“, která umožňuje vytvořit Pie nebo Bar chart a nastavit některé jejich parametry, jakožto výběr atributů, velikosti a barevné provedení.

Ačkoliv tento program obsahuje tzv. Cartographic Composer, kde se dá nastavit výsledný layout a přidat různé kompoziční prvky, tvorba nebo editace hodnotového měřítka zde není možná a v legendě se nikterak neprojeví.

Quantum GIS Desktop 2.10.1 je populární open-source GIS software umožňující zejména prohlížení, tvorbu a editaci rastrových a vektorových geodat, zpracování GPS dat a tvorbu mapových výstupů.

Ve vlastnostech vrstvy nabízí záložku Diagramy, kde si lze vybrat typ diagramu a to buď „Koláčový“, „Textový“ nebo „Histogram“. Tato záložka umožňuje například vytvořit kartodiagram jednoduchý bodový, součtový, strukturní a sloupcový. Kartodiagram složený lze vytvořit překrytím dvou jednoduchých kartodiagramů, ale s tím problémem, že se v určitých místech mohou překrývat.

Hodnotové měřítko zobrazené v legendě se v tomto programu nikterak nezobrazí. Diagram není prezentován v podobě jakkoliv seřazených kruhů či aspoň jednoho kruhu, ale nýbrž pouze jako znak pro polygon s příslušnou barvou diagramu a popiskem významu diagramu.

Janitor 2.7.34b je GIS nástroj JANITORu pro získávání, správu, vyhodnocování a publikování dat s územní vazbou. Pracuje s rastrovými a vektorovými datovými formáty, které umožňuje nejen zobrazit, ale i editovat. Umožňuje také práci s atributovými informacemi jednotlivých vrstev. Přestože tato aplikace začínala velmi skromně jako nástroj určený k jednoduché lokalizaci dat nad digitální mapou, stala se dnes plnohodnotným GIS nástrojem, kde přibývá řada sofistikovaných funkcí (dříve vyhrazených jen velkým a většinou drahým aplikacím). Cílem vývojového týmu je nabídnout nástroj, který uspokojí většinu uživatelů GIS při zachování přehledné struktury a malých hardwarových nároků (Janitor, 2005).

Tvorba kartodiagramů v pravém slova smyslu zde také není možná. JanMap zde ovšem ve vlastnostech vrstvy nabízí záložku „Grafy“, která umožňuje vytvořit graf „Koláčový“, „Sloupcový svislý“, „Sloupcový vodorovný“, „Skládaný svislý“ a „Skládaný vodorovný“. S těmito typy grafů lze dále pracovat – vybrat parametry, nastavit barvy, velikost – konstatní/dle sumy položek, rámování, osy, font hodnot apod.

Hodnotové měřítko je prezentováno stejně jako v předešlém případě – pouze znak pro polygon s příslušnou barvou a popiskem.

Geomedia Viewer 2013 je volný GIS prohlížecí nástroj. Tento software umožní: zobrazit GeoTIFF rastrové i vektorové soubory a provést atributové dotazy, prostorové filtrování dotazu, vizualizace dat vytvářením tematických map, měnit všechny parametry symbolů (barvy a vzorování) a zobrazení dat v nativních formátech při spojení s GIS datovými sklady (AMN, 2005).

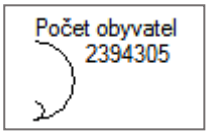
V tomto programu lze pouze aplikovat vizualizační metoda kartogramu. Tvorba kartodiagramu a s ním spojená tvorba hodnotového měřítka zde není k dispozici.

Kristýna GIS 4.01 umožní zobrazovat, zkoumat, dotazovat a analyzovat data prostorově. Klíčovou vlastností Kristýny je snadnost načtení tabelárních dat, jako dBASE® soubory a data z databázových serverů. Použitím Kristýny lze zobrazovat, dotazovat se, sumarizovat a organizovat tato data geograficky (Kristýna GIS, 2015).

S tvorbou kartodiagramů a hodnotového měřítka je na tom velmi podobně jako JanMap. Obsahuje jednu záložku ve vlastnostech vrstvy s názvem „Graf“, kde si lze vybrat mezi typem grafu – „Výsečový“, „Sloupcový“, „Pruhový“ a „Skládaný“. Základní funkce jako nastavení barvy diagramů, velikosti, popisky tento nástroj obsahuje.

Hodnotové měřítko se v legendě vykresluje následovně:

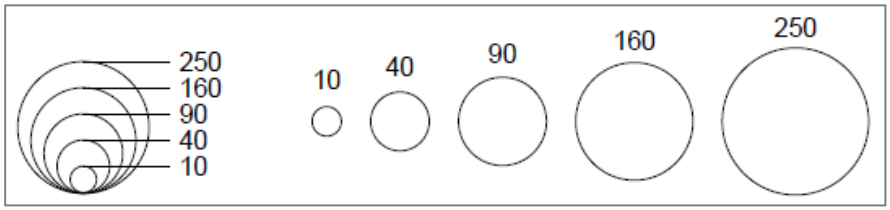
Měnit počet tříd nebo rozsah intervalů ovšem nelze.



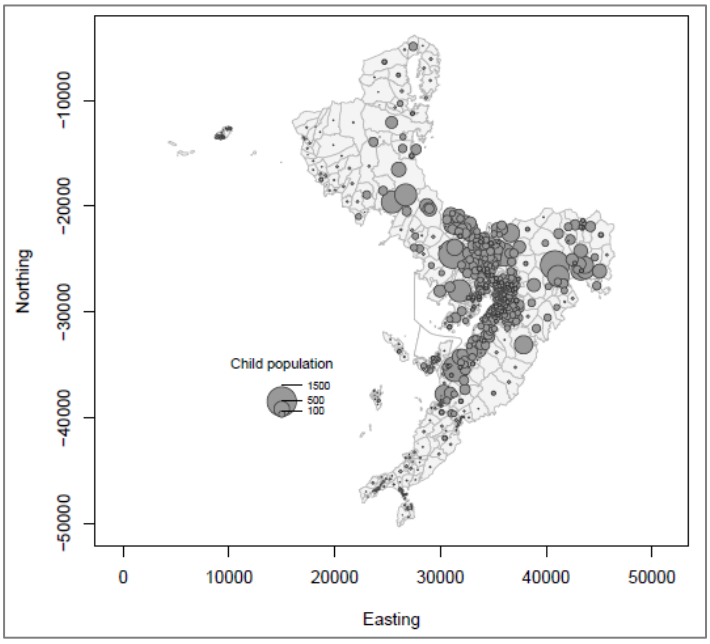
R Studio je prostředím pro statistické výpočty a grafiku. Poskytuje širokou škálu statistických (lineární a nelineární modely, klasické testy, analýza časových řad, klasifikace, klastrování,…) a grafických technik. R je dále snadno rozšiřitelné o další metody. Jednou z nejsilnějších částí je snadnost, s kterou lze vytvářet dobře navrhnuté obrázky a grafy v profesionální kvalitě. Do grafů lze snadno v případě potřeby vkládat matematické symboly a vzorce. Standardní nastavení pro kresbu grafů bylo voleno s maximální pečlivostí, nicméně uživateli je ponechána plná kontrola nad výsledným vzhledem grafu (R projekt, 2003).

R Studio dokáže vykreslit kartodiagramy jednoduché i složené, ale problém nastává při jejich překrývání, které se těžko odstraňuje. Obsahuje širokou škálu symbolů pro tvorbu kartodiagramů – přes kolečko, čtverec, kouli, krychli až po složitější tvary jako např. sloupcové grafy a piktografické symboly.

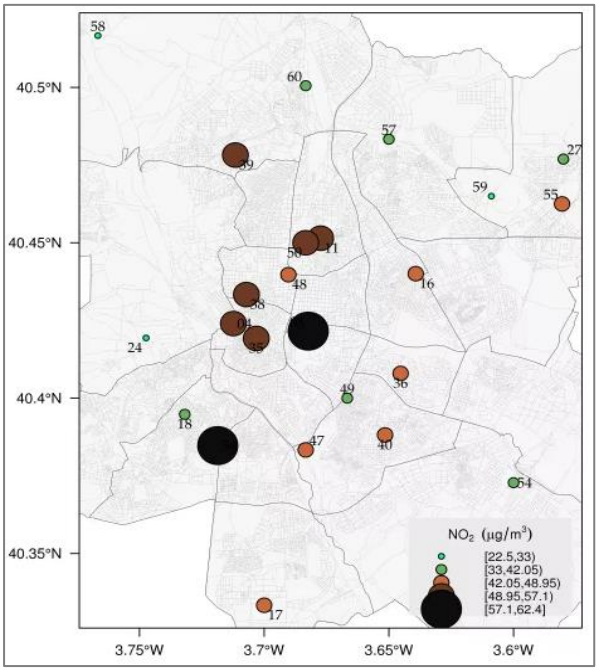
Při programování kartodiagramů si lze vybrat, podle kterého vzorce mají být velikosti symbolů spočítány anebo jaký druh legendy se má použít, jestli hodnotové měřítko s funkční stupnicí nebo intervalovou. Dále se dají nastavit parametry jako počet symbolů v legendě, jaké uspořádání symbolů v legendě, maximální velikost symbolu, barevná škála, …



Obr. 13 Typy uspořádání legendy pro kartodiagramy s funkční stupnicí v R Studiu.



Obr. 14 Ukázka mapy vytvořené v R Studiu pro kartodiagram s funkční stupnicí (Tanimura, Kuroiwa a Mizota, 2006).



Obr. 15 Ukázka mapy vytvořené v R Studiu pro kartodiagram s intervalovou stupnicí (Spatial Data visualization with R, 2012).

Tabulka 1a: Seznam testovaných softwarů a tvorba kartodiagramů plošných a bodových ve vybraných softwarech.

	Jednoduchý	Složený	Součtový	Strukturní	Sloupcový
ArcGIS for Desktop	ano	ano, jen ručně překrytím dvou jednoduchých	ano	ano	ano
Nadstavba Diagram map creator pro ArcGIS for Desktop	ano	ano	ano	ano	ne
AutoCAD	ano, jen bodový	ne	ne	ne	ne
Quantum GIS	ano	ano, překrytím dvou jednoduchých	ano	ano	ano
GRASS GIS	ne	ne	ne	ano	ano
Janitor	ano	ne	ano	ano	ano, i záporné hodnoty
Geomedia Viewer	ne	ne	ne	ne	ne
Kristýna	ano	ne	ano	ano	ano
R Studio	ano	ano	ano	ano	ano

Tabulka 1b: Seznam testovaných softwarů a tvorba kartodiagramů plošných a bodových ve vybraných softwarech.

	Srovnávací	Dynamický	Anamorfózní	Typogram
ArcGIS for Desktop	ne	ne	ne	ne
Nadstavba Diagram map creator pro ArcGIS for Desktop	ano	ano	ne	ne
AutoCAD	ne	ne	ne	ne
Quantum GIS	ne	ne	ne	ne
GRASS GIS	ne	ne	ne	ne
Janitor	ne	ne	ne	ne
Geomedia Viewer	ne	ne	ne	ne
Kristýna	ne	ne	ne	ne
R Studio	ne	ne	ne	ne

Tabulka 2: Seznam testovaných softwarů a tvorba kartodiagramů liniových ve vybraných softwarech.

	Jednoduchý	Vektorový dosahový jednoduchý	Vektorový dosahový součtový	Stuhový součtový	Stuhový strukturní
ArcGIS for Desktop	ano	ne	ne	ano, jen ručně obtížně	ano, jen ručně obtížně
Nadstavba Diagram map creator pro ArcGIS for Desktop	ano	ano	ne	ne	ne
AutoCAD	ano	ne	ne	ne	ne
QuantunGIS	ne	ne	ne	ne	ne
GRASS GIS	ne	ne	ne	ne	ne
Janitor	ne	ne	ne	ne	ne
Geomedia Viewer	ne	ne	ne	ne	ne
Kristýna	ne	ne	ne	ne	ne
R Studio	ne	ne	ne	ne	ne

Tabulka 3: Nastavení a vlastnosti intervalové stupnice ve vybraných programech.

	meze intervalů stupnice ručně	z konstantních intervalů hodnot	metoda přirozených zlomů	metoda kvantilu	stupnice intervalů podle geometrické posloupnosti	metoda podle standardní odchylky	nespojitá intervalová stupnice
ArcGIS	ano	ano	ano	ano	ne	ano	ano
AutoCAD	ano	ano	ano	ano	ne	ano	ano
QuantumGIS	ano	ano	ne	ano	ne	ne	ano
GRASS GIS	ano	ano	ano	ano	ano	ne	ne
Janitor	ne	ano	ne	ano	ne	ne	ne
Geomedia Viewer	ano	ano	ne	ano	ne	ano	ne
Kristýna	ano	ano	ano	ano	ne	ano	ne
R Studio	ano	ano	ano	ano	ne	ne	ano

Tabulka 4: Nastavení a vlastnosti funkční stupnice ve vybraných programech.

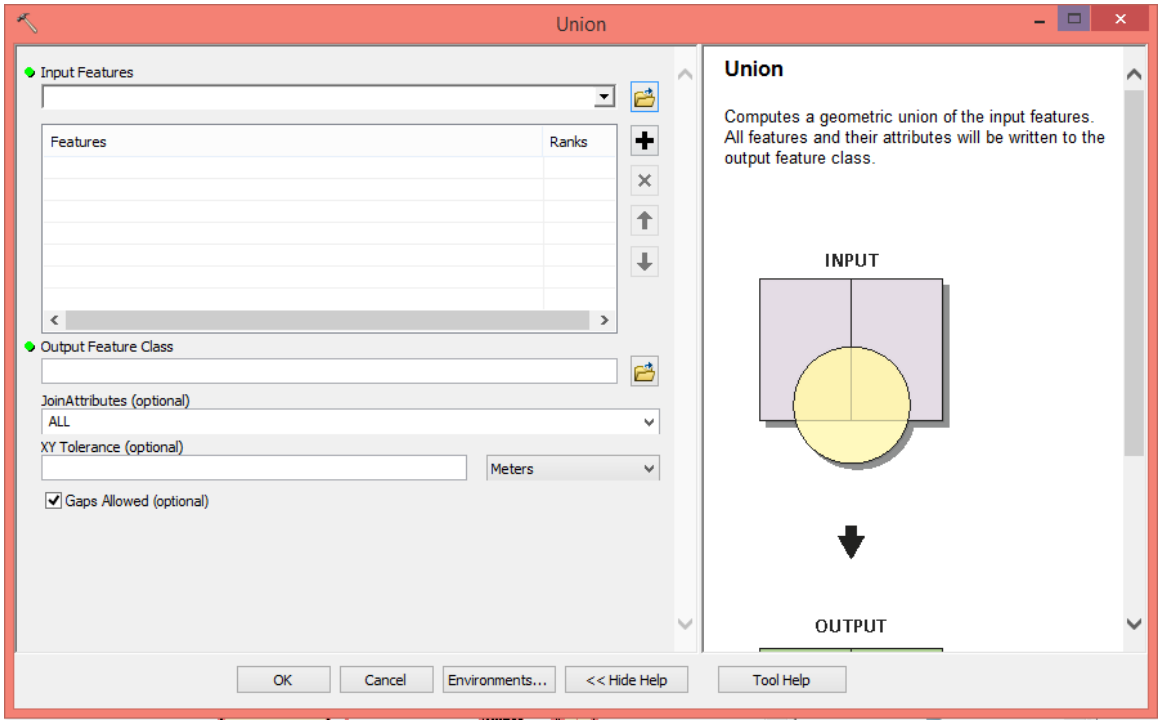
	spojitá stupnice pro určení parametrů znaku	nespojitá stupnice pro určení parametrů znaku se skokovou změnou	nespojitá stupnice pro určení parametrů znaku se změnou funkce
ArcGIS	ne	ne	ne
AutoCAD	ano	ne	ne
QuantumGIS	ne	ne	ne
GRASS GIS	ne	ne	ne
Janitor	ne	ne	ne
Geomedia Viewer	ne	ne	ne
Kristýna	ne	ne	ne
R Studio	ano	ano	ne

Tabulka 5: Možnost tvorby hodnotového měřítka ve vybraných programech.

	Sloupcový	Plynule stupňovaný kruhový	Intervalově stupňovaný kruhový	Úplnost hodnotových měřítek
ArcGIS	ne	ne	ne	ne
AutoCAD	ne	ne	ne	ne
QuantumGIS	ne	ne	ne	ne
GRASS GIS	ne	ne	ne	ne
Janitor	ne	ne	ne	ne
Geomedia Viewer	ne	ne	ne	ne
Kristýna	ne	ano	ne	ne
R Studio	ano	ano	ano	ne

4 AUTOMATIZACE TVORBY HODNOTOVÝCH MĚŘÍTEK KARTODIAGRAMŮ - PYTHON TOOLBOX

Jelikož výsledkem této diplomové práce má být vytvořená nadstavba pro program ArcGIS for Desktop (dále jen ArcGIS), prvním prostředím pro vytváření této nadstavby byl automaticky Python Toolbox (dále jen Toolbox). Dokumentace k vytváření vlastního Toolboxu je v nápovědách na webových stránkách Esri.com velmi bohatá a snadno pochopitelná s širokou škálou různých příkladů a šablon. Pozdější implementace do programu ArcGIS je snadná a většina uživatelů už s Toolboxy umí pracovat, tudíž to pro ně není žádná nová věc nebo překážka.



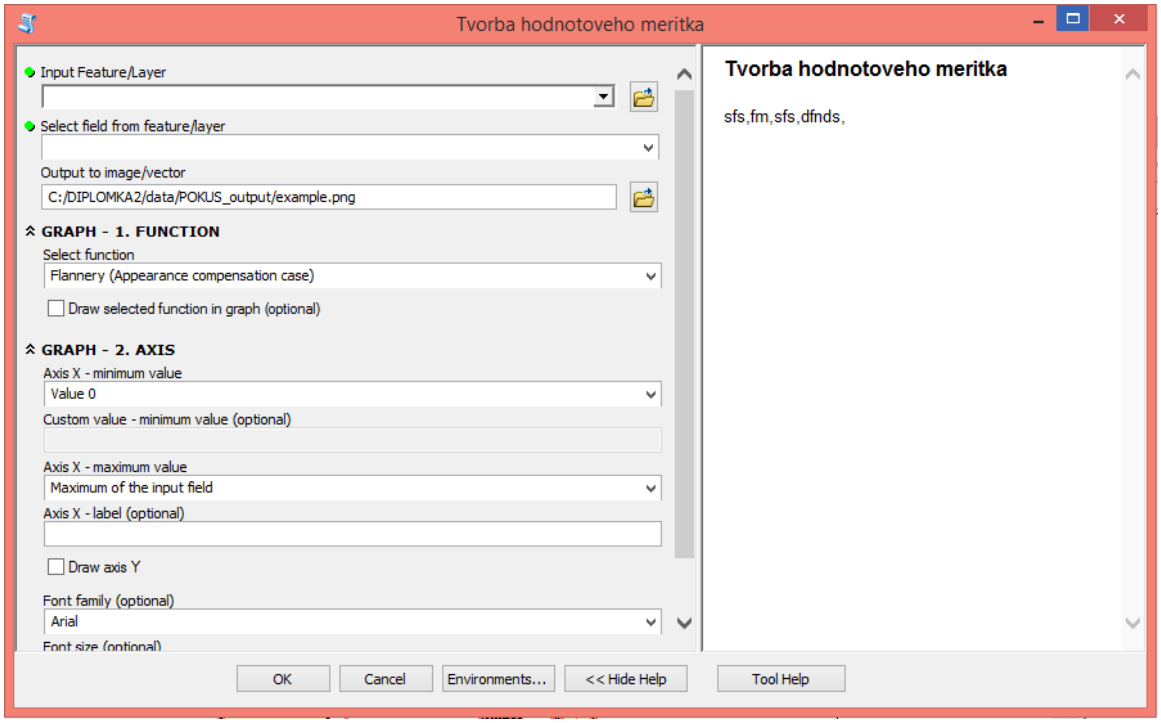
Obr. 16 Ukázka Python Toolboxu v prostředí ArcGIS for Desktop.

Rozložení komponent Toolboxu je zachované stejně u všech Toolboxů. V levé části se zadávají různé parametry – vložení vstupních vrstev, definování výstupních vrstev a vyplnění dalších parametrů potřebných k jednotlivým operacím. V pravé části se nachází vysvětlivky jednotlivých parametrů.

Uspořádání komponent ve vlastním programu bylo z větší částí stejné. Změnu doznala pravá část, která byla plánována k vykreslování náhledu výsledného hodnotového měřítka, které by se automaticky po určité době nebo po zmáčknutí určitého tlačítka obnovovalo podle změn v zadaných parametrech, aby měl uživatel stále pod kontrolou výsledný vzhled měřítka, mohl si ho v době vytváření interaktivně měnit a pracovat s ním do té doby, než by byl spokojený a až poté vyexportovat do výsledného formátu a použít v mapě.

Tvorba vlastního programu v prostředí Python Toolbox trvala asi 3 měsíce. Po této době nastal problém právě ve výše zmíněném prostoru pro náhled výsledného měřítka. Tuto funkcionalitu Toolbox nepodporoval. Další variantou, jak tuto funkcionalitu přidat do vlastního Toolboxu, bylo vytvoření nového paralelního okna určeného speciálně jen pro náhled hodnotového měřítka. Tento způsob opět nebyl úspěšný, jelikož se tato dvě běžící okna navzájem rušila. V praxi to znamená, že pokud se pracovalo s Toolboxem, nové paralelní okno se zacyklilo a spadlo nebo naopak.

Vytváření programu s funkcionalitou neustále se obnovujícího náhledu výsledného hodnotového měřítka bylo pro automatickou tvorbu hodnotových měřítek klíčové a nepostradatelné, tudíž nezbývalo nic jiného, než možnost vytváření vlastního programu v Python Toolboxu opustit a hledat nové možnosti, jak tento cíl splnit.



Obr. 17 Ukázka rozpracovaného vlastního řešení pro automatickou tvorbu hodnotových měřítek kartodiagramů v prostředí Python Toolbox.

Ukázka kódu z rozpracovaného vlastního programu v Python Toolboxu (definování třídy objektu, definování vstupních a výstupních parametrů – lze porovnat s obrázkem výše):

```
class TvorbaHodnotovehoMeritka(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Tvorba hodnotoveho meritka"
        self.description = "sfs,fm,sfs,dfnds,"
```

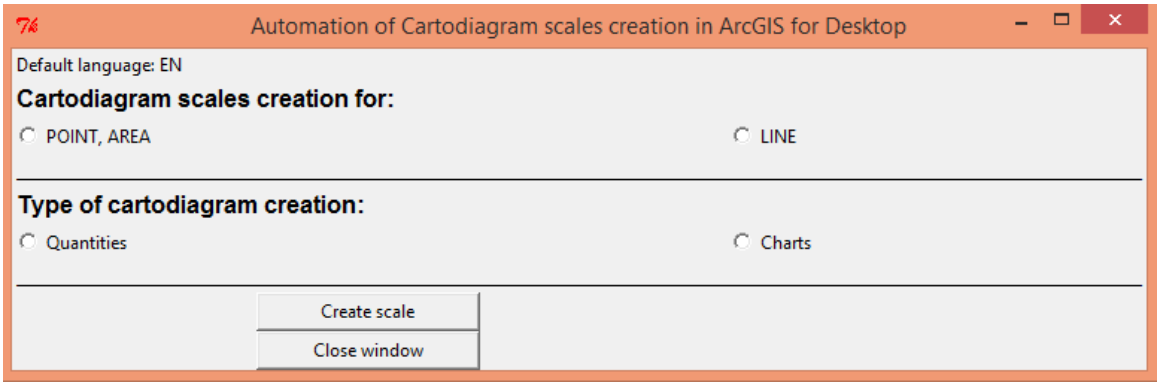
```
def getParameterInfo(self):
    """Define parameter definitions"""

    # parameter INPUT FEATURE
    param0 = arcpy.Parameter(
      displayName="Input Feature/Layer",
      name="in_features",
      datatype="GPFeatureLayer",
      parameterType="Required",
      direction="Input"
      )

    # parameter FIELD FROM INPUT FEATURE
    param1 = arcpy.Parameter(
      displayName="Select field from feature/layer",
      name="field_name",
      datatype="Field",
      parameterType="Required",
      direction="Input"
      )

    param1.parameterDependencies = [param0.name]

    # parameter OUTPUT
    param2 = arcpy.Parameter(
      displayName="Output to image/vector",
      name="out_features",
      datatype="File",
      parameterType="Required",
      direction="Output"
      )

 param2.value="C:/data/POKUS_output/example.png"
```

5 AUTOMATIZACE TVORBY HODNOTOVÝCH MĚŘÍTEK KARTODIAGRAMŮ – PROSTŘEDÍ TKINTER

Po neúspěšné práci v Python Toolboxu, která byla popsána v kapitole 4, bylo potřeba programovat v jiném prostředí. Tvorba hodnotového měřítka tak byla realizována v prostředí **Tkinter**. Jedná se o modul, který má přístup ke knihovně Tk v programovém jazyku Python a umožnuje uživateli vytvářet grafické uživatelské rozhraní (dále jen GUI). Výstupem programování je jedno hlavní okno, které se skládá z několika ovládacích prvků a u konkrétních druhů diagramů i vlastní plochy grafu, pomocí nichž cílový uživatel podstoupí celý proces vytvoření hodnotového měřítka. V následujících podkapitolách je poskytnut návod, jak pracovat s nástrojem a jsou zmíněny podmínky, které je potřeba dodržet pro dosažení výsledku, tj. vygenerování hodnotového měřítka. V některých částech je také vysvětleno, jaké knihovny, moduly či kódy byly v práci použity.

5.1 Spuštění nástroje a implementace knihoven

Ke spuštění nástroje slouží python soubor Automation of Cartodiagram scales creation in ArcGIS for Desktop s koncovkou *.py. Aby došlo k úspěšnému spuštění, musí mít uživatel na svém počítači nainstalovaný produkt **ArcGIS for Desktop**, který při instalaci automaticky nainstaluje do počítače python s potřebnými knihovnami a zároveň s sebou nese i modul arcpy, který je ve zdrojovém kódu používán. Nástroj byl vyzkoušen na Windows 7 a 8.1 a produktu ArcGIS for Desktop verze 10.2 a na zkušební verzi 10.4. Do složky Site-packages pro knihovny jazyka Python (složka Lib) nakopíruje knihovnu PIL, která pracuje s obrázky a která je uložena ve stejnojmenné složce (Site-packages) v balíčku s celým programem. Žádná jiná nastavení nejsou potřeba, Tkinter je multiplatformní.

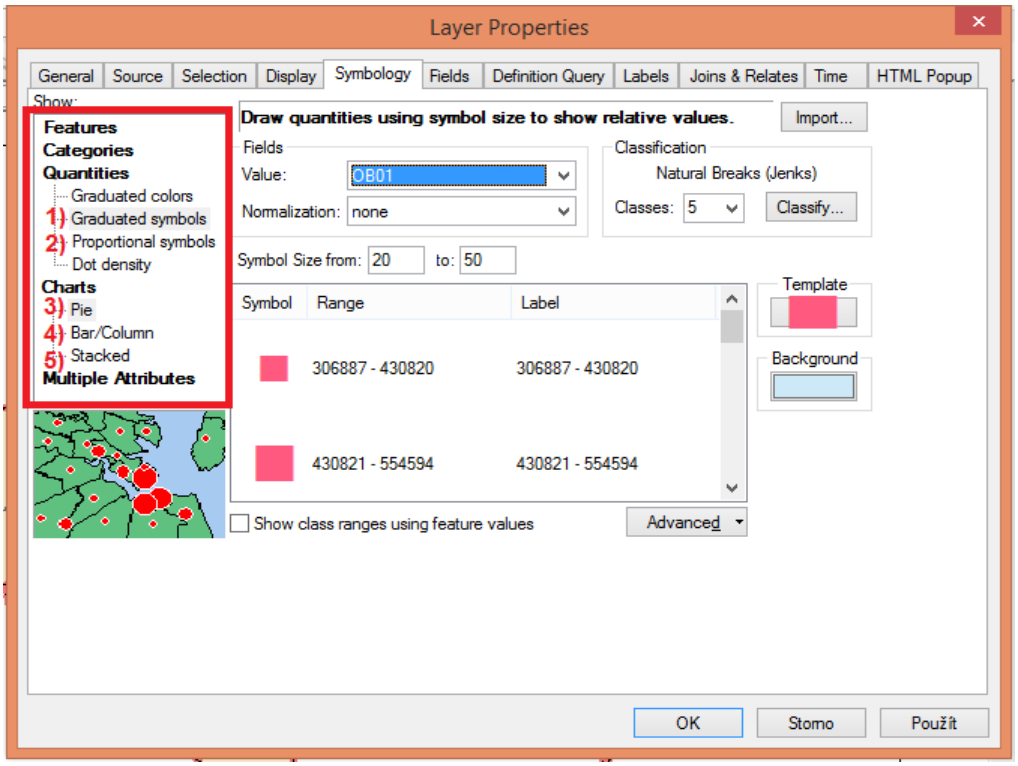


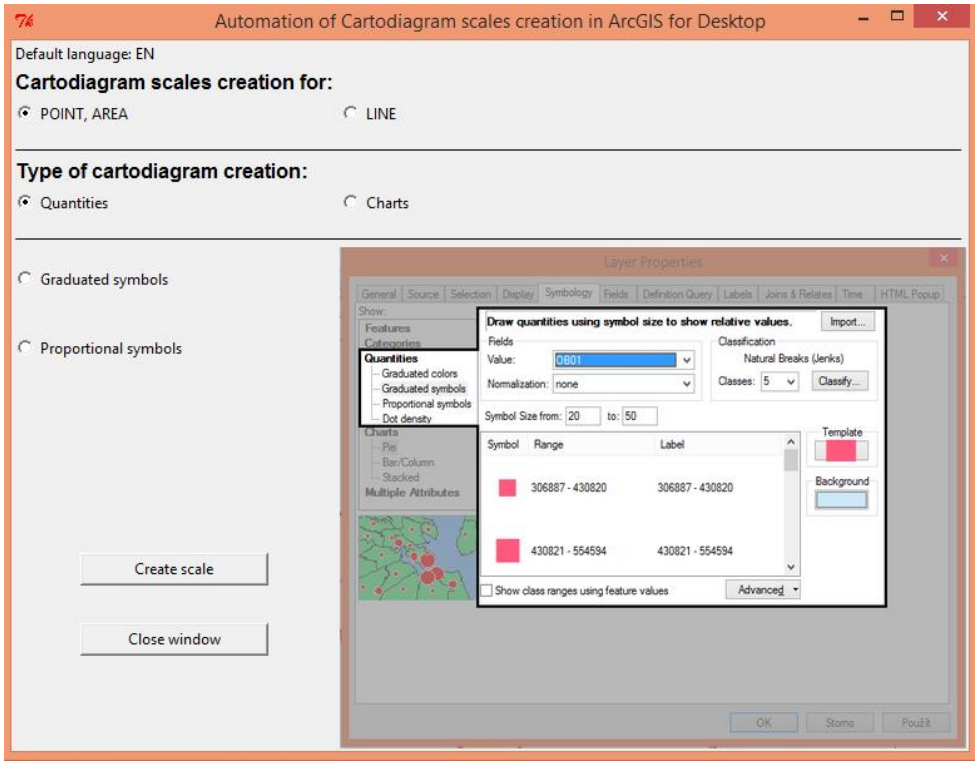
Obr. 18 Ukázka hlavního okna vlastního programu v prostředí Tkinter – tzv. „rozcestník“ pro výběr tvorby konkrétního typu hodnotového měřítka.

Rozhraní po spuštění je velice jednoduché a začíná se větvit až v době, kdy uživatel prochází programem a vybírá si z možností výběrů. První část se týká výběru vrstvy,

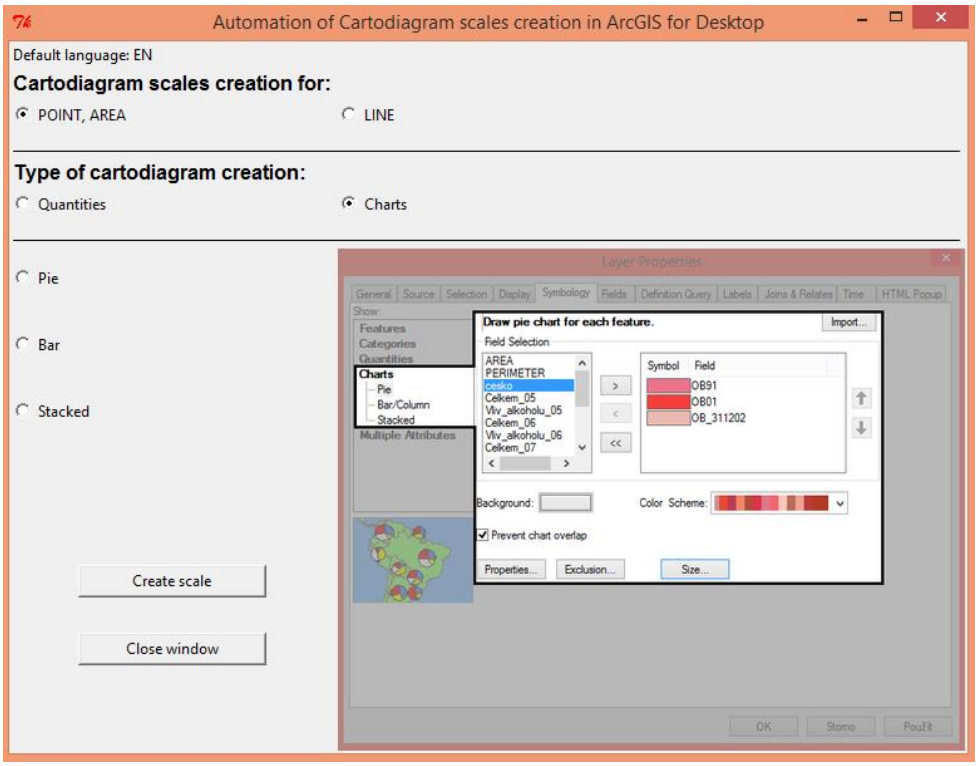
zda bude uživatel vytvářet hodnotové měřítko pro kartodiagram bodový, plošný nebo liniový. Při druhém rozhodování (zda Quantities nebo Charts) se uživateli nabídnou konkrétní typy kartodiagramů podle dané skupiny.

Srovnání hlavní nabídky programu ArcGIS for Desktop s nabídkou vlastního programu pro automatickou tvorbu hodnotových měřítek:





Obr. 19 a 20 Nabídky pro tvorbu hodnotových měřítek kartodiagramů v prostředí ArcGIS a ve vlastním navrženém programu (skupina Quantities).



Obr. 21 Nabídky pro tvorbu hodnotových měřítek kartodiagramů ve vlastním navrženém programu (skupina Charts).

Jak již bylo řečeno, ve vlastním programu se nejprve vybírá, zda se bude generovat hodnotové měřítko pro bod, plochu nebo linii. ArcGIS tuto nabídku změní automaticky podle toho, která vrstva je označená a pro kterou se bude nastavovat symbologie.

Dále se vybere typ, pro který se bude hodnotové měřítko vytvářet a následně zmáčknutím tlačítka *Create scale* se po chvíli načte podprogram s nabídkou parametrů potřebných k tvorbě konkrétního kartodiagramu.

5.2 Graduated symbols pro bodovou a plošnou vrstvu

První byla vybrána nabídka Graduated symbols pro bodovou a plošnou vrstvu. Touto cestou lze v produktu ArcGIS vytvořit kartodiagram jednoduchý, složený, srovnávací nebo dynamický s intervalovou stupnicí a určitý způsob jeho hodnotového měřítka.

Parametry pro tvorbu těchto kartodiagramů v prostředí ArcGIS se skládají z:

A) volby dat, která mají být vizualizována

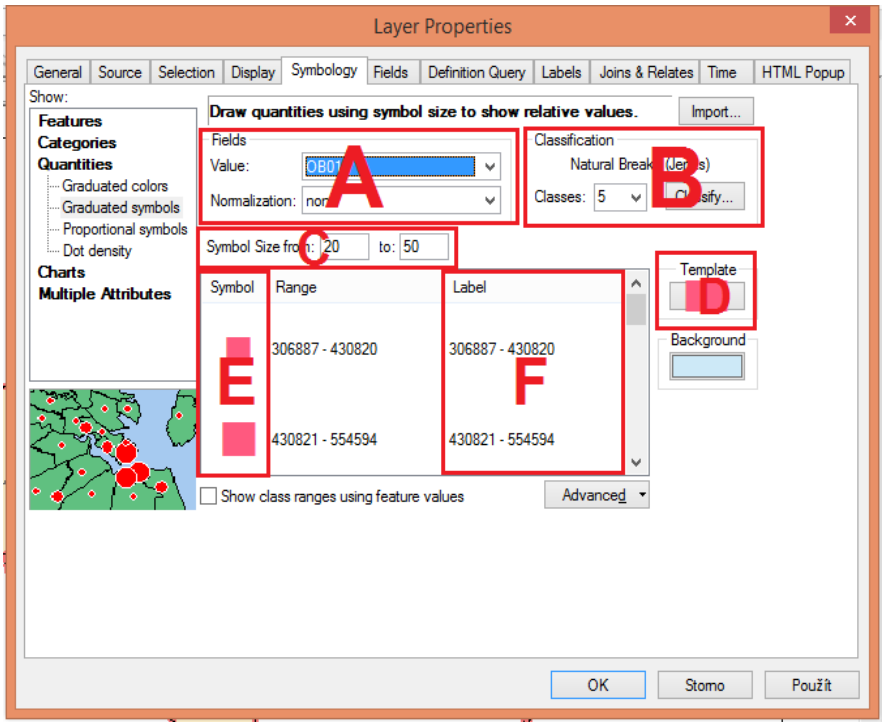
B) počtu tříd, do kterých mají být data klasifikována

C) definování minimální a maximální velikosti symbolů

D) výběru tvaru a barevného provedení symbolu

E) možnosti manuálně editovat každý symbol zvlášť

F) editace popisu každého symbolu neboli definování intervalů



Obr. 22 Definování parametrů pro tvorbu výše zmíněných kartodiagramů v prostředí ArcGIS.

Definice parametrů ve vlastním programu Tkinter v porovnání s označením parametrů v prostředí ArcGIS, bližší vysvětlení parametrů bude popsáno níže v kapitole 5.2.2 Vysvětlení parametrů:

B) počet tříd, do kterých mají být data klasifikována (možnosti: *od 1 až po 10 symbolů*)

C) definování minimální a maximální velikosti symbolů (pokud je zvolena klasifikační metoda – *Automaticky)*

D) volba prvních 5ti základních tvarů v nabídce od Esri (možnosti: *Circle 1, Square 1, Triangle 1, Pentagon 1, Hexagon 1*)

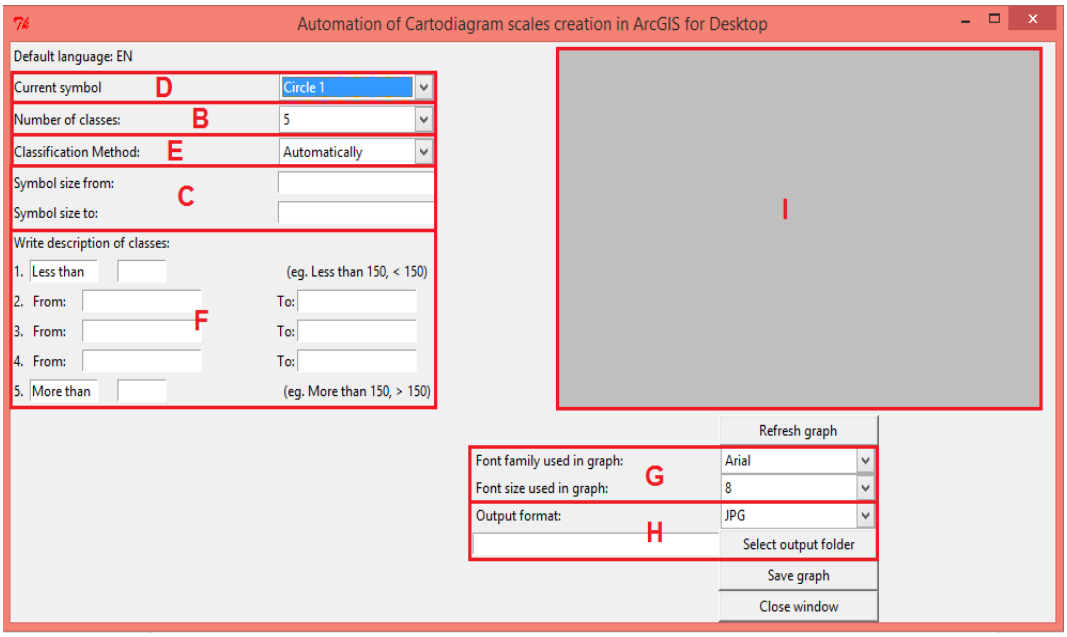
E) klasifikační metoda pro určení velikosti symbolů (*Automaticky* nebo *Manuálně*)

F) editace popisu každého symbolu neboli definování intervalů

G) volba velikosti a druhu fontu použitém v popisu

H) výběr výstupního formátu a místa uložení hodnotového měřítka

I) plocha pro vykreslení hodnotového měřítka



Obr. 23 Odpovídající nabídka ve vlastním navrženém programu k nabídce v prostředí ArcGIS.

Jako první bylo do zdrojového kódu vlastního programu implementováno to nejdůležitější, tím byl samotný modul Tkinter, pomocí něj se dá deklarovat řada widgetů (např. button, menu, combobox, label apod.). Pro zavedení sloužil příkaz *from Tkinter import* *. Pro grafickou část nástroje byla volána knihovna **Matplotlib**, která dokáže vizualizovat data, vytvářet grafy a provést kvalitní grafický výstup v různých formátech. K tvorbě grafů byla zapotřebí ještě knihovna **Math**, která byla využita ve vzorcích při vykreslení různých geometrických tvarů. Nakonec byla volána knihovna **Os**, která obsahuje různé systémové příkazy – ve vlastním programu byla využita pro ukládání výsledných hodnotových měřítek do různých formátů a složek v systému.

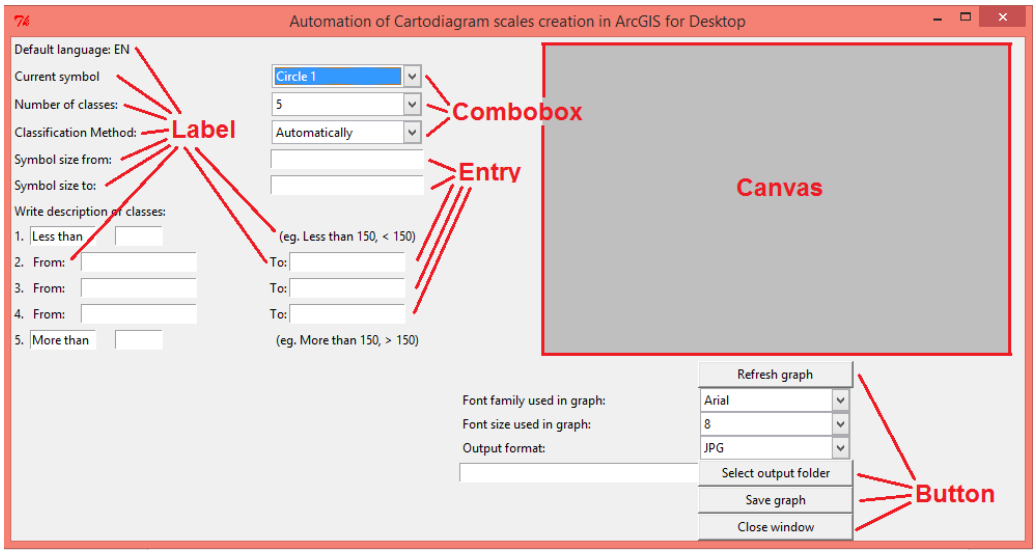
5.2.1 Design programu

Prvním úkolem při tvorbě vlastního programu je vždy ujasnění si a následné nadefinování všech potřebných parametrů, jejich rozmístění a všech ostatních pomocných widgetů, jako například tlačítek, labelů a nabídek, jak dané parametry vybírat a prezentovat, aby byla uživatelem správně pochopena a použita.

Z tohoto důvodu byla záměrně vynechána nabídka výběru dat (načítání souborů z databáze nebo shapefilů), pro která se bude hodnotové měřítko kartodiagramu vytvářet, protože v tomto případě to není nutné a celý program by se tím jen zpomalil. Pro popis dat lze využít část věnovanou popisu intervalů, který se v softwaru ArcGIS ve většině případů stejně upravoval ručně, jelikož generoval nedekadické hodnoty.

V tomto programu byly použity widgety – Label, Combobox, Entry, Button a Canvas (Figure). Widget **Label**, jak už sám název napovídá, byl použit pro textové pojmenování jednotlivých parametrů. Lze jej využít i například pro různé vysvětlivky k jednotlivým

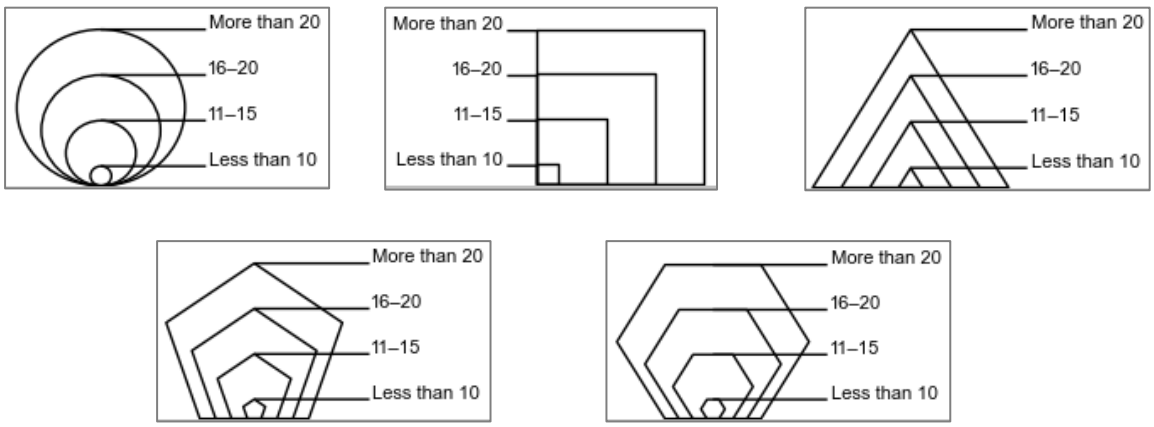
parametrům, pokud je to nutné. Do dvojice k widgetu Label byl vybrán widget **Combobox**, který slouží k vytvoření tzv. menu nebo otevírající se nabídky s možností výběru jedné položky uložené v ní. I widget **Entry** nejčastěji tvoří dvojici k Labelu. Entry je volné vstupní pole, do kterého si uživatel může napsat, co chce. Samozřejmě pro využití v programu je od programátora vždy omezena nabídka datových typů, které uživatel může použít a vepsat do tohoto widgetu, aby nevznikly problémy v následujícím použitím ve výpočtu apod. **Button** neboli tlačítko se používá pro přidělení určité funkce nebo metody, která se spustí, když uživatel klikne na toto tlačítko. Ve vlastním programu bylo využito pro obnovení grafu, vybrání místa uložení výsledného hodnotového měřítka, následný příkaz pro uložení výsledku a zavření celého okna. Posledním widgetem a tím nejdůležitějším byl **Canvas** (plátno), který byl použit na kreslení grafů a rysů.



Obr. 24 Grafické znázornění použitých widgetů ve vlastním programu.

5.2.2 Vysvětlení parametrů

První parametr byl pojmenován *Current symbol* a dává uživateli možnost výběru tvaru symbolu, pro který se má hodnotové měřítko kartodiagramu vytvořit. K dispozici je prvních 5 základních tvarů od společnosti Esri a to *Circle 1*, *Square 1*, *Triangle 1*, *Pentagon 1* a *Hexagon 1*.



Obr. 25 Ukázky 5ti základních tvarů symbolů z vlastního programu.

Další parametr je převzat z prostředí ArcGIS pro snadnější orientaci u uživatele a to *Number of classes*, který znamená počet intervalů a k výběru je hodnota 1 – 10 intervalů.

Parametr *Classification Method* má podobný význam jako v prostředí ArcGIS, kde si uživatel vybírá, jak se mají hodnoty v datech rozklasifikovat a kde mají být definovány hranice jednotlivých intervalů. Ve vlastním programu tento parametr nabízí možnosti:

- *Automatically* – tato možnost znamená, že se velikosti jednotlivých symbolů mají vypočítat lineárně podle počtu symbolů a hodnot minimální a maximální velikosti.
- *Manually* - při zvolení této možnosti se do programu přidá automaticky nový sloupec pro zadání velikosti symbolu pro každý interval zvlášť, kde si uživatel sám ručně stanovuje velikost každého symbolu v jednotce Point.

Dalšími parametry vycházejícími z předchozí nabídky jsou *Symbol size from* a *Symbol size to*. Zpřístupnění nebo zviditelnění těchto parametrů závisí na parametru předchozím a to, zda uživatel zvolí klasifikační metodu automatickou nebo manuální. Defaultně je program nastaven tak, aby při spuštění byla vybrána možnost automatické klasifikační metody, tudíž uživatel následně vidí i tyto dva parametry. Uživatel zde zadá minimální a maximální hodnotu velikosti symbolů v jednotce Point a z těchto hodnot se pak dopočítají podle počtu tříd / intervalů zbývající velikosti symbolů mezi nimi. Pokud uživatel zvolí nabídku manuální klasifikační metody, tyto dva parametry ztrácí smysl a přejdou do tzv. stavu – disabled neboli znepřístupněné pro jakýkoliv zápis od uživatele.

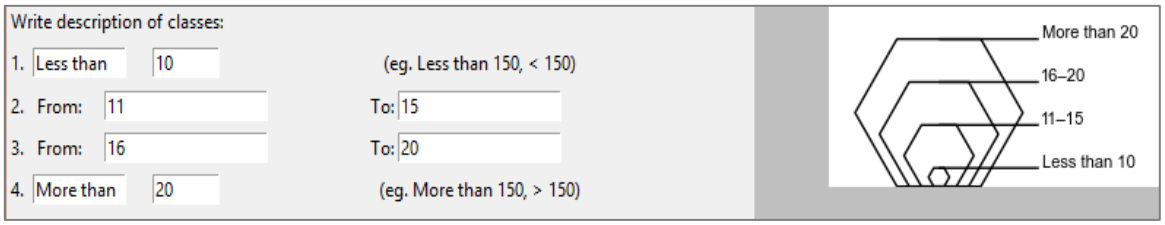
Následuje prostor pro vyplnění popisu každého intervalu – parametr *Description of classes*. Počet widgetů se vygeneruje podle počtu tříd vybraných na začátku programu, kdy první a poslední popis intervalu má jeden styl popisu a všechny ostatní mezi nimi druhý. Tím je myšleno následující:

Styl popisu intervalu pro první a poslední třídu nebo kategorii obsahuje dvě vstupní pole widgetu Entry, kdy do prvního vstupního pole je předdefinován text – „Less than“ nebo „More than“ podle toho, jestli se to týká první nebo poslední kategorie a druhé vstupní pole je prázdné a přichystané pro hodnotu zadanou uživatelem. Text v prvním vstupním poli je sice předdefinován, ale slouží spíše jako pomůcka pro uživatele. Ten, pokud chce, tento text může smazat a napsat si vlastní hodnoty nebo znak pro menší než a větší než určitá hodnota jevu.

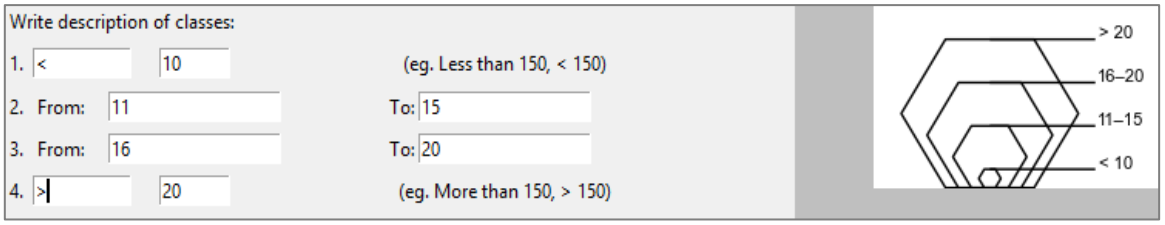
Na rozdíl od výše popsaného stylu popisu je mezi těmito krajními intervaly naprogramován druhý styl, který se skládá ze striktního definování hodnot od–do, který je v legendě vykreslen sloučením zadaných hodnot do jednoho intervalu.

Následující ukázky možnosti popisu tříd intervalů kartodiagramu v zadávací a vykreslovací části vlastního programu:

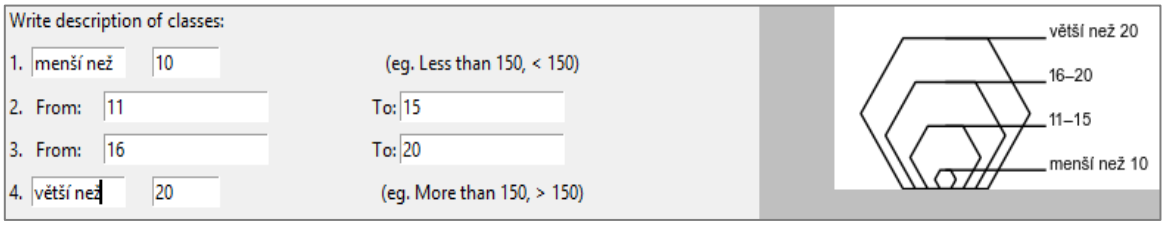
1) Ponechání předdefinovaného textu ve vstupních polích první a poslední kategorie intervalu



2) Vymazání předefinovaného textu a použití matematických znamének - <, >



3) Přestože je program napsán v anglickém jazyce, je možnost vkládat do vstupního pole i česká slova a spojení.



Pro zadávání správných dat do popisu jednotlivých intervalů bylo vytvořeno pár podmínek a varovných upozornění pro uživatele, pokud by se dopustil při vyplňování popisu nějakých chyb.

Popisem intervalů je zakončena levá část programu a v pravé části se nachází dominantní okno nebo prostor sloužící pro vykreslení hodnotového měřítka kartodiagramů. Nastavení pro vykreslení různých grafů a rysů umožňuje knihovna Matplotlib nespočetně mnoho – přes nastavení os, hlavního a vedlejšího dělení, vlastností grafu samotného, popisu os atd. Pro případ této diplomové práce ovšem není nic z vlastností grafů potřeba a tudíž bylo vše skryto.

Předposlední parametry se týkají nastavení písma použitého v popisu intervalů. Jedná se konkrétně o parametry *Font family used in graph* a *Font size used in graph*. Z fontů písma si uživatel může vybrat mezi – *Arial, Calibri, Verdana a Times New Roman* a z velikostí má nabídku – *8, 9, 10, 11, 12 a 15*.

Poslední dva parametry také patří k sobě a to *Output format* a tlačítko *Select output folder* s přiděleným vstupním polem. Combobox pro Label *Output format* obsahuje možnosti uložení výstupního hodnotového měřítka kartodiagramu do rastrových formátů - *.jpg, *.png, *.tiff, ale také do vektorového formátu a tím je formát *.svg. Rozlišení výstupu je nastaveno na 300dpi.

Tlačítko *Select output format* otevře dialogové okno do systému uživatele a vybídne ho, ať zvolí místo uložení jeho výstupu. Po kliknutí na OK se cesta k výslednému měřítku vepíše do vstupního pole widgetu Entry, aby si mohl uživatel zkontrolovat, kam výsledek ukládá.

5.2.3 Výpočetní část, vykreslení hodnotového měřítka

Po definování všech potřebných widgetů v programu, přišlo na řadu naprogramování samotného jádra programu a to deklarace proměnných z vyplněných parametrů od uživatele a výpočet všech potřebných vzorců pro vykreslení všech symbolů jakožto součást tvorby hodnotového měřítka.

Celý proces se spustí po kliknutí na tlačítko *Refresh graph*. Jádro programu obsahuje jeden velký cyklus, který se větví na dvě větve, přičemž jedna větev odpovídá nabídce *Classification Method – Automatically*, respektive jaké další události nastanou, když uživatel zvolí tuhle nabídku, a naopak pokud zvolí *Classification method – Manually*.

Pokud uživatel zvolí, že se velikosti symbolů mají automaticky vypočítat, program si načte hodnoty z parametrů *Symbol size from a Symbol size to*, pro které vytvoří svůj vnořený cyklus, kde tyto hodnoty reprezentují nejnižší a nejvyšší hodnotu na ose Y. Dále podle počtu tříd vymezí počet kroků, neboli co jeden krok, to jeden průchod cyklem a jeden vykreslený symbol. Cyklus končí, když hodnota na ose Y dosáhne maximální zadané velikosti symbolu. Tímto je dosaženo plynulého zvyšování velikosti symbolu. Cyklus samozřejmě přihlíží na to, jaký tvar si uživatel vybral a dle toho vykreslí správný symbol v průchodu cyklem.

Pokud nastane varianta manuálního výpočtu velikosti symbolu, v programu se objeví nový sloupec s odpovídajícím počtem řádků podle počtu tříd /intervalů. Princip průchodu cyklem je založen na stejném principu jako při předchozí možnosti až na fakt, že všechny hodnoty velikosti symbolu, které uživatel ručně zadal pro každý interval zvlášť, se načítají do jedné proměnné jako pole hodnot a podle jejich indexu se tyto hodnoty postupně využijí v cyklu pro vykreslení velikosti daných symbolů. Jako i v předchozím případě se přihlíží na vybraný tvar symbolu.

Hodnotové měřítko kartodiagramů je vyhotoveno v černo-bílé podobě. Pokud by se uživatel chtěl touto vizuální stránkou více zabývat, může si měřítko vyexportovat do vektorové podoby *.svg, která je naimplementována právě z tohoto důvodu.

Ukázka hotového hodnotového měřítka požitého v mapě pro tvar symbolu Circle 1 a Square 1 je k nahlédnutí v Příloze 1 a Příloze 2.

Pro názornou ukázku bylo použito vykreslení symbolu *Square 1* v prvním i druhém případu procházení cyklem:

```
number_of_classes = int(self.box3.get())

if self.boxmethod.get() in ('Automatically'):
      deltay = MaxSymbol - MinSymbol   #rozsah hodnot na ose Y
```

```
        pocet = float(self.box3.get())    #pocet vykreslenych symbolu
                                           (=pocet intervalu)
        step = deltay/(pocet - 1)

        #cx = souradnice stredu elipsy na ose X

        y = MinSymbol
        valueto0 = None
        for i in range(number_of_classes):
            if self.box2.get() in ('Square 1'):
                  cx = 170.0
                  rectangle = plt.Rectangle((cx, 0), width= y, height = y,
                              fill=False)
                  ax1.add_patch(rectangle)
                    ……
            y += step

else:
        valueto0 = None
        rmax = max(float(entry.get())/2.0 for entry in self.size_entry )
        for i in range(number_of_classes):
            y = float(self.size_entry[i].get())
            if self.box2.get() in ('Square 1'):
                   cx = 170.0
                   rectangle = plt.Rectangle((cx, 0), width= y, height =
                               y, fill=False)
                   ax1.add_patch(rectangle)
                               ……
```

5.3 Graduated symbols pro liniovou vrstvu

Další nabídka Graduated symbols pro liniovou vrstvu je obměnou pro předchozí nabídku. Touto cestou lze v produktu ArcGIS vytvořit kartodiagramy liniové s intervalovou stupnicí. Problém s hodnotovým měřítkem je totožný jako v předchozí kapitole. Jeho zpracování je sice kartograficky přijatelné, ovšem hodnotové měřítko vytvořené ve vlastním programu splňuje všechna pravidla správnosti.

Parametry pro tvorbu těchto kartodiagramů v prostředí ArcGIS jsou stejné jako v předchozím případě ovšem přizpůsobené pro liniovou vrstvu a skládají se tedy opět z:

A) volby dat, která mají být vizualizována

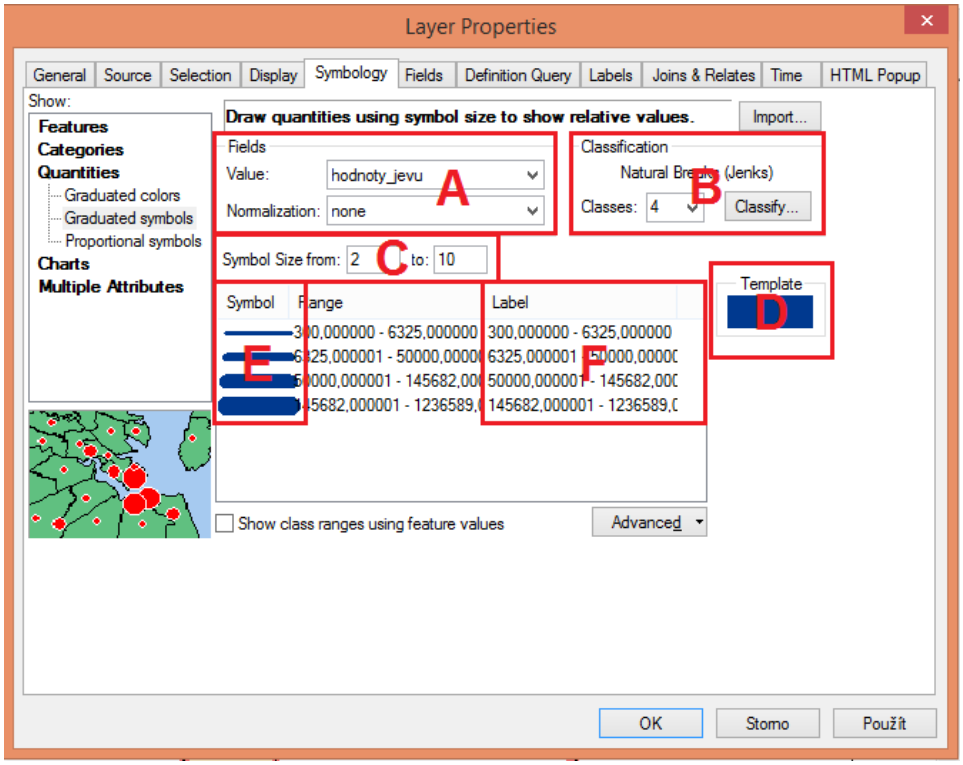
B) počtu tříd, do kterých mají být data klasifikována

C) definování minimální a maximální velikosti symbolů

D) výběru tvaru a barevného provedení symbolu

E) možnosti manuálně editovat každý symbol zvlášť

F) editace popisu každého symbolu neboli definování intervalů



Obr. 26 Definování parametrů pro tvorbu liniové kartodiagramy v prostředí ArcGIS.

Definice parametrů ve vlastním programu v Tkinteru ve srovnání s označením parametrů v prostředí ArcGIS, bližší vysvětlení parametrů bude opět popsáno níže v kapitole 5.3.2 Vysvětlení parametrů:

B) počet tříd, do kterých mají být data klasifikována (možnosti: *od 2 až po 10 symbolů*)

C) definování minimální a maximální velikosti symbolů (pokud je zvolena klasifikační metoda – *Automaticky)*

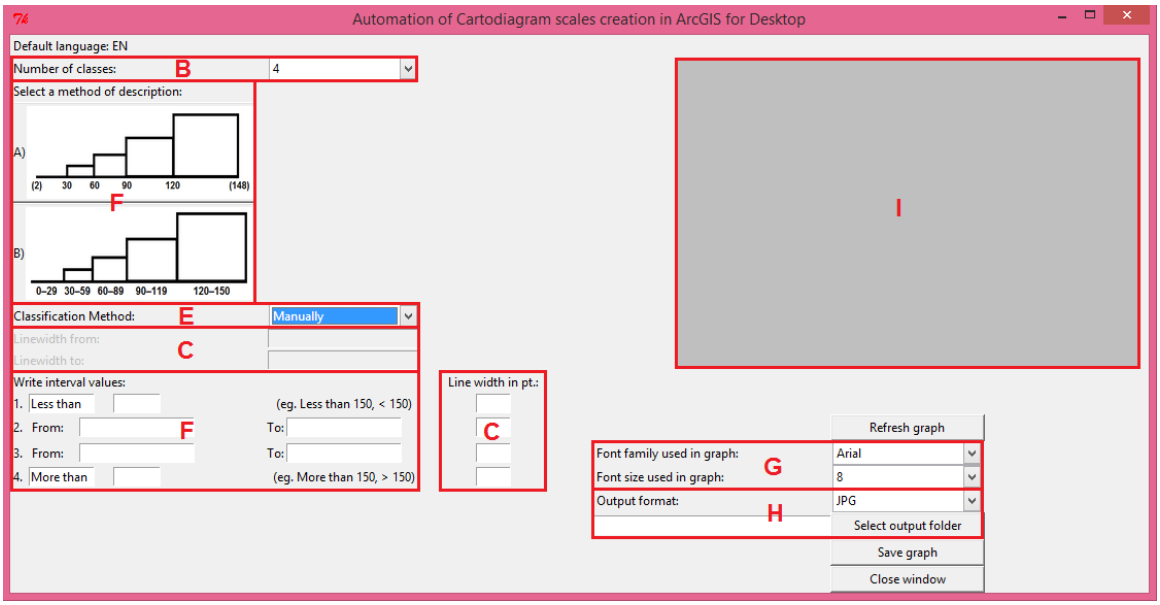
E)klasifikační metoda pro určení velikosti symbolů (*Automaticky* nebo *Manuálně*)

F) editace popisu každého symbolu neboli definování intervalů

G) volba velikosti a druhu fontu použitém v popisu

H) výběr výstupního formátu a místa uložení hodnotového měřítka

I) plocha pro vykreslení hodnotového měřítka



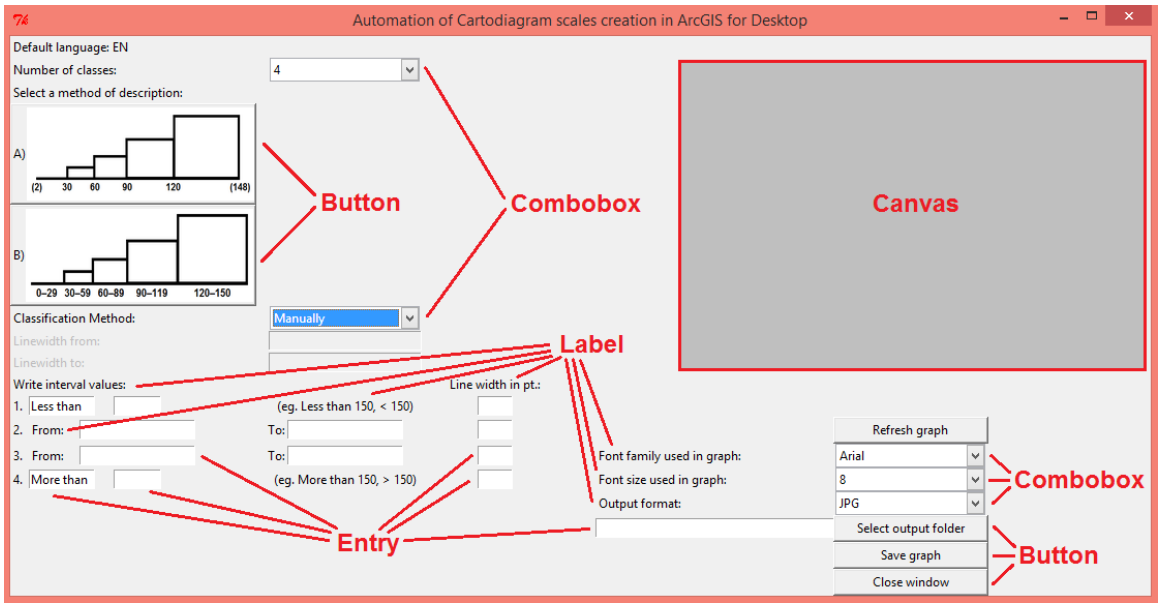
Obr. 27 Nabídka ve vlastním navrženém programu pro liniové kartodiagramy odpovídající k nabídce v prostředí ArcGIS.

Tento program vychází z programu předchozího, je založen na stejném základu a principu, tudíž má implementovány i stejné knihovny ke svému fungování. Navíc ovšem pracuje s knihovnou **PIL** celým názvem Python Imaging Library, která dokáže načíst obrázky ve více než 30ti formátech a konvertovat je do objektů, se kterými je Tkinter kompatibilní. Oproti předchozímu programu nebude využita knihovna Math, jelikož zde není potřeba žádných výpočtů s matematickými vzorci. Dalo by se také říci, že je tento program o něco jednodušší než program předchozí.

5.3.1 Design programu

Jak již bylo výše zmíněno, tvorba programu vycházela z programu předchozího. Tudíž byl částečně převzat i design programu, zvolené parametry a příslušné widgety. Nově přidaným parametrem je nabídka druhu popisu liniového kartodiagramu. Zde si uživatel může vybrat mezi popisem lomových bodů liniového kartodiagramu nebo zvolit popis formou intervalů. Po rozhodnutí, jaký druh popisu si uživatel vybere, se odvíjí i následný design programu. Pro každý druh popisu je totiž naprogramována jiná nabídka, jelikož se parametry související s druhem popisu liší. Naopak byla vynechána nabídka tvaru symbolu.

Ze seznamu použitých widgetů, byl opět použit Label, Combobox, Entry, Button a Canvas z důvodu zachování určité jednotnosti napříč všemi programy. Změnou oproti předchozímu programu je nastylování tlačítka Button o přidanou hodnotu v podobě obrázku. Obrázek byl vytvořen kvůli větší názornosti a vyšší úspoře textového vysvětlení k druhům popisu. Obrázek byl vytvořen v externím programu a následně načten do vlastního programu pomocí příkazů z knihovny Os.



Obr. 28 Grafické znázornění použitých widgetů ve vlastním programu specializujícího se na tvorbu hodnotového měřítka pro liniové kartodiagramy.

5.3.2 Vysvětlení parametrů

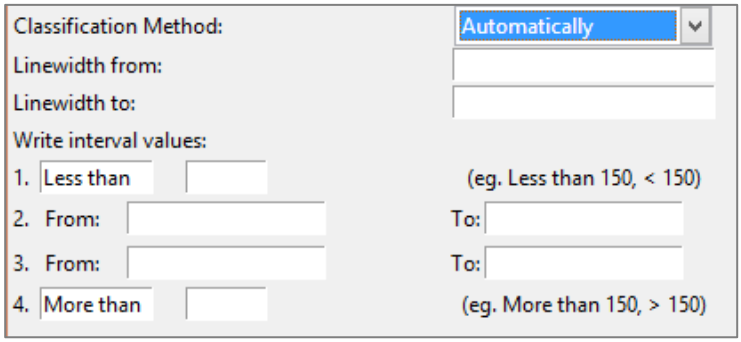
První parametr byl pojmenován *Number of classes* a k vybrání poskytuje možnosti *2 až 10* tříd. Hodnota *1* byla vynechána, protože bylo zhodnoceno, že vytvářet hodnotové měřítko pro jednu třídu nemá smysl.

Select a method of description nabízí uživateli dva druhy popisu hodnotového měřítka:

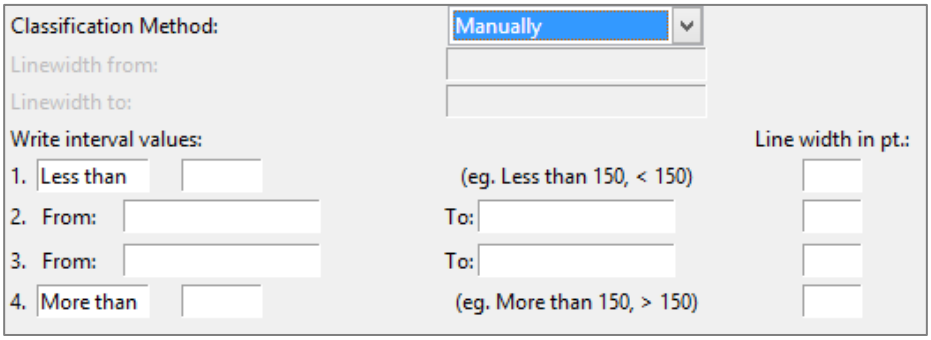
A) Popis lomových bodů hodnotového měřítka, kdy první a poslední hodnota je v závorkách, což znamená, že není povinná a uživatel ji nemusí zadat. Popis je vycentrován na začátek každé nové třídy.

B) Popis třídy hodnotového měřítka pomocí intervalů (tedy hodnot od–do). Tento popis je pak vycentrován na střed každé třídy a všechny hodnoty jsou pro uživatele povinné vyplnit.

Dalším potřebným parametrem pro vytvoření hodnotového měřítka je *Classification Method*, který umožňuje uživateli výběr mezi metodou *Automatically* nebo *Manually*. Tento parametr je převzat z předchozího programu a pracuje na stejném principu tentokrát pro výpočet šířky liniového kartodiagramu samozřejmě v jednotkách používaných v prostředí ArcGIS - tedy v Point. Možnost *Automatically* zpřístupňuje uživateli vidět další dva parametry na něj navazující, tedy parametry *Linewidth from a Linewidth to,* které obsahují vstupní pole pro zadávání minimální šířky liniového kartodiagramu a jeho maximální šířky, také v jednotkách pt. Po vybrání možnosti *Manually* se opět vytvoří nový sloupec s počtem řádků odpovídající počtu tříd. Tyto řádky obsahují vstupní pole widgetu Entry a uživatel zde zadává hodnoty šířky liniového kartodiagramu pro každou třídu zvlášť v jednotkách pt.

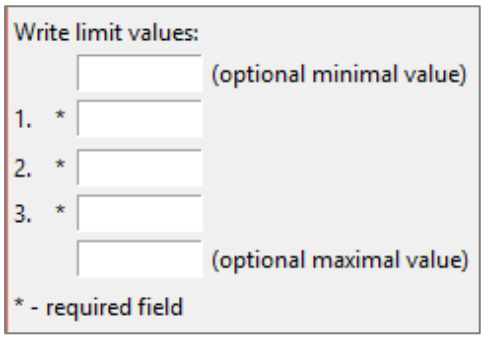


Obr. 29 Výřez z programu ukazující nabídku *Classification Method – Automatically* a zpřístupněné parametry *Linewidth from* a *Linewidth to.*



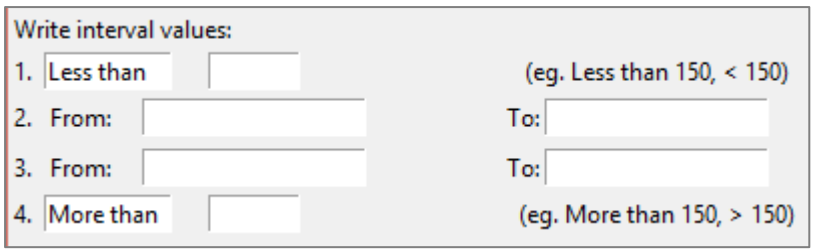
Obr. 30 Výřez z programu ukazující nabídku *Classification Method – Manually*, „zašedlé“ parametry *Linewidth from* a *Linewidth to* a přidaný nový sloupec pro manuální zadávání šířky liniového kartodiagramu.

Podle zvoleného druhu popisu se zobrazí nabídka pro zadání vlastních hodnot. Pokud uživatel zvolil popis lomových bodů intervalů, podle počtu tříd se vygeneruje příslušný počet vstupních polí pro zadání hodnot. Hvězdičkou jsou označeny ty kategorie, jejichž hodnoty uživatel musí vyplnit – jsou povinné. Kategorie bez číselného označení signalizují vstupní pole nepovinné.



Obr. 31 Ukázka druhu popisu – lomové body intervalů.

Druhý druh popisu je znám už z předchozího programu. Skládá se ze sloučení dvou hodnot od–do, které tvoří interval a první a poslední třída má možnost napsání vlastního textu.



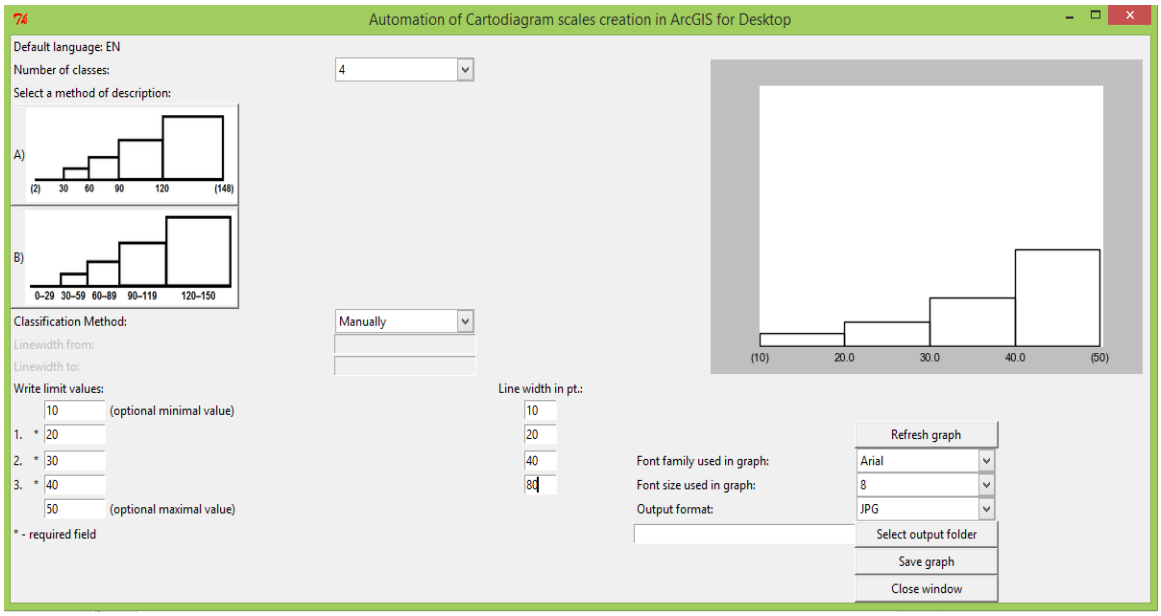
Obr. 32 Ukázka druhu popisu – použití intervalů.

Zbylé parametry, myšleno parametry v pravé části programu se nemění a jsou stejné jako v předchozím případě. Jedná se o parametry: *Font family used in graph, Font size used in graph, Output format a Select output folder.*

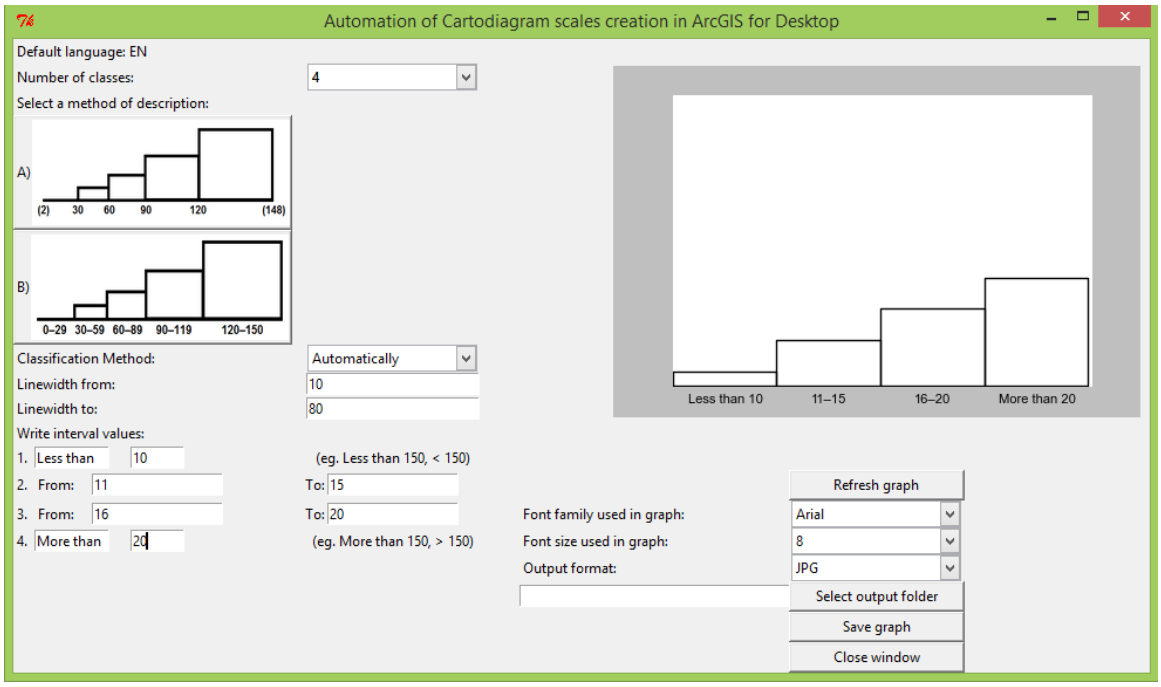
5.3.3 Výpočetní část, vykreslení hodnotového měřítka

Po kliknutí na tlačítko *Refresh* graph si program z vytvořených widgetů a uživatelem vyplněných vstupů načte hodnoty do jednotlivých proměnných, které dále použije ve svém výpočetním jádru podle nabídky, kterou uživatel zvolil. Výsledkem jsou vždy na sebe navazující symboly ve tvaru obdélníku reprezentující liniovou vrstvu s rostoucí velikostí, tedy šířkou linie.

Výpočetní cyklus je napsán stejně jako v předchozím případě, ovšem zjednodušený, jelikož cyklus nemusí při svém průchodu přihlížet ještě na vybraný tvar symbolu.



Obr. 33 Ukázka výsledku automatizované tvorby hodnotového měřítka liniového kartodiagramu (nastavení: 4 třídy, popis lomových bodů, manuální zadávání šířky liniových kartodiagramů).



Obr. 34 Ukázka výsledku automatizované tvorby hodnotového měřítka liniového kartodiagramu (nastavení: 4 třídy, popis pomocí intervalů, automatické zadávání šířky liniových kartodiagramů).

Hodnotové měřítko kartodiagramů je i zde k dispozici v černo-bílé podobě. Pro dodání barevného vjemu si může uživatel měřítko vyexportovat do vektorové podoby *.svg a v příslušném grafickém programu jej dotvořit.

Ukázka použití hodnotového měřítka v mapě je k nahlédnutí v Příloze 3.

Pro ukázku bylo použito naprogramování obou druhů popisu při průchodu cyklem:

1) Popis lomových bodů intervalů kartodiagramu

```
...
for i in range(number_of_classes+1):
    if i != number_of_classes:
        rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                    fill=False)
        ax1.add_patch(rectangle)
    if 0<i<number_of_classes:
        valueto = float(self.to_entry[i-1].get())
        title = valueto
        if valueto < valueto0:
                messagetxt = "Your value %s <  previous value %s, please
                             change your values." % (valueto, valueto0)
                message = tkMessageBox.showerror("Warning", messagetxt)
        valueto0 = valueto
```

```
    elif i==0:
        valueto0 = float(self.class1_entry.get())
        title = u'(%g)' % (valueto0)
    else:
        valueto = float(self.classLast_entry.get())
        title = u'(%g)' % (valueto)
        if valueto < valueto0:
                messagetxt = "Your LAST value %s <  previous value %s,
                     please change your values." % (valueto, valueto0)
                message = tkMessageBox.showerror("Warning", messagetxt)
    start, end = ax1.get_xlim()
    ax1.xaxis.set_ticks(np.arange(start, end, step))
    labels.append(title)
ax1.set_xticklabels(labels)
```

1) Popis lomových bodů intervalů kartodiagramu

```
...
for i in range(number_of_classes):
    if i != number_of_classes:
        rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                     fill=False)
        ax1.add_patch(rectangle)
    if 0<i<number_of_classes-1:
        entryname_from = "class%ia_entry" % (i+1)
        valuefrom = float(getattr(self, entryname_from).get())
        entryname_to = "class%ib_entry" % (i+1)
        valueto = float(getattr(self, entryname_to).get())
        title = u'%g-%g' % (valuefrom, valueto)
        if valueto <= valuefrom:
                messagetxt = "Your value 'To:' %s <= value 'From:' %s,
                    please change your values for the description of
                    class number %i" % (valueto, valuefrom, i+1)
                message = tkMessageBox.showerror("Warning", messagetxt)
        elif valuefrom <= valueto0:
                messagetxt = "Your value 'From:' %s < previous value
                    'To:' %s, please change your values for the
                    description of class number %i" % (valuefrom,
                    valueto0, i+1)
                message = tkMessageBox.showerror("Warning", messagetxt)
        valueto0 = valueto
    elif i==0:
        title = " ".join((self.class1_entry_text.get().rstrip(),
                self.class1_entry.get()))
        valueto0 = float(self.class1_entry.get())
```

```
else:
    title = " ".join((self.classLast_entry_text.get().rstrip(),
            self.classLast_entry.get()))
    valuefrom = float(self.classLast_entry.get())
    if valuefrom < valueto0:
            messagetxt = "Your LAST value %s <  previous value 'To:'
                 %s, please change your values." % (valuefrom,
                 valueto0)
            message = tkMessageBox.showerror("Warning", messagetxt)
start, end = ax1.get_xlim()
ax1.xaxis.set_ticks(np.arange(start, end, step))
labels.append(title)
            ...
```

5.4 Proportional symbols

Nabídka Proportional symbol ve vlastním programu nabízí vytváření kartodiagramu s využitím funkční stupnice. Je společná jak pro bodovou, tak i liniovou vrstvu.

Tvorba samotných kartodiagramů je v prostředí ArcGIS velmi rozmanitá a lze nastavit mnoho detailů. Ovšem, co se týče hodnotového měřítka pro kartodiagramy s využitím funkční stupnice, s tím si ArcGIS už nedokáže tolik poradit a vygeneruje pouze počet symbolů daných počtem tříd s různou velikostí symbolů vypočítanou podle vzorce a ke každému konkrétní hodnotu. Na první pohled se tedy velmi neliší od kartodiagramů s intervalovou stupnicí. Pro použití v mapách bývá toto hodnotové měřítko většinou dotvářeno v jiných externích programech a pak importováno do prostředí ArcGIS.

Parametry pro tvorbu těchto kartodiagramů v prostředí ArcGIS se skládají z:

A) volby dat, která mají být vizualizována

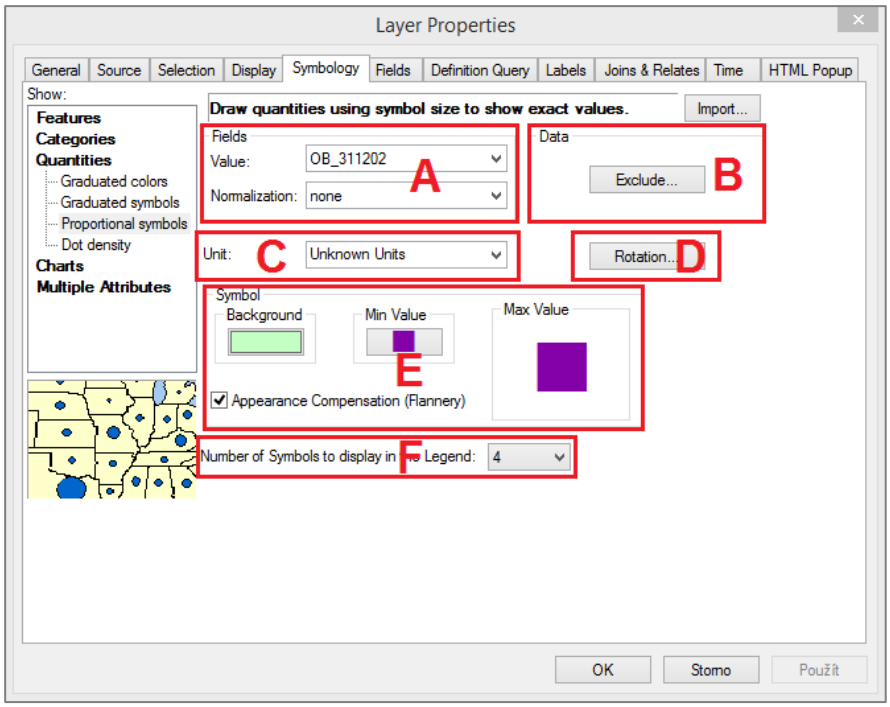
B) prostor pro definování vlastního SQL příkazu

C) volby jednotek pro výpočet velikosti symbolu

D) nastavení rotace symbolu

E) definování minimální a maximální velikosti symbolů, výběru metody výpočtu velikosti kartodiagramu

F) počtu tříd, do kterých mají být data klasifikována



Obr. 35 Nabídka parametrů pro vytváření kartodiagramů s využitím funkční stupnice v prostředí ArcGIS.

Seznam parametrů ve vlastním programu srovnávající se s označením parametrů v prostředí ArcGIS, bližší vysvětlení parametrů bude popsáno opět níže v kapitole 5.4.2 Vysvětlení parametrů:

A) volby dat, která mají být vizualizována

E) definování minimální velikosti symbolů, výběru metody výpočtu velikosti kartodiagramu

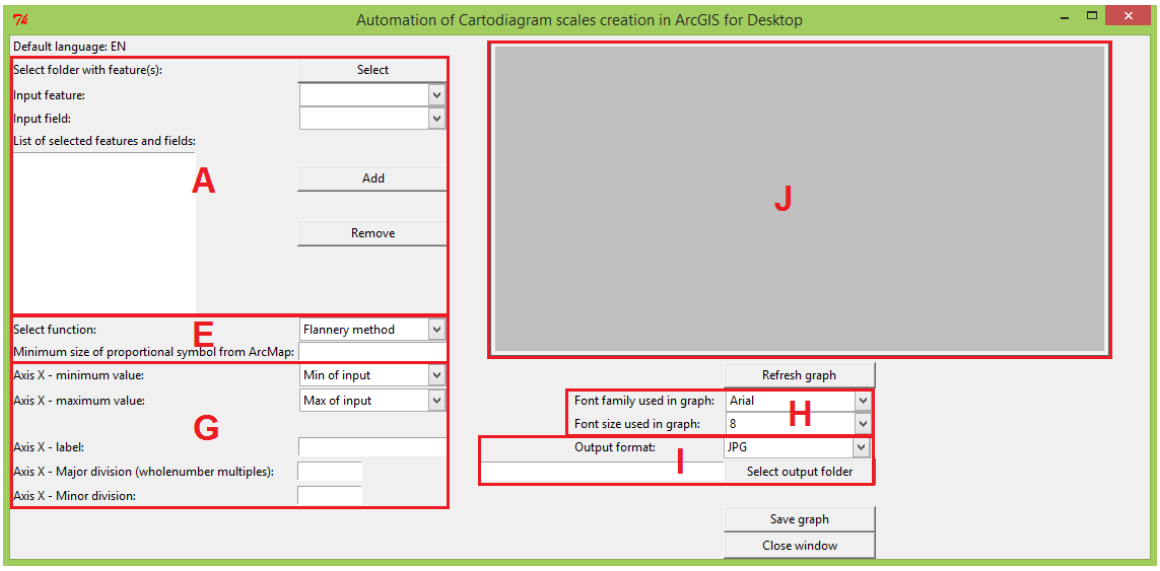
G) vlastnosti osy X

H) volba velikosti a druhu fontu použitém v popisu osy X

I) výběr výstupního formátu a místa uložení hodnotového měřítka

J) plocha pro vykreslení hodnotového měřítka

Skok v označení parametrů znamená, že tyto parametry ve vlastním programu přímo použity nebyly, jelikož pro tvorbu hodnotového měřítka nejsou zapotřebí nebo se u parametru vybrala jedna jeho varianta, se kterou se automaticky dál počítá v celém programu, a tudíž nemusí být pro uživatele viditelná v nabídce programu.



Obr. 36 Grafický náhled na nabídka parametrů ve vlastním programu podle parametrů z programu ArcGIS a nově definované parametry.

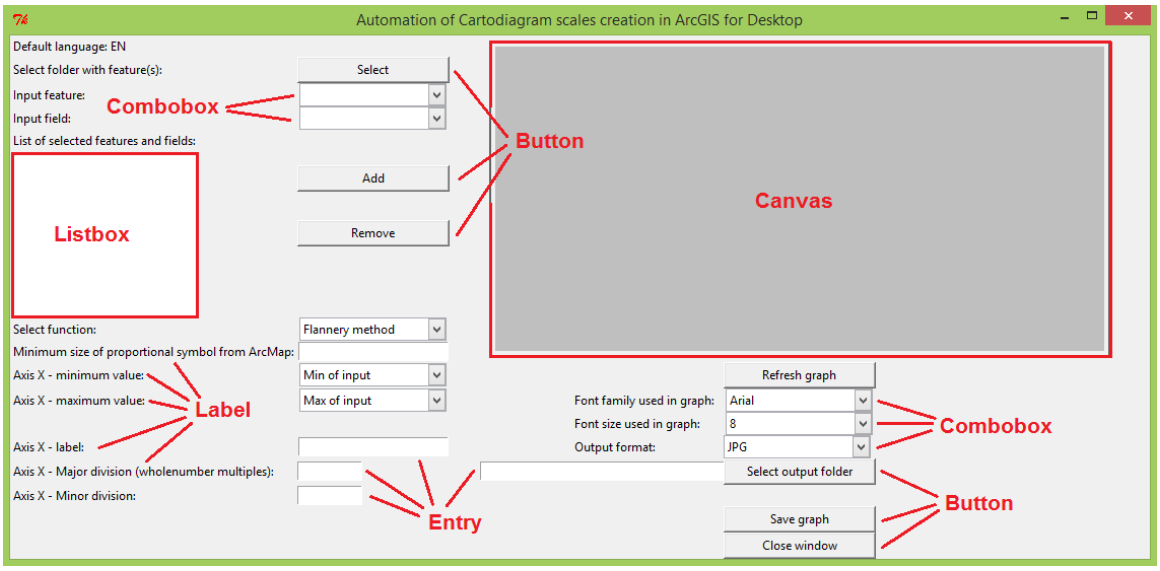
Tento program je ve srovnání s předchozími dvěmi programy založen na zcela jiných principech, parametrech, výpočtech a naprogramovaných cyklech. Věc, kterou má s předchozími programy společnou, jsou naimportované knihovny a použité widgety. Využívané jsou tedy knihovny Tkinter, Matplotlib, Os a nově přidanou je knihovna **Arcpy**. Tato knihovna poskytuje způsob, jak provést geografickou analýzu dat, konverzi dat, správu dat a automatizaci tvorby map v jazyce Python. Knihovna byla vyvinuta společností Esri právě pro využití ve svých produktech jako je například ArcGIS for Desktop. Ve vlastním programu byla využita právě pro práci s geografickými daty, konkrétně pro jejich načtení do programu a procházení jejich atributů.

5.4.1 Design programu

Pro tvorbu hodnotového měřítka s využitím funkční stupnice bylo zaprvé nejdůležitější si rozmyslet, které všechny parametry jsou pro tvorbu měřítka klíčové a zadruhé je pojmenovat tak, aby uživatel správně pochopil, co a kam má vyplnit a jak tento program funguje.

Pořadí parametrů bylo řazeno tak, aby uživatel nejprve vybral a načetl do programu jednu nebo více vrstev, pro které chce měřítko vytvořit, dále vybral výpočetní metodu a nejmenší velikost symbolu kartodiagramu v pt. a dále nastavil různé vlastnosti osy X. Pravá strana programu je opět využita pro nastavení písma a v konečné fázi tvorby hodnotového měřítka pro formát exportu a jeho místa uložení na disku.

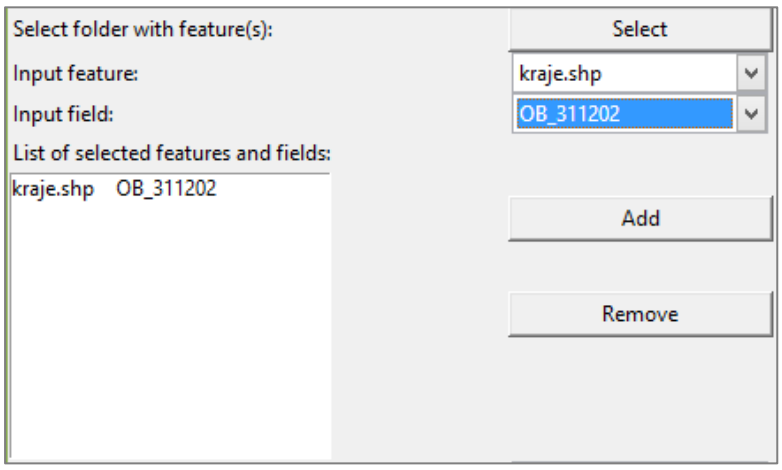
Výčet použitých widgetů zůstává prakticky stejný – Label, Combobox, Entry, Button a Canvas. Novinkou je ovšem přidání widgetu **Listbox**, který se používá pro zobrazení seznamu hodnot. Tento widget může obsahovat jen textové položky a všechny musí mít stejnou barvu a druh fontu. Listbox poskytuje mnoho nastavení a díky tomu si může uživatel označit např. buď jen jednu položku v Listboxu, nebo i více naráz.



Obr. 37 Grafické znázornění použitých widgetů ve vlastním nástroji rozšířen o použití widgetu Listbox.

5.4.2 Vysvětlení parametrů

Výběrem vstupní vrstvy začíná celý proces tvorby hodnotového měřítka. První parametr na nástroji nese název *Select folder with feature(s)*. Po kliknutí na tlačítko *Select* se uživateli otevře okno s adresáři v počítači a vyzve jej k vybrání vstupních souborů. Těmi mohou být jen shapefile (dále jen shp) soubory a může jich být i více. Po načtení souborů se uživatel rozhodne, které atributy shp souboru chce zahrnout do hodnotového měřítka. V záložce *Input Feature* stačí vybrat shp soubor a automaticky se v další záložce *Input Field* načtou všechny jeho atributy (tak jak je známe např. z atributové tabulky) a stačí vybrat příslušný atribut a tlačítkem *Add* jej potvrdit. Tímto způsobem může uživatel navolit i další atributy z dalších shp. Veškeré vybrané atributy se po vybrání ukáží v *List of selected features and fields*, kde je také možné při nespokojenosti svůj výběr vymazat tlačítkem *Remove*.



Obr. 38 Výřez z vlastního nástroje zobrazující část načítání .*shp souboru a jeho atributu do listboxu, z něhož se bude vytvářet hodnotové měřítko.

Dalším krokem uživatele je výběr metody pro výpočet hodnotového měřítka (*Select function*). Nástroj nabízí obě kartograficky známé metody, jimiž jsou:

- **Normální metoda** (tzv. Normal case) – jedná se o standartní metodu, která je nastavena v produktu ArcGIS jako výchozí metoda a její vzorec je následující:

$$\mathrm{Pj} = \left(\frac{Valj}{Valmin}\right)^{0,5} * Pmin$$

Pj……velikost j-tého objektu
Valj…….hodnota j-tého objektu
Valmin……minimální hodnota v rámci využitého výběru dat pro tvorbu kartodiagramu
Pmin……minimální velikost bodu (zadávána ručně z prostředí Arcmap)

- **Flanneryho metoda** (též někdy Appearance compensation), která má oproti normální metodě modifikován vzorec z důvodu empiricky ověřeného vnímání změny velikosti diagramů uživateli map. Vzniklým rozdílem je větší změna ve velikostech prvků (graf stoupá strměji vzhůru). Tento rozdíl je při menších hodnotách téměř nepatrný, ale v případě, že je pole hodnot velké, rozdíly mezi grafy jsou markantní. Modifikovaný vzorec vypadá následovně:

$$\mathrm{Pj} = 1{,}0083 * \left(\frac{Valj}{Valmin}\right)^{0,5716} * Pmin$$

Po výběru metody musí uživatel manuálně zadat minimální velikost symbolu (*Minimum size of proportional symbol from ArcMap*), tedy výše uvedené *Pmin*. Je to obdobné jako v softwaru ArcGIS, potencionálními hodnotami pro vykreslení symbolu jsou 1 až 100pt.

Následuje nastavení minimální hodnoty pro osu X (*Axis X – minimum value*), která představuje hodnoty vybraného atributu. Zde nástroj nabízí 3 hodnoty:

- *Min of input* – jedná se o minimální hodnotu obsaženou v atributu (např. 297 867 u počtů obyvatel ve vrstvě krajů České republiky).
- *Custom value* – vlastní volba uživatele, který zadá manuálně hodnotu, kterou bude osa X začínat. Jestliže uživatel vybere tuto možnost, objeví se textové pole pro zapsání hodnoty.

Obdobě se postupuje i u volby maximální hodnoty na ose X (*Axis X – maximum value*).

Osu X je možné i popsat, slouží k tomu vstupní podle vedle widgetu (*Axis X – label*). Osa Y zůstává skrytá, jelikož ve většině případů hodnotových měřítek vykreslená není. Na této ose by byla číselně znázorněna velikost symbolu, který je v grafu vykreslen.

Na ose X lze nastavit i hlavní a vedlejší dělení. Toto dělení je reprezentováno dvěma parametry *Axis X - Major division* a *Axis X – Minor division*. Hodnoty do těchto dvou parametrů se zadávají jako nejbližší celočíselné násobky od minimální hodnoty na ose X. Popis vedlejšího dělení se kvůli vyšší přehlednosti měřítka na ose X nezobrazuje.

Pro změny vlastností písma v grafickém poli slouží další widgety. Pomocí *Font family used in graph* je možno změnit font písma, na výběr jsou fonty *Arial, Times New Roman, Calibri a Verdana.* Jako další se dá změnit velikost písma (*Font size used in graph*), nabízeny jsou velikosti *8 až 12 a 15*.

Výsledek lze uložit do 4 formátu – 3 rastrové a 1 vektorový. Výběr rastrových formátů je omezen na nejvíce využívané - **.jpg, *.png* a **.tiff*. Vektorový formát je zde zastoupen formátem **.svg*.

Parametr označený písmenem za B) v nabídce z prostředí ArcGIS představující *prostor pro definování vlastního SQL příkazu* byl ve vlastním nástroji vynechán z důvodu přiblížení se co největšímu zjednodušení při automatizaci tvorby hodnotového měřítka pro kartodiagramy využívající funkční stupnici. Pro tvorbu měřítka není tento parametr klíčový a tak byl z nabídky ve vlastním nástroji odstraněn.

Units neboli jednotky násobí hodnotu jevu geografickými (mapovými) jednotkami pro výpočet velikosti symbolů. V možnostech jsou na výběr tyto jednotky – *Unknown unit, Centimeters, Decimal Degrees, Decimeters, Feet, Inches, Kilometers, Meters, Miles, Milimeters, Nautical Miles, Points* a *Yards.* Pokud jsou jednotky nastaveny na *Unknown units*, může uživatel specifikovat hodnotu velikosti symbolu pro nejmenší hodnotu jevu v datech. V ostatních případech se velikosti symbolů vynásobí mapovými jednotkami a uživatel nemůže do tohoto výpočtu zasahovat. Proto bylo ve vlastním nástroji bráno jako defaultní nastavení jednotek možnost Unknown a tím pádem může uživatel ručně nastavit velikost symbolu pro nejmenší hodnotu jevu v datech.

Posledním vynechaným parametrem bylo určení tzv. *Rotace* (*Rotation*). Jednotlivé symboly se v hodnotovém měřítku vizualizovat nebudou, bude vykreslen průběh funkce kartodiagramu, tudíž zadávání rotace symbolu je v tomto případě zbytečné a bezpředmětné.

5.4.3 Výpočetní část, vykreslení hodnotového měřítka

Výpočetní část zde vypadá trochu jinak než v předchozích nástrojích. Po kliknutí na tlačítko *Refresh graph* spustí nástroj celou výpočetní a vykreslovací část. Proces probíhá ovšem následovně a to pokaždé, když uživatel opětovně klikne na toto tlačítko např. při změně nějakého parametru během procesu tvorby měřítka:

Položky v Listboxu si program převede jako uspořádanou dvojici - vrstvy a jejího vybraného atributu. Pomocí knihovny Arcpy a příkazu *SearchCursor* projde každý řádek atributu a hodnotu si uloží do pole jedné proměnné. Tento proces se uskuteční v případě jedné či více položek uložených v Listboxu. Dále si program do dalších proměnných uloží minimální a maximální hodnotu z jednoho či více těchto atributů zároveň. Tyto hodnoty jsou důležité pro vykreslení minimální a maximální hodnoty v grafu, vykreslení funkce jakožto parametry potřebné k výpočtu a stanovení jejich celočíselných násobků – tedy hlavního a vedlejšího dělení.

Následuje celkové seřazení hodnot, které budou vykresleny na ose X. Podle vybrané metody výpočtu (tzv. Normal case nebo Flannery method) se pro hodnoty na ose X vypočítají jejich odpovídající hodnoty na ose Y, tedy hodnoty velikosti symbolu kartodiagramu, které dohromady tvoří křivku funkce výsledného hodnotového měřítka. Vypočítají se tedy hodnoty odpovídající minimu a maximu daného souboru hodnot.

Pokud ovšem uživatel zadá jiné minimum, než je minimum v souboru, provede se stejný proces, který byl popsán doposud a navíc se vypočítá rozdíl hodnot mezi minimem nalezeným v souboru hodnot a minimem zadaným od uživatele, provede se vygenerování hodnot mezi těmito body o určité hustotě a pro ně nástroj následně podle zvolené funkce vypočítá jejich odpovídající hodnoty na ose Y a přidá tuto vzniklou křivku k ostatním hodnotám v souboru. Společně tvoří tedy jednu souvislou křivku.

Stejný proces nastává, pokud uživatel zvolí jiné maximum, než je nalezeno v souboru hodnot načtených ze souborů *.shp. V tomto případě program opět vypočítá hodnoty pro osu Y z dat, které má načtené z příkazu SearchCursor a navíc vypočítá rozdíl hodnot mezi maximem definovaným od uživatele a maximem v souboru hodnot, provede opět vygenerování hodnot mezi těmito body určité hustoty a následně pro ně nástroj dopočítá odpovídající hodnoty na ose Y a vykreslí křivku. Tato křivka je pak souvislým pokračováním křivky vykreslené z originálních dat.

Tyto dva způsoby se pak kombinují podle zadání uživatele.

Po vykreslení křivky si pak program uloží do dalších proměnných hodnoty, jejichž násobky budou sloužit pro počítání hlavního a vedlejšího dělení pro osu X a následně provede výpočet, který přihlédne na zvolené minimum a nalezne nejbližší celočíselný násobek tohoto čísla a zde vykreslí první hlavní dělení. Další hlavní a vedlejší dělení vykreslí podle hodnoty uložené v proměnných zadané uživatelem.

Posledním vykreslovacím prvkem je zde vertikální linie, která je lokalizována v minimu a maximu celého souboru a v hlavních děleních hodnotového měřítka. Maximálních hodnot nabývá v hodnotě na ose Y pro příslušnou hodnotu na ose X podle zvolené metody výpočtu. Tato vertikální linie je zde proto, aby bylo uživateli naznačeno, jak velká je v této hodnotě velikost symbolu kartodiagramu. Simuluje tedy vykreslení symbolů naznačením jejich nejvyššího bodu.

Ukázka hotového hodnotového měřítka použitého v mapě je k nahlédnutí v Příloze 4.

Ukázka kódu při průchodu cyklem při nastavení: Axis X – minimum value – Min of input a Axis X – maximum value – Custom value:

```
...
elif self.box2.get() in ('Min of input') and self.box3.get() in ('Custom
value'):
     x = sorted(values)
     user_valmax = float(self.label2.get())
     if user_valmax < valmax:
               messagetxt = "Please insert higher value than maximum
                            value of input feature."
               message = tkMessageBox.showerror("Warning", messagetxt)

     density = (valmax-valmin)/float(len(x))
     stepx1 = density *(user_valmax-valmax)/(valmax-valmin)
     userx = valmax
     i = (len(x))
     while userx < user_valmax:
           x.insert(i, userx)
           yvalues.insert(i, f(self, userx, valmin, Pmin))
           userx += stepx1
           i += 1

     cx1 = majorTicks
     cy1 = None
     while cx1 < user_valmax:
           cy1 = f(self, cx1, valmin, Pmin)
           ax1.vlines(cx1, 0, cy1, color='black', lw=1)
           cx1 += majorTicks
     if cy1 is None:
           return

     valmax = user_valmax

     ax1.xaxis.set_minor_locator(minorLocator)
     ticks = [valmin]
     xval = (1+int(valmin)/int(majorTicks))*majorTicks
     while xval<=valmax:
           ticks.append(xval)
           xval += majorTicks
     ticks.append(valmax)
     if valmin == 0:
           cy = 0
     else:
           cy = f(self, valmin, valmin, Pmin)
     ax1.vlines(valmin, 0, cy, color='black', lw=1)
```

```
cy = f(self, valmax, valmin, Pmin)
ax1.vlines(valmax-1, 0, cy, color='black', lw=1)
ax1.set_xticks(ticks)
      ...
```

5.5 Charts (Pie, Bar/Column, Stacked)

Skupina Charts v produktu ArcGIS nabízí vytváření speciálních kartodiagramů, kterými jsou:

- **Pie** – strukturní kartodiagram nebo součtový
- **Bar/Column** – sloupcový kartodiagram
- **Stacked** – sloupcový součtový kartodiagram

Tyto kartodiagramy jsou u uživatelů velmi oblíbené, a proto jsou vyzdviženy ze všech ostatních druhů kartodiagramů a mají v programu ArcGIS svou vlastní nabídku pro jejich tvorbu. Společnými znaky jsou – více vstupních dat a důležitost barevného provedení.

5.5.1 Pie

Pomocí Pie chart lze díky prostředí ArcGIS vytvořit kartodiagram strukturní nebo součtový. Nabídka pro tvorbu těchto kartodiagramů je stejná, jediné, co je od sebe dokáže při tvorbě odlišit, je parametr *Size: Variation Type*.

Strukturní kartodiagram není v této práci rozepsán, jelikož tento kartodiagram má vždy stejnou velikost a nelze pro něj tedy vytvořit hodnotové měřítko. V této práci bude tedy rozebírána tvorba kartodiagramu součtového a následně tvorba jeho hodnotového měřítka.

Parametry pro tvorbu součtového kartodiagramu v prostředí ArcGIS vypadají následovně:

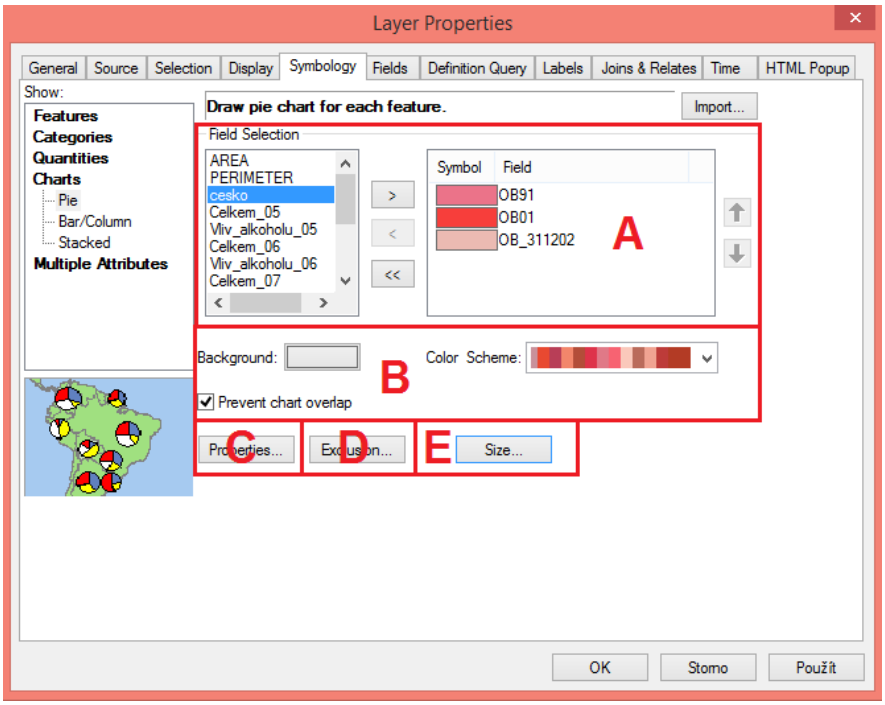
A) volba dat, která mají být vizualizována

B) definování barevného provedení symbolu i pozadí

C) vlastnosti kartodiagramu

D) prostor pro definování vlastního SQL příkazu

E) nastavení velikosti kartodiagramu



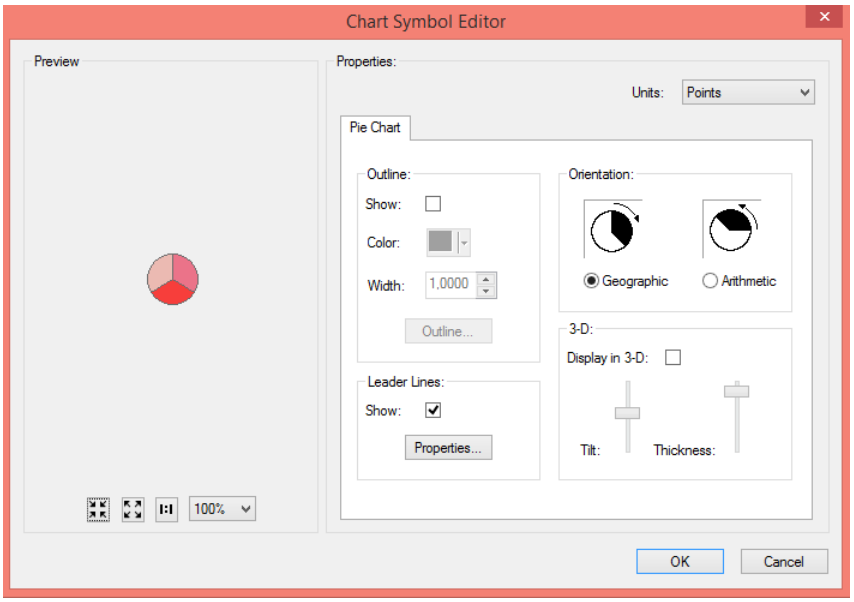
Obr. 39 Grafické znázornění parametrů potřebných pro tvorbu součtového kartodiagramu v programu ArcGIS.

Vysvětlení jednotlivých parametrů:

A) Volba dat, která uživatel chce vizualizovat, se skládá ze dvou částí. Na levé straně si uživatel vybírá ze seznamu všech atributů vrstvy a po dvojitém kliknutí nebo kliknutí na šipku „>“ se do pravé části tyto atributy převedou. Uživatel tak jasně vidí, pro která data se kartodiagram vytváří a jaké barevné provedení budou mít. Použití šipky „<“ vrátí vybraný atribut z pravé strany zpět do seznamu všech atributů vrstvy. Šipka „<<“ vrátí všechny vybrané atributy zpět do seznamu všech atributů. Použitím šipek nahoru a dolů na pravé straně si uživatel reguluje pořadí jednotlivých atributů v kartodiagramu.

B) Zde si uživatel vybírá barvy pro pozadí v mapě pod kartodiagramy a samotné barevné provedení jednotlivých částí součtového kartodiagramu pomocí předdefinovaných barevných palet. Zaškrtnutí checkboxu *Prevent chart overlap* znamená, že nenastane překrytí dvou kartodiagramů přes sebe.

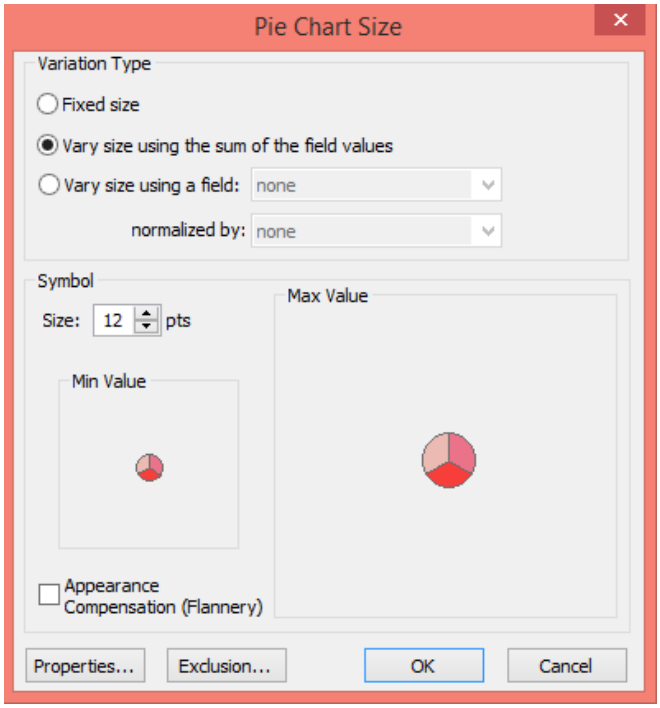
C) Po rozkliknutí nabídky *Properties…* se otevře dialogové okno s další rozmanitější nabídkou, kde si uživatel může detailněji nastavit vlastnosti týkající se samotného symbolu kartodiagramu – např. v jakých má být jednotkách, jestli má být vykreslen vnější okraj symbolu a jak má být široký, jaká má být orientace symbolu – používá se *Geographic* a jestli má být symbol ve 3D.



Obr. 40 Nabídka Properties… v nabídce Pie chart.

D) Po kliknutí na *Exclusion…* se otevře dialogové okno, kde si uživatel může napsat svůj specifický SQL příkaz, kde může např. vyloučit extrémní hodnoty, které by negativně ovlivnily tvorbu kartodiagramů

E) Nabídka *Size…* je pro tvorbu kartodiagramů velmi důležitá, ne-li klíčová. Pro tvorbu součtového kartodiagramu by měla být vybrána možnost *Vary size using the sum of the field values*, kde lze názorně vidět, jak vypadá minimální a maximální hodnota podle metody výpočtu velikosti kartodiagramu. Lze tedy zadat minimální hodnotu v pts a dole lze vybrat metodu výpočtu velikosti kartodiagramu a to konkrétně Flanneryho metodu.



Obr. 41 Nastavení v nabídce Size… v Pie chart.

Hodnotové měřítko součtového kartodiagramu také vychází z funkční stupnice. Nebyl tedy vytvořen další speciální program, ale využije se předchozího programu používaného také pro Proportional symbols. Tvorba legendy a hodnotového měřítka pro součtový kartodiagram se skládá ze dvou částí:

1) Po vytvoření součtového kartodiagramu v prostředí ArcGIS spustí uživatel program pro Automatizaci tvorby hodnotových měřítek kartodiagramů a vybere možnost Pie v nabídce Chart. Vyplní stejné hodnoty parametrů shodujících se s vyplněnými parametry z produktu ArcGIS a vytvoří hodnotové měřítko. Toto měřítko si následně vyexportuje a naimportuje do mapového prostředí v produktu ArcGIS.
2) V tomto kroku není tvorba legendy ještě úplná, měřítko je sice vytvořeno, ale chybí zde vysvětlivky ve struktuře kartodiagramu a jeho barevného provedení. Proto musí uživatel v prostředí ArcGIS vložit legendu, dále vybrat vrstvu, pro kterou byl součtový kartodiagram tvořen a vložit do mapy. Ještě stále není legenda správně vyhotovena. Nyní je nutné odstranit všechny nežádoucí texty, které ArcGIS vygeneroval a hlavně smazat text „Sum of fields“ a jeho hodnotu.

Nyní je už legenda pro součtový kartodiagram hotová – složená z hodnotového měřítka vykresleného ve vlastním programu a upravené části vysvětlující strukturu kartodiagramu a jeho barevné provedení vytvořené v prostředí ArcGIS.

5.5.2 Bar/Column

Díky Bar/Column chart v produktu ArcGIS lze vytvářet sloupcový kartodiagram. Tento druh kartodiagramu je většinou tvořen z více dat a každému sloupci odpovídá určitá hodnota jevu vyobrazená v určité barvě. Tento kartodiagram je pojmenován Bar/Column díky možnému nastavení orientace. Pokud je orientace nastavena jako Bar, je tento kartodiagram orientován horizontálně. Pokud je orientace přepnuta na Column, kartodiagram směřuje vertikálně.

Parametry pro tvorbu sloupcového kartodiagramu v prostředí ArcGIS obsahují stejnou nabídku jako předchozí typ Pie:

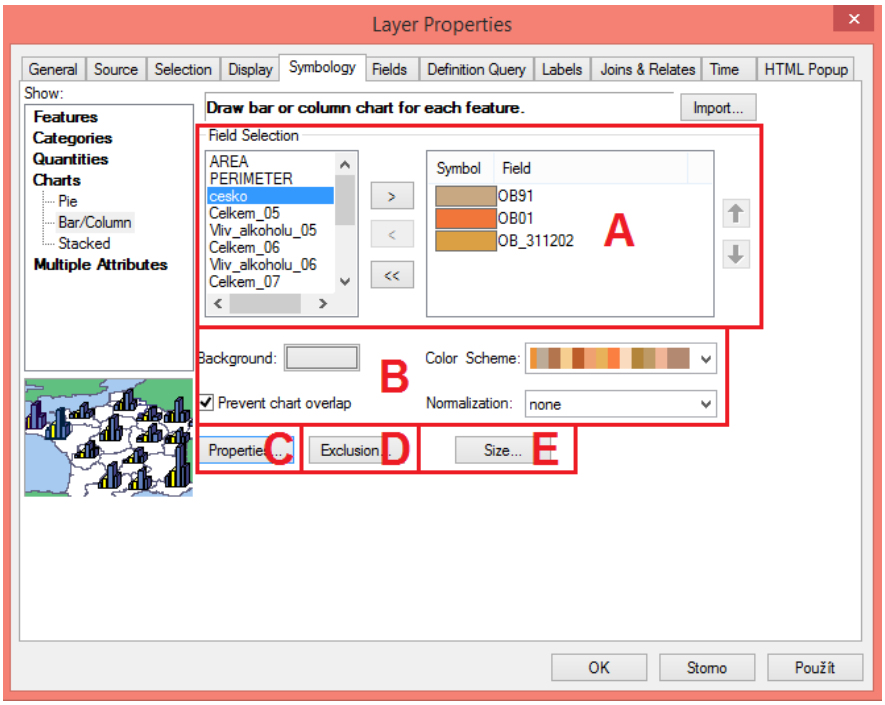
A) volba dat, která mají být vizualizována

B) definování barevného provedení symbolu i pozadí

C) vlastnosti kartodiagramu

D) prostor pro definování vlastního SQL příkazu

E) nastavení velikosti kartodiagramu

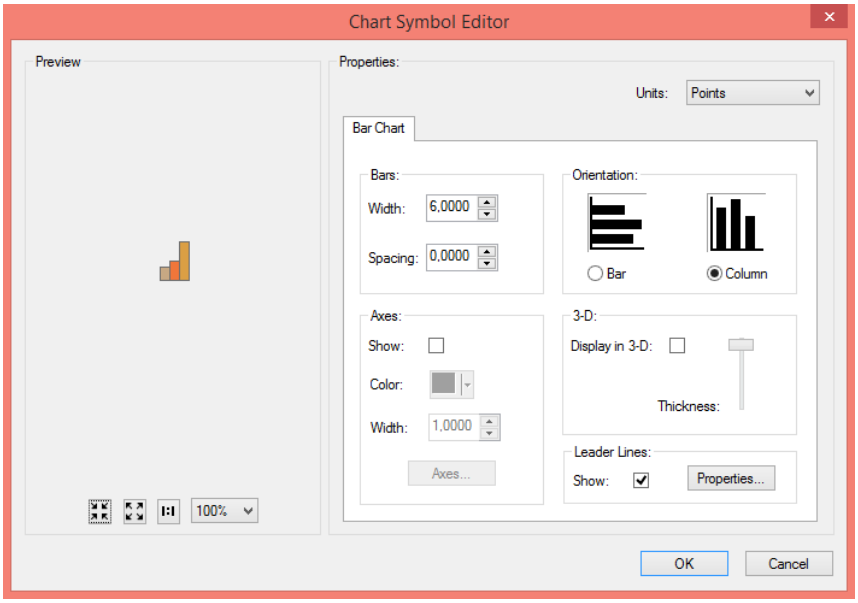


Obr. 42 Grafické znázornění nabídky parametrů potřebných pro tvorbu sloupcového kartodiagramu v produktu ArcGIS.

Vysvětlení jednotlivých parametrů:

A) Volba načítání dat a B) definování barevného provedení symbolu i pozadí je naprogramována ve stejném stylu jako u tvorby Pie chartu.

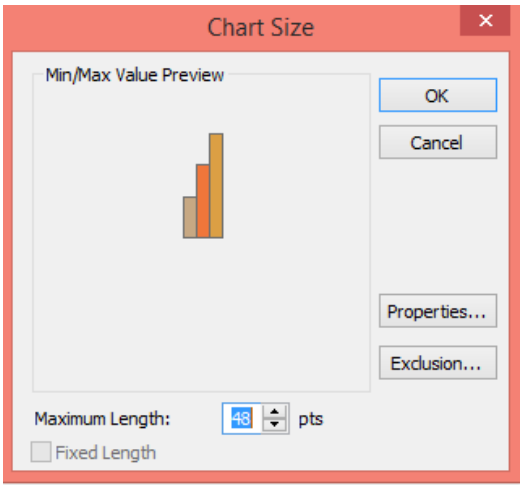
C) Kliknutím na tlačítko *Properties…* otevře další nabídku, kde lze nastavit například šířka každého sloupce, nastavení osy X a Y, výše zmiňovaná orientace nebo zobrazení ve 3D.



Obr. 43 Nabídka *Properties…* navržena pro tvorbu sloupcového kartodiagramu.

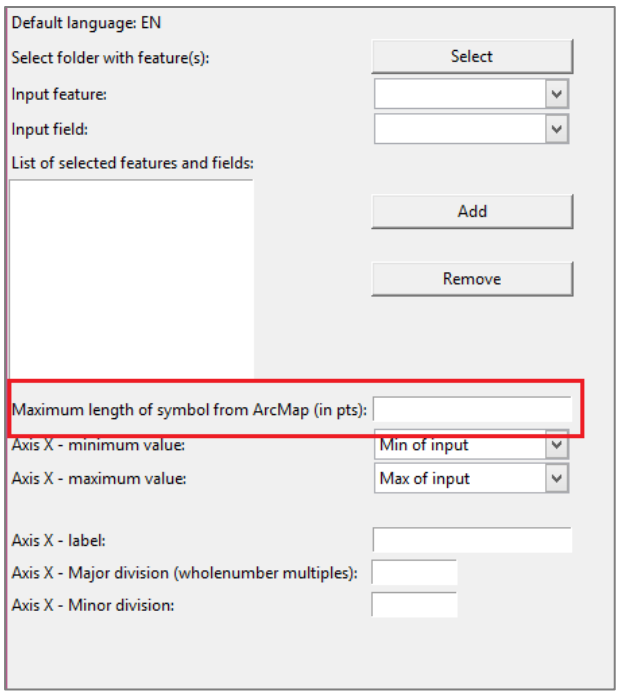
D) Tlačítko *Exclusion…* slouží pro definování slastního SQL.

E) Po kliknutí na tlačítko *Size…* se otevře dialogové okno, které dovoluje uživateli nastavit maximální délku sloupce v jednotce Points.



Obr. 44 Nabídka *Size…* vytvořená pro tvorbu sloupcového kartodiagramu v softwaru ArcGIS.

Tvorba hodnotového měřítka pro tento kartodiagram probíhá téměř stejným způsobem jako při volbě Pie chart. Nejprve si uživatel vytvoří hodnotové měřítko v navrženém programu. Nabídka parametrů je velmi podobná nabídce pro Pie chart s tím rozdílem, že si uživatel nevybírá funkci, podle které se bude hodnotové měřítko vytvářet a místo nejmenší velikosti symbolu v pts. zadává uživatel naopak *Maximální délku symbolu z prostředí ArcMap*, samozřejmě také v jednotce pts (viz obr. 45).



Obr. 45 Výřez levé strany z vlastního programu pro vytváření hodnotového měřítka sloupcového kartodiagramu.

Funkce se nevybírá, jelikož byl stanoven pouze jeden vzorec, podle kterého se vytvoří celé hodnotové měřítko a tento vzorec vypadá následovně:

$$Pj = \frac{Valj}{Valmax} * Pmax$$

Pj……velikost j-tého objektu
Valj…….hodnota j-tého objektu
Valmax……maximální hodnota v rámci využitého výběru dat pro tvorbu kartodiagramu
Pmax……maximální délka kartodiagramu (zadávána ručně z prostředí Arcmap)

Dále už tvorba hodnotového měřítka kartodiagramu probíhá opět stejným způsobem jako u Pie chartu a Proportional symbol, která opět končí uložením výsledku tvorby.

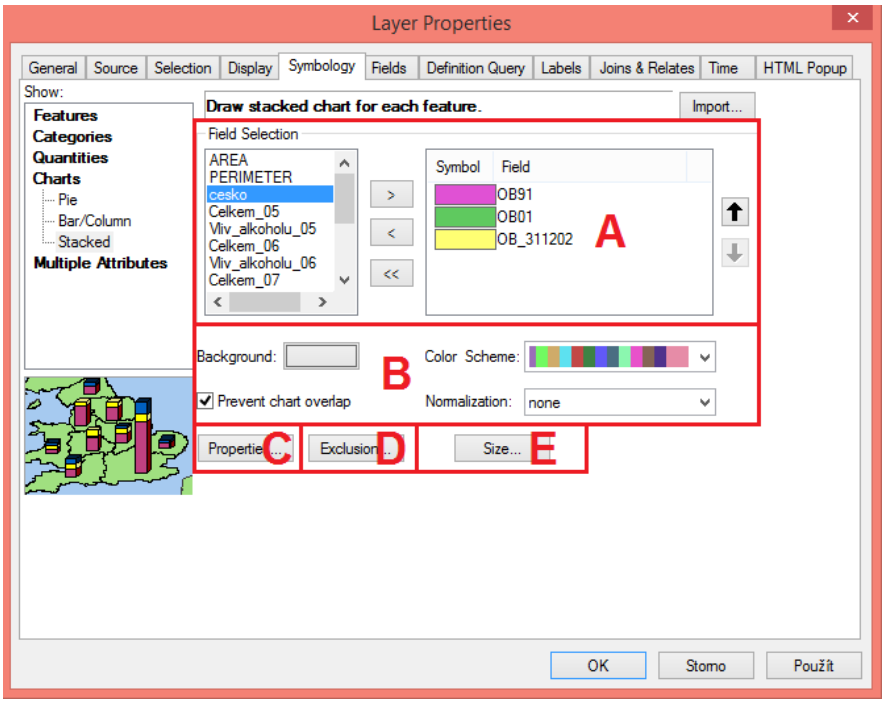
Následuje tvorba sloupcového kartodiagramu v prostředí ArcGIS, kam si uživatel toto hodnotové měřítko vloží do své mapy a vyexportuje svůj vytvořený sloupcový kartodiagram do legendy. Legendu kartodiagramu převede do editačního módu, kde si uživatel musí odmazat hodnotu napsanou u kartodiagramu a všechna další textová označení, která ArcGIS do legendy vyexportuje, až na popis jednotlivých sloupců v kartodiagramu. Tímto posledním krokem je tvorba hodnotového měřítka pro sloupcový kartodiagram ukončena.

5.5.3 Stacked

Poslední kartodiagram, který nabídka Chart v programu ArcGIS poskytuje, je pojmenován názvem Stacked, neboli sloupcový součtový kartodiagram. Jak už název napovídá, jedná se o kartodiagram, jehož hodnoty jevů jsou zobrazeny do jednoho sloupce a výška sloupce tedy udává celkový součet těchto hodnot.

ArcGIS zachovává velmi podobné prostředí pro tvorbu kartodiagramů v nabídce Chart, zde jsou tedy parametry pro tvorbu sloupcového kartodiagramu:

A) volba dat, která mají být vizualizována
B) definování barevného provedení symbolu i pozadí
C) vlastnosti kartodiagramu
D) prostor pro definování vlastního SQL příkazu
E) nastavení velikosti kartodiagramu

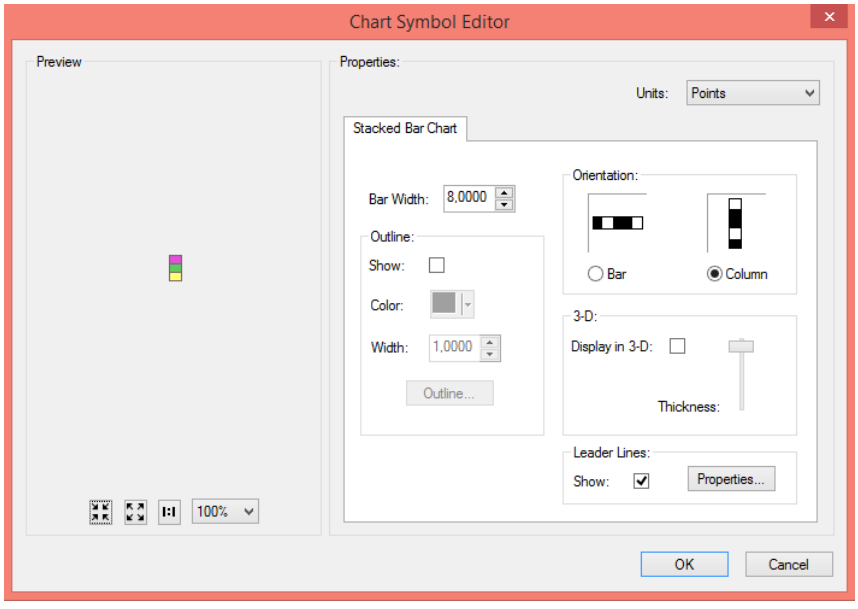


Obr. 46 Grafické znázornění parametrů v nabídce Stacked.

Vysvětlení parametrů:

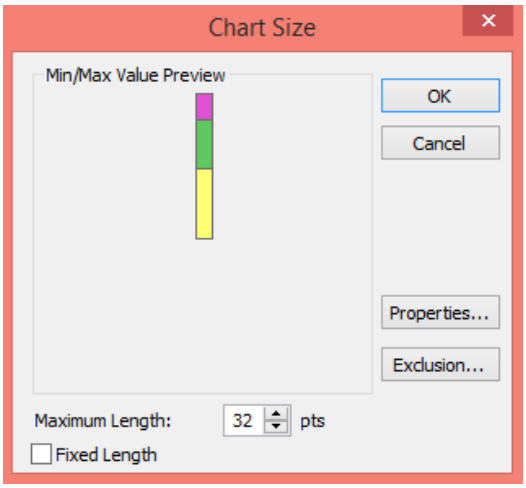
A) Volba načítání dat a B) definování barevného provedení symbolu i pozadí je opět stejná.

C) Kliknutím na tlačítko *Properties…* lze nastavit šířka sloupce, vlastnosti vnější linie, orientace, zobrazení ve 3D nebo jednotky.



Obr. 47 Nabídka *Properties…* u tvorby kartodiagramu Stacked.

E) Nabídka nastavení po kliknutí na tlačítko *Size…* - definování maximální velikosti sloupce



Obr. 48 Definování velikosti sloupcového součtového kartodiagramu.

Tento případ je stejný jako tvorba sloupcového kartodiagramu a jeho hodnotového měřítka. I zde se postupuje podle kroků popsaných u typu Bar/Column chart.

6 VÝSLEDKY

Výsledkem teoretické části bylo sepsání teorie o tom, co diagramy znamenají, do jakých druhů se dělí a podle jakých parametrů, jak se používají v mapách, jak se dají vypočítat a jak tyto kartodiagramy vysvětlit v legendě neboli jak sestavit hodnotová měřítka pro jednotlivé typy kartodiagramů.

Dalším dílčím výsledkem v teoretické části diplomové práce bylo představení a zhodnocení vybraných nástrojů, u kterých bylo popsáno, které kartodiagramy a jejich hodnotová měřítka dokáží vytvořit. Tato analýza byla pro větší přehlednost vizualizována pomocí tabulek.

Prvním výsledkem praktické části byl pro automatizovanou tvorbu hodnotových měřítek kartodiagramů zvolen nástroj a prostředí Python Toolbox. Toto prostředí je velmi hojně využíváno v samotném produktu ArcGIS for Desktop, pro kterou má být vlastní nástroj vytvořen jakožto určitý druh nadstavby. Po pár měsících programování v Python Toolboxu, definování potřebných parametrů pro tvorbu hodnotových měřítek, sestrojení výpočetního systému, byla tato činnost ukončena z důvodu nemožné realizace přidání vlastního okna pro náhled hodnotového měřítka při samotné tvorbě tohoto měřítka. Bližší popsání problematiky bylo sepsáno v kapitole 4 Automatizace tvorby hodnotových měřítek kartodiagramů – Python Toolbox.

Po hledání dalšího prostředí, které by splňovalo požadavky programování v jazyce Python a možnost vlastního okna pro vykreslování grafů fungujícího v jednom okně společně s ostatními parametry, bylo nakonec vybráno prostředí Tkinter. Tento modul má přístup ke knihovně Tk v programovém jazyku Python a umožnuje uživateli vytvářet grafické uživatelské rozhraní. Pomocí tohoto modulu bylo vytvořeno celkem 5 programů (1 hlavní program a 4 podprogramy), které jsou hlavním výsledkem praktické části této diplomové práce:

1. Program tzv. „rozcestník“ (kapitola 5 Automatizace tvorby hodnotových měřítek kartodiagramů – prostředí Tkinter), kde byla naprogramována pouze hlavní nabídka a její možné varianty pro tvorbu konkrétních typů kartodiagramu. Byly použity screenshoty z nabídky v prostředí ArcGIS, aby se uživatel rychleji a snadněji orientoval v příslušné nabídce v programu. Jakmile si uživatel označil typ, pro který chce hodnotové měřítko vytvořit a klikl na tlačítko *Create scale*, otevřelo se speciálně naprogramované prostředí pro tvorbu hodnotového měřítka pro daný typ kartodiagramu.

2. Program Graduated symbols pro bodovou vrstvu (kapitola 5.2 Graduated symbols pro bodovou vrstvu) byl vytvořen jako podprogram pro tzv. „rozcestník“ a specializoval se na tvorbu hodnotového měřítka kartodiagramů s využitím intervalové stupnice, konkrétně pro bodovou vrstvu. Nejdříve byla provedena analýza parametrů v prostředí ArcGIS a parametry, bez kterých by hodnotové měřítko nevzniklo, byly převzaty i do vlastního programu.

3. Program Graduated symbols (kapitola 5.3 Graduated smybols pro liniovou vrstvu) pro liniovou vrstvu byl dalším podprogramem hlavního programu „rozcestníku“, který se také soustředil na tvorbu hodnotového měřítka pro kartodiagramy s využitím intervalové stupnice. Ovšem na rozdíl od předchozího typu byl zaměřen na liniovou vrstvu. Tento podprogram převzal většinu svého kódu od předchozího typu. Prošel procesem zjednodušení potřebných parametrů a výpočetního systému.

4. Program Proportional symbols (kapitola 5.4 Proportional symbols) byl vytvořen jako předposlední podprogram pro „rozcestník“ a byl navržen pro tvorbu hodnotového měřítka pro kartodiagramy využívající funkční stupnici. Tento program byl nejsložitější na realizaci, obsahoval nejvíce parametrů a pracoval jako jediný s knihovnou Arcpy, která zprostředkovávala práci s geografickými daty. Speciálním případem využití tohoto programu bylo použití pro tvorbu hodnotového měřítka pro nabídku Chart (kapitola 5.5 Charts) v prostředí ArcGIS, tedy pro Pie (podkapitola 5.5.1 Pie), jelikož i tento druh kartodiagramu využívá funkční stupnici. Tvorba legendy pro tento případ ovšem musela probíhat částečně v prostředí ArcGIS pro vygenerování symbolu kartodiagramu a vysvětlení jeho částí - jak textové, tak i barevné provedení a tvorba jejich hodnotového měřítka probíhala v tomto programu.

5. Program pro tvorbu kartodiagramů v nabídce Charts – Bar/Column a Stacked (kapitola 5.5.2 Bar/Column a 5.5.3 Stacked) byl naprogramován jako poslední program pro „rozcestník“. Tento program je částečnou obměnou předchozího programu pro tvorbu kartodiagramů s využitím funkční stupnice. Došlo zde k vynechání parametru týkajícího se volby funkce výpočtu velikosti symbolů a upravení jednoho parametru – pro tyto dva případy je od uživatele vyžadována místo minimální velikosti symbolu v produktu ArcGIS ta maximální velikost symbolu. Tvorba legendy probíhá rovněž napůl v prostředí ArcGIS a napůl ve vlastním programu.

7 DISKUZE

Využívání kartodiagramů je velmi rozšířené a mezi uživateli oblíbené jakožto jedna z vyjadřovacích prostředků různých statistických údajů. S rostoucím zájmem o tuto vyjadřovací metodu se zvyšuje i počet softwarů, které tuto metodu mají implementovanou ve svém prostředí a nabízí ji uživateli k tvorbě. Nejvhodnějším prostředím pro tvorbu různých kartodiagramů je pravděpodobně produkt ArcGIS for Desktop, i když ani s jeho pomocí nedokáže uživatel vytvořit všechny typy kartodiagramů. Dochází tedy k naprogramování různých nadstaveb pro tento produkt a tím i snaze rozšířit dostupnou nabídku, aby uživatel při tvorbě map nemusel kombinovat více softwarů dohromady, pokud chce dosáhnout kýženého výsledku a dodržet všechna kartografická pravidla.

Hlavním důvodem pro zadání této diplomové práce byl tedy fakt, že v programu ArcGIS nelze vytvořit kartograficky správné hodnotové měřítko pro funkční stupnice a možnost pro tvorbu intervalových stupnic je omezená. Touto myšlenkou začínala tvorba programu pro Automatizovanou tvorbu hodnotového měřítka kartodiagramů. V prostředí ArcGIS for Desktop byl jako první možnost zvolen Python Toolbox. Problémy s tímto řešením byly popsány v předešlých kapitolách. Byla by to nejjednodušší a nejlepší forma pro vytvoření této nadstavby. Zajímavostí je, že ArcGIS podobné spojení částí („levá část“ pro vyplnění parametrů a „pravá část“ pro náhled grafu) v jednom okně podporuje při volbě na záložce View → Graphs → Create Graph → Create Graph Wizard.

Bylo tedy upuštěno od řešení implementované přímo v produktu ArcGIS. Program v modulu Tkinter pro jazyk Python ovšem není velmi vzdálené řešení. Program byl tvořen již od počátku v anglickém jazyce pro větší obsáhlost skupiny uživatelů. V zadání této práce jazyk definován nebyl a po konzultaci s vedoucím práce byla tedy zvolena angličtina. Do budoucna by nebylo tak obtížné přidat například i českou translaci programu. Uživatel by měl v hlavním programu i následujících podprogramech vždy možnost výběru jazyků, které jsou podporovány, a po kliknutí na určitý jazyk by se celý program do tohoto jazyku přepnul.

Při tvorbě programu pro tvorbu hodnotového měřítka kartodiagramů s využitím funkční stupnice bylo nejprve zamýšleno vykreslit do průběhu funkce i konkrétní tvary symbolů kartodiagramů, aby se dosáhlo větší názornosti a spojitosti s mapou. Při samotné realizaci ovšem došlo na problém poměru velikosti plochy neboli okna pro vykreslení celého hodnotového měřítka a rozsahu hodnot na jednotlivých osách grafu. Jestliže byl poměr okna do obdélníkového tvaru, tedy více do šířky než délky, ale osy X a Y dosahovaly stejných hodnot od 0 do 100, docházelo k roztažení symbolů do šířky a z tvaru kolečka se stala např. elipsa protažená horizontálním směrem. Pokud naopak hodnoty na ose X dosahovaly až do řádů statisíců nebo miliónů, tvary symbolů se naopak protahovaly vertikálním směrem a došlo k jejich zploštění. Vzhledem k tomu, že nikdy dopředu není známo, jaké řády hodnot bude vrstva obsahovat a případné zdeformování

tvaru není přípustné, upustilo se od vykreslování konkrétních tvarů symbolů a ponechána byla jen křivka funkce. Do budoucna by se tento problém mohl vyřešit nalezením určitého koeficientu nebo individuálního přepočtu hodnot tak, aby byl zachován tvar symbolů bez ohledu na to, jaké řády hodnot bude vrstva obsahovat.

Dalším vylepšením do budoucna by byla tvorba celé legendy pro Pie, Bar/Column a Stacked chart ve vlastním programu. Legenda pro tyto typy kartodiagramů se skládá z tvorby hodnotového měřítka a dále symbolu s popisem jeho částí, kde je důležité i jeho barevné provedení. Nyní je tvorba celé legendy rozdělená na dvě části, kde se ve vlastním programu vytvoří hodnotové měřítko a z prostředí ArcGIS se dodá část s popisem kartodiagramu. Při zkoušce naprogramovat obě tyto části v prostředí Tkinter vyvstal problém, kdy možnost vykreslení symbolu s jeho popisem by bylo možno vložit vedle hodnotového měřítka do jednoho okna jen v případě umístění do tzv. legendy grafu, kterou knihovna Matplotlib umožňuje naprogramovat. Do legendy je ale možné umístit jen popis jevu a jeho příslušný znak. V našem případě bylo ale zapotřebí vložit do legendy i celý symbol s jeho strukturou a barevným provedením, což pro knihovnu by byl příkaz k vykreslení dalšího samostatného grafu na úrovni grafu pro hodnotové měřítko a ne jen jako pomocný symbol v legendě. Pro vyřešení tohoto problému by byl zapotřebí delší čas na prostudování všech příkazů a programových kódů, které by tohle byly schopny uskutečnit v tomto prostředí možná s nejistým výsledkem.

8 ZÁVĚR

Hlavním cílem magisterské práce byla **automatizace tvorby hodnotových měřítek kartodiagramů v prostředí ArcGIS for Desktop**.

Realizace hlavního cíle práce proběhla splněním dílčích cílů práce, které byly stanoveny pro teoretickou i praktickou část práce. V teoretické části práce byly stanoveny dílčí cíle **podrobná rešerše literatury** (české i zahraniční) věnující se problematice kartodiagramů se zaměřením na tvorbu příslušných hodnotových měřítek pro různé typy kartodiagramů a **ověření funkcionality** dostupných nástrojů na tvorbu různých typů kartodiagramů a k tomu odpovídajících hodnotových měřítek (v různých softwarových nástrojích). Podrobná rešerše literatury věnující se problematice kartodiagramů se zaměřením na tvorbu příslušných hodnotových měřítek pro různé typy kartodiagramů je popsána v kapitole *3 Současný stav řešené problematiky* a v kapitole *3.2 Nástroje pro tvorbu kartodiagramů a hodnotových měřítek* je popsáno hodnocení funkcionality dostupných nástrojů na tvorbu různých typů kartodiagramů a k tomu odpovídajících hodnotových měřítek.

Na základě získaných informací bylo v praktické části diplomové práce cílem vytvořit **aplikaci a návod** (toolbox nebo jiný vhodný nástroj), jakým bude možné **vytvářet hodnotová měřítka** pro různé typy kartodiagramů realizovaných v softwaru ArcGIS for Desktop, a to s takovým výstupem, který bude dále využitelný (volba velikosti měřítka, fontu popisu, export do křivkového formátu apod.). Tento návod s popisem aplikace je popsán v kapitole *5 Automatizace tvorby hodnotových měřítek kartodiagramů – prostředí Tkinter*. Po naprogramování aplikace byla ověřena její funkcionalita na **ukázkových příkladech** (viz přílohy).

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

[1]: VOŽENÍLEK, V., KAŇOK, J. a kol. *Metody tematické kartografie* - Vizualizace prostorových jevů. Univerzita Palackého v Olomouci, 2011, 216 s. ISBN 9788024427904

[2]: KAŇOK J. *Tematická kartografie*. Ostrava: Ostravská univerzita v Ostravě, 1999. 318 s. ISBN 80-7042-781-7.

[3]: Proportional symbol maps. *Proportional Symbol Maps - indiemapper.com* [online]. 2010 [cit. 2015-10-16]. Dostupné z: http://indiemapper.com/app/learnmore.php?l=proportional_symbols

[4]: VALENT, Tomáš. 2010. *PROGRAMOVÁNÍ NADSTAVBY PRO TVORBU KARTODIAGRAMŮ V ARCGIS* [online]. Olomouc [cit. 2015-10-18]. Dostupné z: http://www.geoinformatics.upol.cz/dprace/magisterske/valent10/index.php?page=stazeni. Diplomová. Univerzita Palackého. Vedoucí práce Ing. Zdena Dobešová PhD.

[5]: Vítězné strany a zisk mandátů podle stran ve volbách do Poslanecké sněmovny podle krajů ve dnech 25. 26. října 2013. 2013. *394588d5-8eaa-4d85-a6a5-5bb68925223d (1200x848)* [online]. ČSÚ, Praha [cit. 2015-10-18]. Dostupné z: https://www.czso.cz/documents/11248/17856641/2013_PS_mandat_kraje.jpg/394588d5-8eaa-4d85-a6a5-5bb68925223d

[6]: Kartodiagramy. *Is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=60031* [online]. Brno [cit. 2015-10-18]. Dostupné z: http://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=60031

[7]: 07_TK_Kartodiagramy_liniove. 2012. *07_TK_Kartodiagramy_liniove.pdf* [online]. Košice [cit. 2015-10-18]. Dostupné z: http://kosice.upjs.sk/~michal.gallay/tk/07_TK_Kartodiagramy_liniove.pdf

[8]: State of the Presidental Race. 2007. *State_cropped_2002 (2357x2021)* [online]. York University [cit. 2015-10-18]. Dostupné z: http://www.datavis.ca/gallery/images/state_cropped_2002.jpg

[9]: AutoCad Map 3D. *AutoCad Map 3D* [online]. Praha, 2015 [cit. 2015-10-26]. Dostupné z: http://www.cadstudio.cz/map

[10]: GRASS GIS - Home. *GRASS GIS - Home* [online]. United States: GRASS Development team, 1998 [cit. 2016-08-01]. Dostupné z: https://grass.osgeo.org/

[11]: JanMap. *JANITOR - systém pro analýzu a syntézu dat* [online]. 2005 [cit. 2015-10-26]. Dostupné z: http://janitor.cenia.cz/www/j2_html.php?id=11&lang=cze&idmn=27

[12]: Intergraph (GeoMedia Viewer - FREE GIS!). *Intergraph (GeoMedia Viewer - FREE GIS!) | Domorodé mapování sítě webových stránek* [online]. 2005 [cit. 2015-10-26]. Dostupné z: http://nativemaps.org/node/1499

[13]: Kristýna-GIS. *Kristýna-GIS* [online]. 2015 [cit. 2015-10-26]. Dostupné z: http://www.christine-gis.com/cz/

[14]: R projekt v ČR: Co je R? *R projekt pro statistické výpočty v České republice Co je R?* [online]. ČR: The R Foundation, 2003 [cit. 2016-08-01]. Dostupné z: http://www.r-project.cz/about.html

[15]: Proportional Symbol Mapping in R. *Journal of Statistical Software* [online]. 2006, **2006**(15), 7 [cit. 2016-08-01]. Dostupné z: https://www.jstatsoft.org/article/view/v015i05/v15i05.pdf

[16]: Spatial Data visualization with R. *R-bloggers|R news and tutorials contributed by (573) R bloggers*[online]. US: R-bloggers, 2012 [cit. 2016-08-01]. Dostupné z: https://www.r-bloggers.com/spatial-data-visualization-with-r/

PŘÍLOHY

SEZNAM PŘÍLOH

Vázané přílohy

Volné přílohy

PŘÍLOHA 1

Návod na instalaci programu a technické parametry pro spuštění programu:

Program byl testován na:

- Windows 7 v kombinaci s ArcGIS for Desktop verze 10.2 a trial verze 10.4

- Windows 8.1 v kombinaci s ArcGIS for Desktop verze 10.2 a trial verze 10.4

Postup:

Z balíčku programu překopírovat do složky Site-packages pro knihovny jazyka Python pro ArcGIS (složka Lib) knihovnu PIL – nezbytné.

(Pro větší bezpečnost nakopírovat obsah celé složky Site-packages z balíčku programu do složky Site-packages na uživatelském počítači).

PŘÍLOHA 2

```
# encoding: utf-8

import Tkinter
from Tkinter import *
tk = Tkinter
from PIL import Image, ImageTk
import os

class Application(tk.Frame):

     button_id = None

     def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.toberemoved = []
        self.createWidgets()

     def del_descriptions(self):
        for toberemoved in self.toberemoved:
            toberemoved.destroy()
        self.toberemoved = []

     def OnButtonClick(self, button_id):
         master = self
         if button_id == 1:
            self.button_id = 1
            self.del_descriptions()

            self.variable = IntVar()
            self.b1 = Radiobutton(master, text='Graduated symbols', value = 1,
                  variable = self.variable)
            self.b1.grid(row=7, sticky=W)

            self.b2 = Radiobutton(master, text='Proportional symbols', value = 2,
                  variable = self.variable)
            self.b2.grid(row=8, sticky=W)

            if self.my_var.get()==1:
```

```
            fn1 = os.path.join(os.path.dirname(__file__), 'point_quantities.png')
            image1 = Image.open(fn1)
            image1 = image1.resize((500, 400), Image.ANTIALIAS)
            photo1 = ImageTk.PhotoImage(image1)
            panel1 = Label(master, image = photo1)
            panel1.grid(row=7, rowspan = 10, column=1, sticky=W)
            panel1.image = photo1
            self.toberemoved.extend([panel1])
        elif self.my_var.get()==2:
            fn2 = os.path.join(os.path.dirname(__file__), 'line_quantities.png')
            image2 = Image.open(fn2)
            image2 = image2.resize((500, 400), Image.ANTIALIAS)
            photo2 = ImageTk.PhotoImage(image2)
            panel2 = Label(master, image = photo2)
            panel2.grid(row=7, rowspan = 10, column=1, sticky=W)
            panel2.image = photo2
            self.toberemoved.extend([panel2])

        self.toberemoved.extend([self.b1, self.b2])

    elif button_id == 2:
        self.button_id = 2
        self.del_descriptions()

        self.variable2 = IntVar()
        self.b3 = Radiobutton(master, text='Pie', value = 3,
              variable = self.variable2)
        self.b3.grid(row=7, sticky=W)

        self.b4 = Radiobutton(master, text='Bar', value = 4,
              variable = self.variable2)
        self.b4.grid(row=8, sticky=W)

        self.b5 = Radiobutton(master, text='Stacked', value = 5,
              variable = self.variable2)
        self.b5.grid(row=9, sticky=W)

        fn3 = os.path.join(os.path.dirname(__file__), 'charts.png')
        image3 = Image.open(fn3)
```

```
            image3 = image3.resize((500, 400), Image.ANTIALIAS)
            photo3 = ImageTk.PhotoImage(image3)
            panel3 = Label(master, image = photo3)
            panel3.grid(row=7, rowspan = 10, column=1, sticky=W)
            panel3.image = photo3

            self.toberemoved.extend([self.b3, self.b4, self.b5, panel3])

    def createWidgets(self):
        master = self
        Label(master, text="Default language: EN").grid(row=0, sticky=W)
        Label(master, text="Cartodiagram scales creation for:",
              font=("Arial",12,"bold")).grid(row=1,sticky=W)
        Label(master, text="Type of cartodiagram creation:",
              font=("Arial",12,"bold")).grid(row=4, sticky=W)

        self.my_var = IntVar()

        self.rb1 = Radiobutton(master, value = 1, variable = self.my_var,text="POINT,
                   AREA")
        self.rb1.grid(row=2, sticky=W)


        self.rb2 = Radiobutton(master, value = 2, variable = self.my_var,text="LINE")
        self.rb2.grid(row=2, column = 1, sticky=W)


        Label(master, text=
"_____
_____").grid(row=3,
columnspan = 2, sticky=W)


        self.v = IntVar()
        self.rb3 = Radiobutton(master, value = 1, variable = self.v,text="Quantities",
                   command=lambda: self.OnButtonClick(1))
        self.rb3.grid(row=5, sticky=W)

        self.rb4 = Radiobutton(master, value = 2, variable = self.v,text="Charts",
                   command=lambda: self.OnButtonClick(2))
```

```
        self.rb4.grid(row=5, column = 1, sticky=W)


        Label(master,
text="_____
_____").grid(row=6,
columnspan = 2, sticky=W)


        self.button3 = Button (master, text='Create scale', width=20,
command=self.open_script)
        self.button3.grid(row=13)


        self.button4 = Button(master, text='Close window', width=20,
command=self.close_window)
        self.button4.grid (row=14)


    def open_script(self):


        if self.my_var.get()==1:    # choices for point
           if self.button_id==1:
               if self.variable.get()==1:   # choices for graduated symbols for point
                   fn1 = os.path.join(os.path.dirname(__file__),
                         'graduated_interval_point_final.py')
                   os.system(fn1)
               elif self.variable.get()==2:   # choices for proportional symbols for
                                                point
                   fn2 = os.path.join(os.path.dirname(__file__),
                         'proportional_charts_functional_final.py')
                   os.system(fn2)
           else:
               if self.variable2.get()==3:     # choices for chart - Pie for point
                   fn3 = os.path.join(os.path.dirname(__file__),
                         'proportional_charts_functional_final.py')
                   os.system(fn3)
               elif self.variable2.get()==4:     # choices for chart - Bar for point
                   fn4 = os.path.join(os.path.dirname(__file__),
                         'proportional_column_stacked_functional_final.py')
                   os.system(fn4)
               elif self.variable2.get()==5:     # choices for chart - Stacked for
                                                   point
                   fn5 = os.path.join(os.path.dirname(__file__),
                         'proportional_column_stacked_functional_final.py')
```

```
                    os.system(fn5)

        elif self.my_var.get()==2:     # choices for line

            if self.button_id==1:
                if self.variable.get()==1:    # choices for graduated symbols for line
                    fn1 = os.path.join(os.path.dirname(__file__),
                          'graduated_interval_line_final.py')
                    os.system(fn1)
                elif self.variable.get()==2:  # choices for proportional symbols for
                                                line
                    fn2 = os.path.join(os.path.dirname(__file__),
                          'proportional_charts_functional_final.py')
                    os.system(fn2)
            else:
                if self.variable2.get()==3:     # choices for chart - Pie for line
                    fn3 = os.path.join(os.path.dirname(__file__),
                          'proportional_charts_functional_final.py')
                    os.system(fn3)
                elif self.variable2.get()==4:     # choices for chart - Bar for line
                    fn4 = os.path.join(os.path.dirname(__file__),
                          'proportional_column_stacked_functional_final.py')
                    os.system(fn4)
                elif self.variable2.get()==5:  # choices for chart - Stacked for point
                    fn5 = os.path.join(os.path.dirname(__file__),
                          'proportional_column_stacked_functional_final.py')
                    os.system(fn5)

    def close_window(self):
        root.destroy()



root = tk.Tk()
app = Application(master=root)
app.master.title(u'Automation of Cartodiagram scales creation in ArcGIS for Desktop')
app.mainloop()
#root.destroy()
```

PŘÍLOHA 3

```
# encoding: utf-8

from Tkinter import *
import Tkinter
tk = Tkinter
import tkFileDialog
from tkFileDialog import askopenfilename
from tkColorChooser import askcolor
import tkMessageBox
import ttk
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.patches as patches
import matplotlib.font_manager as font_manager
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from matplotlib.text import Text
from matplotlib.text import Annotation
import pylab
import os
import sys
import math

class Application(tk.Frame):
    chart = None

    def create_chart(self, root):
        f = Figure(figsize=(4.5, 3), dpi=100)
        #canvas.resize(w,h)
        self.figure = f
        # a tk.DrawingArea
        canvas = FigureCanvasTkAgg(f, master=root)
        return canvas

    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.toberemoved = []
        self.grid()
        self.createWidgets()

    def clear_figure(self, widget):
        self.figure.clf()

    def add_first_description(self, i):
       class1 = Label(self, text="%i." % (i+1))
       class1.grid(row=i+7, sticky=W)
       class1a = Label(self, text="(eg. Less than 150, < 150)")
       class1a.grid(row=i+7, column=1)
       entry_text = Entry(self, width=10)
       entry_text.insert(END, "Less than ")
       entry_text.grid(row = i+7, padx=18, sticky=W)
       entry = Entry(self, width=7)
       entry.grid(row= i+7, padx = 100)
       self.toberemoved.extend([class1, class1a, entry_text, entry])
       return entry_text, entry

    def add_from_to_description(self, i):
       class2 = Label(self, text="%i." % (i+1))
       class2.grid(row=i+7, sticky=W)
       class2a = Label(self, text="From:")
       class2a.grid(row=i+7, padx = 18, sticky=W)
       class2b = Label(self, text="To:")
       class2b.grid(row=i+7, column=1, sticky=W)
       entrya = Entry(self, width=18)
       entrya.grid(row=i+7, padx = 47)
       entryb = Entry(self, width=18)
       entryb.grid(row=i+7, column=1, padx = 22)
       self.toberemoved.extend([class2, class2a, class2b, entrya, entryb])
       return entrya, entryb

    def add_last_description(self, i):
       classLast = Label(self, text="%i." % (i+1))
```

```
    classLast.grid(row=i+7, sticky=W)
    classLastA = Label(self, text="(eg. More than 150, > 150)")
    classLastA.grid(row=i+7, column=1)
    entry_text = Entry(self, width=10)
    entry_text.insert(END, "More than ")
    entry_text.grid(row = i+7, padx=18, sticky=W)
    entry = Entry(self, width=7)
    entry.grid(row= i+7, padx = 100)
    self.toberemoved.extend([classLast, classLastA, entry_text, entry])
    return entry_text, entry

def del_descriptions(self):
    for toberemoved in self.toberemoved:
        toberemoved.destroy()
    self.toberemoved = []

def add_manual_size(self, number_of_classes):
    self.size_entry = []
    for i in range(number_of_classes):
        size_name=Label(self, text="Symbol size in pt.:")
        size_name.grid(row=6, column=2)
        size_entry = Entry(self, width=5)
        size_entry.grid(row= i+7, column = 2, padx = 30, sticky = W)
        self.toberemoved.extend([size_name, size_entry])
        self.size_entry.append(size_entry)

def add_descriptions(self, number_of_classes=3):
    self.del_descriptions()
    self.class1_entry_text, self.class1_entry = self.add_first_description(0)
    for i in range(number_of_classes-2):
        entrya, entryb = self.add_from_to_description(i+1)
        namea = "class%ia_entry" % (i+2)
        setattr(self, namea, entrya)
        nameb = "class%ib_entry" % (i+2)
        setattr(self, nameb, entryb)
    self.classLast_entry_text, self.classLast_entry =
        self.add_last_description(number_of_classes-1)

def toggle(self, event):
    master=self
    Label1 = Label(master, text="Symbol size from:")
    Label1.grid(row=4, sticky=W)
    Label2 = Label(master, text="Symbol size to:")
    Label2.grid(row=5, sticky=W)
    self.symbol_size_from = Entry(master, width=24)
    self.symbol_size_from.grid(row= 4, column=1)
    widget = self.symbol_size_from
    self.symbol_size_to = Entry(master, width=24)
    self.symbol_size_to.grid(row= 5, column=1)
    widget2 = self.symbol_size_to
    Label3= Label(master, text="Write description of classes:")
    Label3.grid(row=6, sticky=W)

    if self.boxmethod.get() in ('Automatically'):
        Label1.grid()
        Label2.grid()
        Label3.grid()
        widget.grid()
        widget2.grid()
        self.del_descriptions()
        number_of_classes = int(self.box3.get())
        self.add_descriptions(number_of_classes)
    else:
        Label1.config(fg = 'grey')
        Label2.config(fg = 'grey')
        Label3.grid()
        widget.config(state=DISABLED)
        widget2.config(state=DISABLED)
        self.del_descriptions()
        number_of_classes = int(self.box3.get())
        self.add_descriptions(number_of_classes)
        self.add_manual_size(number_of_classes)
```

```
def savetofolder(self):
    self.saveoutput = tkFileDialog.asksaveasfilename(parent=root)
    if self.saveoutput is None:
       return
    self.label_savefolder.delete(0,END)
    self.label_savefolder.insert(0,self.saveoutput)

def createWidgets(self):
    #import pdb;pdb.set_trace()
    master = self
    Label(master, text="Default language: EN").grid(row=0, sticky=W)
    Label(master, text="Current symbol").grid(row=1, sticky=W)
    Label(master, text="Number of classes:").grid(row=2, sticky=W)
    Label(master, text="Classification Method:").grid(row=3, sticky=W)
    Label(master, text="Font family used in graph:").grid(row=16, column=3, padx=30,
    sticky=W)
    Label(master, text="Font size used in graph:").grid(row=17, column=3, padx=30,
    sticky=W)
    Label(master, text="Output format:").grid(row=19, column=3, padx=30, sticky=W)

    value = StringVar()
    self.box2 = ttk.Combobox(master,
      values = ['Circle 1', 'Square 1', 'Triangle 1', 'Pentagon 1', 'Hexagon 1']
      )
    self.box2.state(['readonly'])
    self.box2.bind('<<ComboboxSelected>>')
    self.box2.current(0)
    self.box2.grid(row = 1, column = 1)

    value = StringVar()
    self.box3 = ttk.Combobox(master,
      values = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
      )
    self.box3.state(['readonly'])
    self.box3.bind('<<ComboboxSelected>>',self.toggle)
    self.box3.current(0)
    self.box3.grid(row = 2, column = 1)

    value = StringVar()
    self.boxmethod = ttk.Combobox(master, values = ['Automatically','Manually'])
    self.boxmethod.state(['readonly'])
    self.boxmethod.bind('<<ComboboxSelected>>', self.toggle)
    self.boxmethod.current(0)
    self.boxmethod.grid(row = 3, column = 1)

    self.redraw = Button(master, text="Refresh graph", width=20, command = self.readshape)
    self.redraw.grid(row=15,column=3, padx=30)

    value = StringVar()
    self.box4 = ttk.Combobox(master,
      values = ['Arial','Calibri','Verdana','Times New Roman']
      )
    self.box4.state(['readonly'])
    self.box4.bind('<<ComboboxSelected>>')
    self.box4.current(0)
    self.box4.grid(row = 16, column = 3, padx=160)

    value = StringVar()
    self.box5 = ttk.Combobox(master,
      values = ['8', '9', '10', '11', '12', '15']
      )
    self.box5.state(['readonly'])
    self.box5.bind('<<ComboboxSelected>>')
    self.box5.current(0)
    self.box5.grid(row = 17, column = 3, padx=160)

    self.chart = self.create_chart(master)
    self.chart._tkcanvas.grid(row=0, column=3, rowspan=14)

    self.outputboxvalue = StringVar()
    self.outputbox = ttk.Combobox(self, values=["JPG","PNG","TIFF","SVG"],
        )
    self.outputbox.state(['readonly'])
    self.outputbox.bind('<<ComboboxSelected>>')
```

```
        self.outputbox.current(0)
        self.outputbox.grid(row=19, column=3, padx=160)

        self.label_savefoldervalue = StringVar()
        self.label_savefolder= Entry(master, width=45, textvariable =
                                     self.label_savefoldervalue)
        self.label_savefolder.grid(row= 20, column=3, padx=30, sticky=W)

        self.saveto=Button(master, text="Select output folder", width=20,
                                     command=self.savetofolder)
        self.saveto.grid(row= 20, column=3, padx=260, sticky=W)

        self.button = Button (master, text='Save graph', width=20, command=self.save_figure)
        self.button.grid(row=22, column=3)

        self.button2 = Button(master, text='Close window', width=20, command=self.close_window)
        self.button2.grid (row=23, column=3)

    def get_symbol_size_from(self):
        try:
            return self.symbol_size_from.get()
        except:
            return self.symbol_size_from
        pass

    def get_symbol_size_to(self):
        try:
            return self.symbol_size_to.get()
        except:
            return self.symbol_size_to
        pass

    def get_savegraphtofolder(self):
        try:
            return self.label_savefolder.get()
        except:
            return self.savegraphtofolder
        pass

    def get_manual_size_symbol(self):
        try:
            return self.size_entry.get()
        except:
            return self.manual_size_symbol
        pass

    def readshape(self):
        self.figure.clf()
        fig = self.figure
        ax1 = fig.add_subplot(111)
        self.axis = ax1

        title_font = {'fontname':self.box4.get(), 'size':self.box5.get(), 'color':'black',
                      'weight':'normal','verticalalignment':'bottom'}
        axis_font = {'fontname':self.box4.get(), 'size':self.box5.get()}

        # Set the font properties (for use in legend)
        font_path = 'C:\Windows\Fonts\Arial.ttf'
        font_prop = font_manager.FontProperties(fname=font_path, size=14)

        # Set the tick labels font
        for label in (ax1.get_xticklabels() + ax1.get_yticklabels()):
            label.set_fontname(self.box4.get())
            label.set_fontsize(self.box5.get())

        ax1.set_ylim(-1, 272)
        ax1.set_xlim(0, 386)
        ax1.yaxis.set_visible(False)

        ax1.spines['right'].set_visible(False)
        ax1.spines['top'].set_visible(False)
        ax1.spines['left'].set_visible(False)
        ax1.spines['bottom'].set_visible(False)
```

```
ax1.tick_params(left='off', top='off', right='off', bottom='off',
                labelleft='off', labeltop='off', labelright='off', labelbottom='off')


number_of_classes = int(self.box3.get())

if self.boxmethod.get() in ('Automatically'):

    #####Podminky na MinSymbol a MaxSymbol####
    try:
      MinSymbol = float(self.symbol_size_from.get())
    except:
       messagetxt = "Please insert value to field - 'Symbol size from'."
       message = tkMessageBox.showerror("Warning", messagetxt)
    try:
      MaxSymbol = float(self.symbol_size_to.get())
    except:
       messagetxt = "Please insert value to field - 'Symbol size to'."
       message = tkMessageBox.showerror("Warning", messagetxt)

    MaxSymbol = float(self.symbol_size_to.get())
    if MaxSymbol == 0:
       messagetxt = "Field - 'Symbol size to' can´t be 0. Please insert another value."
       message = tkMessageBox.showerror("Warning", messagetxt)
    if MaxSymbol < MinSymbol:
       messagetxt = "Field - 'Symbol size to' must be higher than field - 'Symbol size
                     from'. Please insert another value."
       message = tkMessageBox.showerror("Warning", messagetxt)
    ######

    deltay = MaxSymbol - MinSymbol   #rozsah hodnot na ose Y
    pocet = float(self.box3.get())   #pocet vykreslenych symbolu (=pocet intervalu)
    step = deltay/(pocet - 1)   (stred a max)

    #cx = souradnice stredu elipsy na ose X
    #cy = odpovidajici hodnota pro cx na ose Y

    y = MinSymbol
    valueto0 = None
    for i in range(number_of_classes):
          if self.box2.get() in ('Circle 1'):
             cx = 65.0
             cy = y/2.0
             rmax = MaxSymbol/2.0
             circle = plt.Circle((cx, cy), cy, fill=False)
             ax1.add_patch(circle)
             ax1.hlines(y, cx, (cx+rmax)+10, color='black', lw=1)
             if 0<i<number_of_classes-1:
                 entryname_from = "class%ia_entry" % (i+1)
                 valuefrom = float(getattr(self, entryname_from).get())
                 entryname_to = "class%ib_entry" % (i+1)
                 valueto = float(getattr(self, entryname_to).get())
                 title = u'%g-%g' % (valuefrom, valueto)
                 if valueto <= valuefrom:
                        messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                      change your values for the description of class
                                      number %i" % (valueto, valuefrom, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                 elif valuefrom <= valueto0:
                        messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                      please change your values for the description of
                                      class number %i" % (valuefrom, valueto0, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                 valueto0 = valueto
             elif i==0:
                 title = " ".join((self.class1_entry_text.get().rstrip(),
                                   self.class1_entry.get()))
                 valueto0 = float(self.class1_entry.get())
             else:
                 title = " ".join((self.classLast_entry_text.get().rstrip(),
                                   self.classLast_entry.get()))
                 valuefrom = float(self.classLast_entry.get())
```

```
            if valuefrom < valueto0:
                   messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                please change your values." % (valuefrom, valueto0)
                   message = tkMessageBox.showerror("Warning", messagetxt)

        ax1.annotate(title, xy=((cx+rmax+12),y), size = self.box5.get(), family =
                     self.box4.get())

    elif self.box2.get() in ('Square 1'):
        cx = 250.0
        rectangle = plt.Rectangle((cx, 0), width= y, height = y, fill=False)
        ax1.add_patch(rectangle)
        ax1.hlines(y, cx, (cx-15), color='black', lw=1)
        if 0<i<number_of_classes-1:
            entryname_from = "class%ia_entry" % (i+1)
            valuefrom = float(getattr(self, entryname_from).get())
            entryname_to = "class%ib_entry" % (i+1)
            valueto = float(getattr(self, entryname_to).get())
            title = u'%g-%g' % (valuefrom, valueto)
            if valueto <= valuefrom:
                   messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                change your values for the description of class
                                number %i" % (valueto, valuefrom, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            elif valuefrom <= valueto0:
                   messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                please change your values for the description of
                                class number %i" % (valuefrom, valueto0, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            valueto0 = valueto
        elif i==0:
            title = " ".join((self.class1_entry_text.get().rstrip(),
                              self.class1_entry.get()))
            valueto0 = float(self.class1_entry.get())
        else:
            title = " ".join((self.classLast_entry_text.get().rstrip(),
                              self.classLast_entry.get()))
            valuefrom = float(self.classLast_entry.get())
            if valuefrom < valueto0:
                   messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                please change your values." % (valuefrom, valueto0)
                   message = tkMessageBox.showerror("Warning", messagetxt)
        ax1.annotate(title, xy=((cx-17),y), size = self.box5.get(), family =
                                  self.box4.get(), horizontalalignment='right')

    elif self.box2.get() in ('Triangle 1'):
        cx = 70.0
        rmax = MaxSymbol/2
        ro = y/3.0
        r = 2*ro
        cy = ro
        triangle = patches.RegularPolygon((cx, cy), 3, r, fill=False)
        ax1.add_patch(triangle)
        ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
        if 0<i<number_of_classes-1:
            entryname_from = "class%ia_entry" % (i+1)
            valuefrom = float(getattr(self, entryname_from).get())
            entryname_to = "class%ib_entry" % (i+1)
            valueto = float(getattr(self, entryname_to).get())
            title = u'%g-%g' % (valuefrom, valueto)
            if valueto <= valuefrom:
                   messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                change your values for the description of class
                                number %i" % (valueto, valuefrom, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            elif valuefrom <= valueto0:
                   messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                please change your values for the description of
                                class number %i" % (valuefrom, valueto0, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            valueto0 = valueto
        elif i==0:
            title = " ".join((self.class1_entry_text.get().rstrip(),
                              self.class1_entry.get()))
```

```
            valueto0 = float(self.class1_entry.get())
        else:
            title = " ".join((self.classLast_entry_text.get().rstrip(),
                              self.classLast_entry.get()))
            valuefrom = float(self.classLast_entry.get())
            if valuefrom < valueto0:
                   messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                 please change your values." % (valuefrom, valueto0)
                   message = tkMessageBox.showerror("Warning", messagetxt)
        ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                 self.box4.get())

    elif self.box2.get() in ('Pentagon 1'):
        cx = 65.0
        rmax = MaxSymbol/2
        q0 = math.sqrt(50+10*math.sqrt(5))/10
        qv = math.sqrt(25+10*math.sqrt(5))/10
        a = y/(q0+qv)
        r = q0*a
        rv = qv*a
        cy = rv
        pentagon = patches.RegularPolygon((cx, cy), 5, r, fill=False)
        ax1.add_patch(pentagon)
        ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
        if 0<i<number_of_classes-1:
            entryname_from = "class%ia_entry" % (i+1)
            valuefrom = float(getattr(self, entryname_from).get())
            entryname_to = "class%ib_entry" % (i+1)
            valueto = float(getattr(self, entryname_to).get())
            title = u'%g-%g' % (valuefrom, valueto)
            if valueto <= valuefrom:
                   messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                 change your values for the description of class
                                 number %i" % (valueto, valuefrom, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            elif valuefrom <= valueto0:
                   messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                 please change your values for the description of
                                 class number %i" % (valuefrom, valueto0, i+1)
                   message = tkMessageBox.showerror("Warning", messagetxt)
            valueto0 = valueto
        elif i==0:
            title = " ".join((self.class1_entry_text.get().rstrip(),
                              self.class1_entry.get()))
            valueto0 = float(self.class1_entry.get())
        else:
            title = " ".join((self.classLast_entry_text.get().rstrip(),
                               self.classLast_entry.get()))
            valuefrom = float(self.classLast_entry.get())
            if valuefrom < valueto0:
                   messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                 please change your values." % (valuefrom, valueto0)
                   message = tkMessageBox.showerror("Warning", messagetxt)
        ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                 self.box4.get())

    elif self.box2.get() in ('Hexagon 1'):
        cx = 70.0
        rmax = MaxSymbol/2
        rv = y/2.0
        a = 2*rv/math.sqrt(3)
        r = a
        cy = rv
        hexagon = patches.RegularPolygon((cx, cy), 6, r, fill=False, orientation =
                                 math.pi/2.0)
        ax1.add_patch(hexagon)
        ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
        if 0<i<number_of_classes-1:
            entryname_from = "class%ia_entry" % (i+1)
            valuefrom = float(getattr(self, entryname_from).get())
            entryname_to = "class%ib_entry" % (i+1)
            valueto = float(getattr(self, entryname_to).get())
            title = u'%g-%g' % (valuefrom, valueto)
            if valueto <= valuefrom:
```

```
                        messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                     change your values for the description of class
                                     number %i" % (valueto, valuefrom, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                    elif valuefrom <= valueto0:
                        messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                     please change your values for the description of
                                     class number %i" % (valuefrom, valueto0, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                    valueto0 = valueto
                elif i==0:
                    title = " ".join((self.class1_entry_text.get().rstrip(),
                                     self.class1_entry.get()))
                    valueto0 = float(self.class1_entry.get())
                else:
                    title = " ".join((self.classLast_entry_text.get().rstrip(),
                                     self.classLast_entry.get()))
                    valuefrom = float(self.classLast_entry.get())
                    if valuefrom < valueto0:
                        messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                    please change your values." % (valuefrom, valueto0)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                     self.box4.get())
            y += step
    else:
        valueto0 = None
        rmax = max(float(entry.get())/2.0 for entry in self.size_entry )
        for i in range(number_of_classes):
            y = float(self.size_entry[i].get())
            if self.box2.get() in ('Circle 1'):
                cx = 65.0
                cy = y/2.0
                circle = plt.Circle((cx, cy), cy, fill=False)
                ax1.add_patch(circle)
                ax1.hlines(y, cx, (cx+rmax)+10, color='black', lw=1)
                if 0<i<number_of_classes-1:
                    entryname_from = "class%ia_entry" % (i+1)
                    valuefrom = float(getattr(self, entryname_from).get())
                    entryname_to = "class%ib_entry" % (i+1)
                    valueto = float(getattr(self, entryname_to).get())
                    title = u'%g-%g' % (valuefrom, valueto)
                    if valueto <= valuefrom:
                        messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                     change your values for the description of class
                                     number %i" % (valueto, valuefrom, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                    elif valuefrom <= valueto0:
                        messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                     please change your values for the description of
                                     class number %i" % (valuefrom, valueto0, i+1)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                    valueto0 = valueto
                elif i==0:
                    title = " ".join((self.class1_entry_text.get().rstrip(),
                                     self.class1_entry.get()))
                    valueto0 = float(self.class1_entry.get())
                else:
                    title = " ".join((self.classLast_entry_text.get().rstrip(),
                                     self.classLast_entry.get()))
                    valuefrom = float(self.classLast_entry.get())
                    if valuefrom < valueto0:
                        messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                    please change your values." % (valuefrom, valueto0)
                        message = tkMessageBox.showerror("Warning", messagetxt)

                ax1.annotate(title, xy=((cx+rmax+12),y), size = self.box5.get(), family =
                                     self.box4.get())

            elif self.box2.get() in ('Square 1'):
                cx = 250.0
                rectangle = plt.Rectangle((cx, 0), width= y, height = y, fill=False)
                ax1.add_patch(rectangle)
                ax1.hlines(y, cx, (cx-15), color='black', lw=1)
```

```
            if 0<i<number_of_classes-1:
                entryname_from = "class%ia_entry" % (i+1)
                valuefrom = float(getattr(self, entryname_from).get())
                entryname_to = "class%ib_entry" % (i+1)
                valueto = float(getattr(self, entryname_to).get())
                title = u'%g-%g' % (valuefrom, valueto)
                if valueto <= valuefrom:
                       messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                    change your values for the description of class
                                    number %i" % (valueto, valuefrom, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                elif valuefrom <= valueto0:
                       messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                    please change your values for the description of
                                    class number %i" % (valuefrom, valueto0, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                valueto0 = valueto
            elif i==0:
                title = " ".join((self.class1_entry_text.get().rstrip(),
                                    self.class1_entry.get()))
                valueto0 = float(self.class1_entry.get())
            else:
                title = " ".join((self.classLast_entry_text.get().rstrip(),
                                    self.classLast_entry.get()))
                valuefrom = float(self.classLast_entry.get())
                if valuefrom < valueto0:
                       messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                   please change your values." % (valuefrom, valueto0)
                       message = tkMessageBox.showerror("Warning", messagetxt)
            ax1.annotate(title, xy=((cx-17),y), size = self.box5.get(), family =
                                    self.box4.get(), horizontalalignment='right')

        elif self.box2.get() in ('Triangle 1'):
            cx = 70.0
            ro = y/3.0
            r = 2*ro
            cy = ro
            triangle = patches.RegularPolygon((cx, cy), 3, r, fill=False)
            ax1.add_patch(triangle)
            ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
            if 0<i<number_of_classes-1:
                entryname_from = "class%ia_entry" % (i+1)
                valuefrom = float(getattr(self, entryname_from).get())
                entryname_to = "class%ib_entry" % (i+1)
                valueto = float(getattr(self, entryname_to).get())
                title = u'%g-%g' % (valuefrom, valueto)
                if valueto <= valuefrom:
                       messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                    change your values for the description of class
                                    number %i" % (valueto, valuefrom, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                elif valuefrom <= valueto0:
                       messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                    please change your values for the description of
                                    class number %i" % (valuefrom, valueto0, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                valueto0 = valueto
            elif i==0:
                title = " ".join((self.class1_entry_text.get().rstrip(),
                                    self.class1_entry.get()))
                valueto0 = float(self.class1_entry.get())
            else:
                title = " ".join((self.classLast_entry_text.get().rstrip(),
                                    self.classLast_entry.get()))
                valuefrom = float(self.classLast_entry.get())
                if valuefrom < valueto0:
                       messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                   please change your values." % (valuefrom, valueto0)
                       message = tkMessageBox.showerror("Warning", messagetxt)
            ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                    self.box4.get())

        elif self.box2.get() in ('Pentagon 1'):
            cx = 65.0
```

```
            q0 = math.sqrt(50+10*math.sqrt(5))/10
            qv = math.sqrt(25+10*math.sqrt(5))/10
            a = y/(q0+qv)
            r = q0*a
            rv = qv*a
            cy = rv
            pentagon = patches.RegularPolygon((cx, cy), 5, r, fill=False)
            ax1.add_patch(pentagon)
            ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
            if 0<i<number_of_classes-1:
                entryname_from = "class%ia_entry" % (i+1)
                valuefrom = float(getattr(self, entryname_from).get())
                entryname_to = "class%ib_entry" % (i+1)
                valueto = float(getattr(self, entryname_to).get())
                title = u'%g-%g' % (valuefrom, valueto)
                if valueto <= valuefrom:
                       messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                    change your values for the description of class
                                    number %i" % (valueto, valuefrom, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                elif valuefrom <= valueto0:
                       messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                    please change your values for the description of
                                    class number %i" % (valuefrom, valueto0, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                valueto0 = valueto
            elif i==0:
                title = " ".join((self.class1_entry_text.get().rstrip(),
                                  self.class1_entry.get()))
                valueto0 = float(self.class1_entry.get())
            else:
                title = " ".join((self.classLast_entry_text.get().rstrip(),
                                   self.classLast_entry.get()))
                valuefrom = float(self.classLast_entry.get())
                if valuefrom < valueto0:
                       messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                   please change your values." % (valuefrom, valueto0)
                       message = tkMessageBox.showerror("Warning", messagetxt)
            ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                   self.box4.get())

        elif self.box2.get() in ('Hexagon 1'):
            cx = 70.0
            rv = y/2.0
            a = 2*rv/math.sqrt(3)
            r = a
            cy = rv
            hexagon = patches.RegularPolygon((cx, cy), 6, r, fill=False, orientation =
                      math.pi/2.0)
            ax1.add_patch(hexagon)
            ax1.hlines(y, cx, (cx+rmax)+15, color='black', lw=1)
            if 0<i<number_of_classes-1:
                entryname_from = "class%ia_entry" % (i+1)
                valuefrom = float(getattr(self, entryname_from).get())
                entryname_to = "class%ib_entry" % (i+1)
                valueto = float(getattr(self, entryname_to).get())
                title = u'%g-%g' % (valuefrom, valueto)
                if valueto <= valuefrom:
                       messagetxt = "Your value 'To:' %s <= value 'From:' %s, please
                                    change your values for the description of class
                                    number %i" % (valueto, valuefrom, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                elif valuefrom <= valueto0:
                       messagetxt = "Your value 'From:' %s < previous value 'To:' %s,
                                    please change your values for the description of
                                    class number %i" % (valuefrom, valueto0, i+1)
                       message = tkMessageBox.showerror("Warning", messagetxt)
                valueto0 = valueto
            elif i==0:
                title = " ".join((self.class1_entry_text.get().rstrip(),
                                  self.class1_entry.get()))
                valueto0 = float(self.class1_entry.get())
            else:
                title = " ".join((self.classLast_entry_text.get().rstrip(),
```

```
                                        self.classLast_entry.get()))
                        valuefrom = float(self.classLast_entry.get())
                        if valuefrom < valueto0:
                               messagetxt = "Your LAST value %s <  previous value 'To:' %s,
                                           please change your values." % (valuefrom, valueto0)
                               message = tkMessageBox.showerror("Warning", messagetxt)
                    ax1.annotate(title, xy=((cx+rmax+17),y), size = self.box5.get(), family =
                                        self.box4.get())

        if self.chart:
            self.chart.draw()

    def save_figure(self):
        fig = self.figure
        ax1 = fig.add_subplot(111)
        self.axis = ax1
        outputpath = self.saveoutput

        if self.outputbox.get() in ('PNG'):
           suffix = '.png'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as png"
           messagetxt = "Image was saved as png."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('SVG'):
           suffix = '.svg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as svg"
           messagetxt = "Image was saved as svg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('JPG'):
           suffix = '.jpg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as jpg"
           messagetxt = "Image was saved as jpg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        else:
           suffix = '.tiff'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as tiff"
           messagetxt = "Image was saved as tiff."
           message = tkMessageBox.showinfo("Info", messagetxt)

    def close_window(self):
        root.destroy()


root = tk.Tk()
app = Application(master=root)
app.master.title(u'Automation of Cartodiagram scales creation in ArcGIS for Desktop')
app.mainloop()
#root.destroy()
```

PŘÍLOHA 4

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

from Tkinter import *
import Tkinter
tk = Tkinter
import tkFileDialog
from tkFileDialog import askopenfilename
from tkColorChooser import askcolor
import tkMessageBox
import ttk
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.patches as patches
import matplotlib.font_manager as font_manager
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from matplotlib.text import Text
from matplotlib.text import Annotation
import matplotlib.ticker as ticker
from matplotlib.ticker import MultipleLocator, FormatStrFormatter, NullFormatter
from matplotlib.ticker import AutoMinorLocator
import pylab
from PIL import Image, ImageTk
import os
from os import path
import sys
import math

class Application(tk.Frame):
    chart = None
    button_id = None

    def create_chart(self, root):
        f = Figure(figsize=(4.5, 3), dpi=100)
        #canvas.resize(w,h)
        self.figure = f
        # a tk.DrawingArea
        canvas = FigureCanvasTkAgg(f, master=root)
        return canvas

    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.toberemoved = []
        self.grid()
        self.createWidgets()

    def clear_figure(self, widget):
        self.figure.clf()

    def add_first_description_b1(self, i):
       class1 = Label(self, text="(optional minimal value)")
       class1.grid(row=i+14, padx = 100)
       entry1 = Entry(self, width=10)
       entry1.grid(row= i+14, padx = 35, sticky = W)
       self.toberemoved.extend([class1, entry1])
       return entry1

    def add_first_description_b2(self, i):
       class1 = Label(self, text="%i." % (i+1))
       class1.grid(row=i+14, sticky=W)
       class1a = Label(self, text="(eg. Less than 150, < 150)")
       class1a.grid(row=i+14, column=1)
       entry_text = Entry(self, width=10)
       entry_text.insert(END, "Less than ")
       entry_text.grid(row = i+14, padx=18, sticky=W)
       entry = Entry(self, width=7)
       entry.grid(row= i+14, padx = 100)
       self.toberemoved.extend([class1, class1a, entry_text, entry])
       return entry_text, entry

    def add_to_description_b1(self, number_of_classes):
       self.to_entry = []
```

```
    for i in range(number_of_classes):
         to_name=Label(self, text="%i." % (i+1))
         to_name.grid(row=i+15, sticky=W)
         to_name_a = Label(self, text=" *")
         to_name_a.grid(row=i+15, padx = 18, sticky = W)
         to_entry = Entry(self, width=10)
         to_entry.grid(row= i+15, padx = 35, sticky = W)
         self.toberemoved.extend([to_name, to_name_a, to_entry])
         self.to_entry.append(to_entry)

def add_from_to_description_b2(self, i):
    class2 = Label(self, text="%i." % (i+1))
    class2.grid(row=i+14, sticky=W)
    class2a = Label(self, text="From:")
    class2a.grid(row=i+14, padx = 18, sticky=W)
    class2b = Label(self, text="To:")
    class2b.grid(row=i+14, column=1, sticky=W)
    entrya = Entry(self, width=18)
    entrya.grid(row=i+14, padx = 47)
    entryb = Entry(self, width=18)
    entryb.grid(row=i+14, column=1, padx = 22)
    self.toberemoved.extend([class2, class2a, class2b, entrya, entryb])
    return entrya, entryb

def add_last_description_b1(self, i):
    classLast_b1 = Label(self, text="(optional maximal value)")
    classLast_b1.grid(row=i+14, padx = 100)
    entryLast_b1 = Entry(self, width=10)
    entryLast_b1.grid(row= i+14, padx = 35, sticky = W)
    Last_a = Label (self, text="* - required field")
    Last_a.grid(row=i+15, sticky = W)
    self.toberemoved.extend([classLast_b1, entryLast_b1, Last_a])
    return entryLast_b1

def add_last_description_b2(self, i):
    classLast = Label(self, text="%i." % (i+1))
    classLast.grid(row=i+14, sticky=W)
    classLastA = Label(self, text="(eg. More than 150, > 150)")
    classLastA.grid(row=i+14, column=1)
    entry_text = Entry(self, width=10)
    entry_text.insert(END, "More than ")
    entry_text.grid(row = i+14, padx=18, sticky=W)
    entry = Entry(self, width=7)
    entry.grid(row= i+14, padx = 100)
    self.toberemoved.extend([classLast, classLastA, entry_text, entry])
    return entry_text, entry

def del_descriptions(self):
     for toberemoved in self.toberemoved:
         toberemoved.destroy()
     self.toberemoved = []

def add_manual_size(self, number_of_classes):
    self.size_entry = []
    for i in range(number_of_classes):
         size = Label(self, text="Line width in pt.:")
         size.grid(row=13, column=2)
         size_entry = Entry(self, width=5)
         size_entry.grid(row= i+14, column = 2, padx = 50, sticky=W)
         self.toberemoved.extend([size, size_entry])
         self.size_entry.append(size_entry)

def add_descriptions_b2(self, number_of_classes=3):
     self.del_descriptions()
     self.class1_entry_text, self.class1_entry = self.add_first_description_b2(0)
     for i in range(number_of_classes-2):
         entrya, entryb = self.add_from_to_description_b2(i+1)
         namea = "class%ia_entry" % (i+2)
         setattr(self, namea, entrya)
         nameb = "class%ib_entry" % (i+2)
         setattr(self, nameb, entryb)
     self.classLast_entry_text, self.classLast_entry =
                 self.add_last_description_b2(number_of_classes-1)
```

```
def add_descriptions_b1(self, number_of_classes=3):
    self.del_descriptions()
    self.class1_entry = self.add_first_description_b1(0)
    for i in range(number_of_classes-1):
        self.add_to_description_b1(i+1)
    self.classLast_entry = self.add_last_description_b1(number_of_classes)

def OnButtonClick(self, button_id):
    master = self
    if button_id == 1:
        self.button_id = 1
        Label3= Label(master, text="Write limit values:     ")
        Label3.grid(row=13, sticky=W)
        self.del_descriptions()
        number_of_classes = int(self.box3.get())
        self.add_descriptions_b1(number_of_classes)
    elif button_id == 2:
        self.button_id = 2
        Label4= Label(master, text="Write interval values:")
        Label4.grid(row=13, sticky=W)
        self.del_descriptions()
        number_of_classes = int(self.box3.get())
        self.add_descriptions_b2(number_of_classes)

def toggle(self, event):
    master=self
    Label1 = Label(master, text="Linewidth from:")
    Label1.grid(row=10, sticky=W)
    Label2 = Label(master, text="Linewidth to:")
    Label2.grid(row=11, sticky=W)
    self.linewidth_from = Entry(master, width=24)
    self.linewidth_from.grid(row= 10, column=1)
    widget = self.linewidth_from
    self.linewidth_to = Entry(master, width=24)
    self.linewidth_to.grid(row= 11, column=1)
    widget2 = self.linewidth_to

    if self.boxmethod.get() in ('Automatically'):
        Label1.grid()
        Label2.grid()
        widget.grid()
        widget2.grid()
        self.OnButtonClick(self.button_id)
    else:
        Label1.config(fg = 'grey')
        Label2.config(fg = 'grey')
        widget.config(state=DISABLED)
        widget2.config(state=DISABLED)
        self.OnButtonClick(self.button_id)
        number_of_classes = int(self.box3.get())
        self.add_manual_size(number_of_classes)

def savetofolder(self):
    self.saveoutput = tkFileDialog.asksaveasfilename(parent=root)
    if self.saveoutput is None:
       return
    self.label_savefolder.delete(0,END)
    self.label_savefolder.insert(0,self.saveoutput)

def createWidgets(self):
    #import pdb;pdb.set_trace()
    master = self
    Label(master, text="Default language: EN").grid(row=0, sticky=W)
    Label(master, text="Number of classes:").grid(row=1, sticky=W)
    Label(master, text="Select a method of description:").grid(row=2, sticky=W)
    Label(master, text="Classification Method:").grid(row=9, sticky=W)
    Label(master, text="Font family used in graph:").grid(row=16, column=3, padx=30,
    sticky=W)
    Label(master, text="Font size used in graph:").grid(row=17, column=3, padx=30,
    sticky=W)
    Label(master, text="Output format:").grid(row=18, column=3, padx=30, sticky=W)
```

```
value = StringVar()
self.box3 = ttk.Combobox(master,
  values = ['2', '3', '4', '5', '6', '7', '8', '9', '10']
  )
self.box3.state(['readonly'])
self.box3.bind('<<ComboboxSelected>>', self.toggle)
self.box3.current(0)
self.box3.grid(row = 1, column = 1)

value = StringVar()
self.boxmethod = ttk.Combobox(master, values = ['Automatically','Manually'])
self.boxmethod.state(['readonly'])
self.boxmethod.bind('<<ComboboxSelected>>', self.toggle)
self.boxmethod.current(0)
self.boxmethod.grid(row = 9, column = 1)

fn1 = os.path.join(os.path.dirname(__file__), 'description1.png')
image1 = Image.open(fn1)
image1 = image1.resize((220, 90), Image.ANTIALIAS)
photo1 = ImageTk.PhotoImage(image1)
self.b1 = Button(master, text= "A)", width=230, height=90, image = photo1,
                 compound=tk.RIGHT, command=lambda: self.OnButtonClick(1))
self.b1.grid(row=4, sticky=W)
self.b1.image = photo1

fn2 = os.path.join(os.path.dirname(__file__), 'description2.png')
image2 = Image.open(fn2)
image2 = image2.resize((220, 90), Image.ANTIALIAS)
photo2 = ImageTk.PhotoImage(image2)
self.b2 = Button(master, text= "B)", width=230, height=90, image = photo2,
                 compound=tk.RIGHT, command=lambda: self.OnButtonClick(2))
self.b2.grid(row=7, sticky=W)
self.b2.image = photo2

self.redraw = Button(master, text="Refresh graph", width=20, command = self.readshape)
self.redraw.grid(row=15,column=3, padx=30)

value = StringVar()
self.box4 = ttk.Combobox(master,
  values = ['Arial','Calibri','Verdana','Times New Roman']
  )
self.box4.state(['readonly'])
self.box4.bind('<<ComboboxSelected>>')
self.box4.current(0)
self.box4.grid(row = 16, column = 3, padx=160)

value = StringVar()
self.box5 = ttk.Combobox(master,
  values = ['8', '9', '10', '11', '12', '15']
  )
self.box5.state(['readonly'])
self.box5.bind('<<ComboboxSelected>>')
self.box5.current(0)
self.box5.grid(row = 17, column = 3, padx=160)

self.chart = self.create_chart(master)
self.chart._tkcanvas.grid(row=0, column=3, rowspan=14)

self.outputboxvalue = StringVar()
self.outputbox = ttk.Combobox(self, values=["JPG","PNG","TIFF","SVG"],
    )
self.outputbox.state(['readonly'])
self.outputbox.bind('<<ComboboxSelected>>')
self.outputbox.current(0)
self.outputbox.grid(row=18, column=3, padx=160)

self.label_savefoldervalue = StringVar()
self.label_savefolder= Entry(master, width=45, textvariable =
                 self.label_savefoldervalue)
self.label_savefolder.grid(row= 19, column=3, padx=30, sticky=W)

self.saveto=Button(master, text="Select output folder", width=20,
                 command=self.savetofolder)
self.saveto.grid(row= 19, column=3, padx=260, sticky=W)
```

```
    self.button = Button (master, text='Save graph', width=20, command=self.save_figure)
    self.button.grid(row=22, column=3)

    self.button2 = Button(master, text='Close window', width=20, command=self.close_window)
    self.button2.grid (row=23, column=3)

def get_linewidth_from(self):
    try:
        return self.linewidth_from.get()
    except:
        return self.linewidth_from
    pass

def get_linewidth_to(self):
    try:
        return self.linewidth_to.get()
    except:
        return self.linewidth_to
    pass

def get_savegraphtofolder(self):
    try:
        return self.label_savefolder.get()
    except:
        return self.savegraphtofolder
    pass

def get_manual_size_symbol(self):
    try:
        return self.size_entry.get()
    except:
        return self.manual_size_symbol
    pass


def readshape(self):
    self.figure.clf()
    fig = self.figure
    ax1 = fig.add_subplot(111)
    self.axis = ax1

    title_font = {'fontname':self.box4.get(), 'size':self.box5.get(), 'color':'black',
                  'weight':'normal', 'verticalalignment':'bottom'}
    axis_font = {'fontname':self.box4.get(), 'size':self.box5.get()}

    # Set the font properties (for use in legend)
    font_path = 'C:\Windows\Fonts\Arial.ttf'
    font_prop = font_manager.FontProperties(fname=font_path, size=14)

    # Set the tick labels font
    for label in (ax1.get_xticklabels() + ax1.get_yticklabels()):
        label.set_fontname(self.box4.get())
        label.set_fontsize(self.box5.get())

    ax1.set_ylim(-1, 168)
    ax1.set_xlim(0, 101)
    ax1.yaxis.set_visible(False)

    ax1.spines['right'].set_visible(False)
    ax1.spines['top'].set_visible(False)
    ax1.spines['left'].set_visible(False)
    ax1.spines['bottom'].set_visible(False)

    ax1.tick_params(left='on', top='off', right='off', bottom='off',
                    labelleft='on', labeltop='off', labelright='off', labelbottom='on')

    number_of_classes = int(self.box3.get())

    if self.boxmethod.get() in ('Automatically'):

       ##Podminky na LineWidthFrom a LineWidthTo ##
        try:
          LineWidthFrom = float(self.linewidth_from.get())
```

```
except:
   messagetxt = "Please insert value to field - 'Linewidth from'."
   message = tkMessageBox.showerror("Warning", messagetxt)
try:
  LineWidthTo = float(self.linewidth_to.get())
except:
   messagetxt = "Please insert value to field - 'Linewidth to'."
   message = tkMessageBox.showerror("Warning", messagetxt)

LineWidthTo = float(self.linewidth_to.get())
if LineWidthTo == 0:
   messagetxt = "Field - 'Linewidth to' can't be 0. Please insert another value."
   message = tkMessageBox.showerror("Warning", messagetxt)
if LineWidthTo < LineWidthFrom:
   messagetxt = "Field - 'Linewidth to' must be higher than field - 'Linewidth
                 from'. Please insert another value."
   message = tkMessageBox.showerror("Warning", messagetxt)
##

deltay = LineWidthTo - LineWidthFrom   #rozsah hodnot na ose Y
deltax = 100  #rozsah hodnot na ose X
pocet = float(self.box3.get())   #pocet vykreslenych symbolu (=pocet intervalu)
step = deltax/(pocet)
stepY =   deltay/(pocet-1)

y = LineWidthFrom
cx = 0
valueto0 = None
labels = []
if self.button_id == 1:
    for i in range(number_of_classes+1):
            if i != number_of_classes:
                rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                                          fill=False)
                ax1.add_patch(rectangle)
            if 0<i<number_of_classes:
                 valueto = float(self.to_entry[i-1].get())
                 title = valueto
                 if valueto < valueto0:
                        messagetxt = "Your value %s <  previous value %s, please
                                      change your values." % (valueto, valueto0)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                 valueto0 = valueto
            elif i==0:
                 try:
                     valueto0 = float(self.class1_entry.get())
                     title = u'(%g)' % (valueto0)
                 except:
                     title = ''
            else:
                 try:
                     valueto = float(self.classLast_entry.get())
                     if valueto < valueto0:
                        messagetxt = "Your LAST value %s <  previous value %s,
                                    please change your values." % (valueto, valueto0)
                        message = tkMessageBox.showerror("Warning", messagetxt)
                     title = u'(%g)' % (valueto)
                 except:
                     title = ''
            start, end = ax1.get_xlim()
            ax1.xaxis.set_ticks(np.arange(start, end, step))
            labels.append(title)

            cx += step
            y += stepY

    ax1.set_xticklabels(labels)

else:
    for i in range(number_of_classes):
            if i != number_of_classes:
                rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                                          fill=False)
                ax1.add_patch(rectangle)
```

```
                if 0<i<number_of_classes-1:
                     entryname_from = "class%ia_entry" % (i+1)
                     valuefrom = float(getattr(self, entryname_from).get())
                     entryname_to = "class%ib_entry" % (i+1)
                     valueto = float(getattr(self, entryname_to).get())
                     title = u'%g\u2013%g' % (valuefrom, valueto)
                     if valueto <= valuefrom:
                            messagetxt = "Your value 'To:' %s <= value 'From:' %s,
                                        please change your values for the description
                                        of class number %i" % (valueto, valuefrom, i+1)
                            message = tkMessageBox.showerror("Warning", messagetxt)
                     elif valuefrom <= valueto0:
                            messagetxt = "Your value 'From:' %s <= previous value 'To:'
                                        %s, please change your values for the
                                        description of class number %i" % (valuefrom,
                                        valueto0, i+1)
                            message = tkMessageBox.showerror("Warning", messagetxt)
                     valueto0 = valueto
                elif i==0:
                     title = " ".join((self.class1_entry_text.get().rstrip(),
                                        self.class1_entry.get()))
                     valueto0 = float(self.class1_entry.get())
                else:
                     title = " ".join((self.classLast_entry_text.get().rstrip(),
                                        self.classLast_entry.get()))
                     valuefrom = float(self.classLast_entry.get())
                     if valuefrom < valueto0:
                            messagetxt = "Your LAST value %s <  previous value 'To:'
                                %s, please change your values." % (valuefrom, valueto0)
                            message = tkMessageBox.showerror("Warning", messagetxt)
                start, end = ax1.get_xlim()
                ax1.xaxis.set_ticks(np.arange(start, end, step))
                labels.append(title)

                cx += step
                y += stepY

        minorLocator = AutoMinorLocator(2)
        ax1.xaxis.set_minor_locator(minorLocator)
        ax1.xaxis.set_major_formatter(NullFormatter())
        ax1.xaxis.set_ticklabels(labels,minor=True, **axis_font)
        ax1.xaxis.set_ticks_position('none')

else:
    deltax = 100  #rozsah hodnot na ose Y
    pocet = float(self.box3.get())   #pocet vykreslenych symbolu (=pocet intervalu)
    step = deltax/(pocet)

    cx = 0
    labels = []
    valueto0 = None
    if self.button_id == 1:
        for i in range(number_of_classes+1):
                if i != number_of_classes:
                    y = float(self.size_entry[i].get())
                    rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                                              fill=False)
                    ax1.add_patch(rectangle)
                if 0<i<number_of_classes:
                     valueto = float(self.to_entry[i-1].get())
                     title = valueto
                     if valueto < valueto0:
                            messagetxt = "Your value %s <  previous value %s, please
                                         change your values." % (valueto, valueto0)
                            message = tkMessageBox.showerror("Warning", messagetxt)
                     valueto0 = valueto
                elif i==0:
                     try:
                         valueto0 = float(self.class1_entry.get())
                         title = u'(%g)' % (valueto0)
                     except:
                         title = ''
                else:
                     try:
```

```
                            valueto = float(self.classLast_entry.get())
                            if valueto < valueto0:
                               messagetxt = "Your LAST value %s <  previous value %s,
                                         please change your values." % (valueto, valueto0)
                               message = tkMessageBox.showerror("Warning", messagetxt)
                            title = u'(%g)' % (valueto)
                        except:
                            title = ''
                    start, end = ax1.get_xlim()
                    ax1.xaxis.set_ticks(np.arange(start, end, step))
                    labels.append(title)

                    cx += step

            ax1.set_xticklabels(labels)

        else:
            for i in range(number_of_classes):
                    if i != number_of_classes:
                        y = float(self.size_entry[i].get())
                        rectangle = plt.Rectangle((cx, 0), width= step, height= y,
                                                  fill=False)
                        ax1.add_patch(rectangle)
                    if 0<i<number_of_classes-1:
                         entryname_from = "class%ia_entry" % (i+1)
                         valuefrom = float(getattr(self, entryname_from).get())
                         entryname_to = "class%ib_entry" % (i+1)
                         valueto = float(getattr(self, entryname_to).get())
                         title = u'%g\u2013%g' % (valuefrom, valueto)
                         if valueto <= valuefrom:
                                messagetxt = "Your value 'To:' %s <= value 'From:' %s,
                                             please change your values for the description
                                             of class number %i" % (valueto, valuefrom, i+1)
                                message = tkMessageBox.showerror("Warning", messagetxt)
                         elif valuefrom <= valueto0:
                                messagetxt = "Your value 'From:' %s <= previous value 'To:'
                                             %s, please change your values for the
                                             description of class number %i" % (valuefrom,
                                             valueto0, i+1)
                                message = tkMessageBox.showerror("Warning", messagetxt)
                         valueto0 = valueto
                    elif i==0:
                         title = " ".join((self.class1_entry_text.get().rstrip(),
                                           self.class1_entry.get()))
                         valueto0 = float(self.class1_entry.get())
                    else:
                         title = " ".join((self.classLast_entry_text.get().rstrip(),
                                           self.classLast_entry.get()))
                         valuefrom = float(self.classLast_entry.get())
                         if valuefrom < valueto0:
                                messagetxt = "Your LAST value %s <  previous value 'To:'
                                     %s, please change your values." % (valuefrom,valueto0)
                                message = tkMessageBox.showerror("Warning", messagetxt)
                    start, end = ax1.get_xlim()
                    ax1.xaxis.set_ticks(np.arange(start, end, step))
                    labels.append(title)

                    cx += step

            minorLocator = AutoMinorLocator(2)
            ax1.xaxis.set_minor_locator(minorLocator)
            ax1.xaxis.set_major_formatter(NullFormatter())
            ax1.xaxis.set_ticklabels(labels,minor=True, **axis_font)
            ax1.xaxis.set_ticks_position('none')

    if self.chart:
        self.chart.draw()

def save_figure(self):
    fig = self.figure
    ax1 = fig.add_subplot(111)
    self.axis = ax1
    outputpath = self.saveoutput
```

```
        if self.outputbox.get() in ('PNG'):
           suffix = '.png'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as png"
           messagetxt = "Image was saved as png."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('SVG'):
           suffix = '.svg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as svg"
           messagetxt = "Image was saved as svg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('JPG'):
           suffix = '.jpg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as jpg"
           messagetxt = "Image was saved as jpg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        else:
           suffix = '.tiff'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as tiff"
           messagetxt = "Image was saved as tiff."
           message = tkMessageBox.showinfo("Info", messagetxt)

    def close_window(self):
        root.destroy()


root = tk.Tk()
app = Application(master=root)
app.master.title(u'Automation of Cartodiagram scales creation in ArcGIS for Desktop')
app.mainloop()
#root.destroy()
```

PŘÍLOHA 5

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

from Tkinter import *
import Tkinter
tk = Tkinter
import tkFileDialog
from tkFileDialog import askopenfilename
import ttk
import tkMessageBox
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.font_manager as font_manager
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
import arcpy
from matplotlib.figure import Figure
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
import pylab
import os
import sys
import math

# density of points in x values
STEPS = 200

# Functions, how ArcMap calculates the proportional symbols
def f_normalcase(self, value, valmin, Pmin):
    return (float(value)/valmin)**0.5 * Pmin

def f_flannery(self, value,  valmin, Pmin):
    return 1.0083*(float(value)/valmin)**0.5716 * Pmin

methods = {"Normal case":f_normalcase, "Flannery method":f_flannery}

class Application(tk.Frame):
    chart = None
    soubory = []
    figsize=(6.0, 3.0)
    seznam = []

    def create_chart(self, root):
        f = Figure(figsize=(6, 3), dpi=100)
        #canvas.resize(w,h)
        self.figure = f
        # a tk.DrawingArea
        canvas = FigureCanvasTkAgg(f, master=root)
        return canvas

    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.createWidgets()

    def toggle(self, event):
        master = self
        self.label1 = Entry(master, width=24)
        self.label1.grid(row=13, column=1)

        if self.box2.get() in ('Min of input', 'Value 0'):
            self.label1.config(state=DISABLED)
        else:
            self.label1.grid()

    def toggle2(self, event):
        master=self
        self.label2 = Entry(master,  width=24)
        self.label2.grid(row=15, column=1)

        if self.box3.get() in ('Max of input'):
            self.label2.config(state=DISABLED)
        else:
            self.label2.grid()
```

```
def fileschanged(self, event):
    self.readfields(self.filesbox.get())

def load_file(self):
    filetypes = [
      ("Shapefile", "*.shp"),
      ("Geodatabase", "*.gdb"),

    ]
    self.files = tkFileDialog.askopenfilename(parent=root, filetypes=filetypes)
    self.filepath = os.path.dirname(self.files)

    self.soubory = [
          s
          for s
          in os.listdir(self.filepath)
          if s.endswith(".shp")
          ]

    self.filesbox.config(values = self.soubory)
    filename = os.path.split(self.files)[-1]
    self.filesbox.set(filename)
    self.readfields(filename)

def add_item_to_list(self):
    item = (self.filesbox.get(), self.fieldbox.get())
    self.seznam.append(item)
    self.seznamSHPbox.insert(END, "    ".join(item))

def clear_figure(self, widget):
    self.figure.clf()

def remove_item_button(self):
    i = self.seznamSHPbox.index(ANCHOR)
    del self.seznam[i]
    self.seznamSHPbox.delete(ANCHOR)

def savetofolder(self):
    self.saveoutput = tkFileDialog.asksaveasfilename(parent=root)

    if self.saveoutput is None:
       return

    self.label_savefolder.delete(0,END)
    self.label_savefolder.insert(0,self.saveoutput)

def createWidgets(self):
    #import pdb;pdb.set_trace()
    master = self
    Label(master, text="Default language: EN").grid(row=0, sticky=W)
    Label(master, text="Select folder with feature(s):").grid(row=1, sticky=W)
    Label(master, text="Input feature:").grid(row=2, sticky=W)
    Label(master, text="Input field:").grid(row=3, sticky=W)
    Label(master, text="List of selected features and fields:").grid(row=4, sticky=W)
    Label(master, text="Select function:").grid(row=10, sticky=W)
    Label(master, text="Minimum size of proportional symbol from ArcMap:").grid(row=11,
    sticky=W)
    Label(master, text="Axis X - minimum value:").grid(row=12, sticky=W)
    Label(master, text="Axis X - maximum value:").grid(row=14, sticky=W)
    Label(master, text="Axis X - label:").grid(row=16, sticky=W)
    Label(master, text="Axis X - Major division (wholenumber multiples):").grid(row=17,
    sticky=W)
    Label(master, text="Axis X - Minor division:").grid(row=18, sticky=W)
    Label(master, text="Font family used in graph:").grid(row=14, column = 2, padx = 120,
    sticky = W)
    Label(master, text="Font size used in graph:").grid(row=15, column = 2, padx = 120,
    sticky = W)
    Label(master, text="Output format:").grid(row=16, column = 2, padx = 120, sticky = W)

    self.button = Button(self, text = "Select", width=20, command=self.load_file)
    self.button.grid(row=1,column=1)
```

```
self.filesboxvalue = StringVar()
self.filesbox = ttk.Combobox(self, values=self.soubory,
    textvariable=self.filesboxvalue)
self.filesbox.bind('<<ComboboxSelected>>', self.fileschanged)
self.filesbox.state(['readonly'])
self.filesbox.grid(row=2, column=1)

self.fieldboxvalue = StringVar()
self.fieldbox = ttk.Combobox(self, values=[],
    textvariable=self.fieldboxvalue)
self.fieldbox.state(['readonly'])
self.fieldbox.bind('<<ComboboxSelected>>')
self.fieldbox.grid(row=3, column=1)

self.seznamSHPbox = Listbox(self, selectmode=EXTENDED, width=30, height=10)
for item in self.seznam:
    self.seznamSHPbox.insert(END, "   ".join(item))
self.seznamSHPbox.grid(row=5, sticky=W, rowspan=4)

self.add_item_button = Button(self, text = "Add", width=20,
                              command=self.add_item_to_list)
self.add_item_button.grid(row=5, column=1)

self.remove_item_button = Button(self,text = "Remove", width=20,
                              command=self.remove_item_button)
self.remove_item_button.grid(row=6,column=1)

self.methodvalue = StringVar()
self.box1 = ttk.Combobox(master, textvariable=self.methodvalue,
  values = methods.keys(),
  )
self.box1.state(['readonly'])
self.box1.bind('<<ComboboxSelected>>')
self.box1.current(0)
self.box1.grid(row = 10, column = 1)

self.MinSizeSymbol = Entry(master, width=24)
self.MinSizeSymbol.grid(row=11, column=1)

value = StringVar()
self.box2 = ttk.Combobox(master,
  values = ['Min of input', 'Custom value']
  )
self.box2.state(['readonly'])
self.box2.bind('<<ComboboxSelected>>', self.toggle)
self.box2.current(0)
self.box2.grid(row = 12, column = 1)

value = StringVar()
self.box3 = ttk.Combobox(master,
  values = ['Max of input', 'Custom value']
  )
self.box3.bind('<<ComboboxSelected>>', self.toggle2)
self.box3.state(['readonly'])
self.box3.current(0)
self.box3.grid(row = 14, column = 1)

self.e8 = Entry(master, width=24)
self.e8.grid(row= 16, column=1)

value = StringVar()
self.box4 = ttk.Combobox(master,
  values = ['Arial','Calibri','Verdana','Times New Roman']
  )
self.box4.state(['readonly'])
self.box4.bind('<<ComboboxSelected>>')
self.box4.current(0)
self.box4.grid(row = 14, column = 2, padx = 170)

value = StringVar()
self.box5 = ttk.Combobox(master,
  values = ['8', '9', '10', '11', '12', '15']
  )
```

```
        self.box5.state(['readonly'])
        self.box5.bind('<<ComboboxSelected>>')
        self.box5.current(0)
        self.box5.grid(row = 15, column = 2, padx = 170)

        self.majorticksvalue = StringVar()
        self.majorticks = Entry(master,width=10)
        self.majorticks.grid(row=17, column=1, sticky =W)

        self.minorticksvalue = StringVar()
        self.minorticks = Entry(master,width=10)
        self.minorticks.grid(row=18, column=1, sticky =W)

        self.chart = self.create_chart(master)
        self.chart._tkcanvas.grid(row=0, column=2, rowspan=12)

        self.outputboxvalue = StringVar()
        self.outputbox = ttk.Combobox(self, values=["JPG","PNG","TIFF","SVG"],
            )
        self.outputbox.state(['readonly'])
        self.outputbox.bind('<<ComboboxSelected>>')
        self.outputbox.current(0)
        self.outputbox.grid(row=16, column=2, padx = 170)

        self.label_savefoldervalue = StringVar()
        self.label_savefolder= Entry(master, width=45, textvariable =
                                     self.label_savefoldervalue)
        self.label_savefolder.grid(row= 17, column = 2, padx = 30, sticky=W)

        self.saveto=Button(master, text="Select output folder", width=20,
                                     command=self.savetofolder)
        self.saveto.grid(row= 17, column=2, padx = 270, sticky = W)

        self.redraw = Button(master, text="Refresh graph", width=20, command = self.readshape)
        self.redraw.grid(row=12,column=2)

        self.button = Button (master, text='Save graph', width=20, command=self.save_figure)
        self.button.grid(row=19, column=2)

        self.button2 = Button(master, text='Close window', width=20, command=self.close_window)
        self.button2.grid (row=20, column=2)

    def get_minsizesymbol(self):
        try:
            return self.MinSizeSymbol.get()
        except:
            return self.minsizesymbol
        pass

    def get_xlabel(self):
        try:
            return self.e8.get()
        except:
            return self.xlabel
        pass

    def get_xmin_value(self):
        try:
            return self.label1.get()
        except:
            return self.xmin_value
        pass

    def get_xmax_value(self):
        try:
            return self.label2.get()
        except:
            return self.xmax_value
        pass

    def get_majorticks(self):
        try:
            return self.majorticks.get()
        except:
```

```
        return self.majorticks
    pass

def get_minorticks(self):
    try:
        return self.minorticks.get()
    except:
        return self.minorticks
    pass

def get_savegraphtofolder(self):
    try:
        return self.label_savefolder.get()
    except:
        return self.savegraphtofolder
    pass

def readshape(self):
    self.figure.clf()
    values = []
    for feature, field in self.seznam:
        dbpath = os.path.join(self.filepath, feature)
        cursor = arcpy.SearchCursor(dbpath)
        values.extend(
            [
                getattr(row, field)
                for row in cursor
            ]
        )
        del cursor, row

    self.xlabel = self.get_xlabel()

    if not values:
        return
    valmin = min(values)

    valmax = max(values)
    step = (valmax-valmin)/STEPS

    values = []
    cx = valmin
    while cx < valmax:
        values.append(cx)
        cx += step
    values.append(valmax)

    Pmin = float(self.MinSizeSymbol.get())
    try:
          Pmin = float(self.MinSizeSymbol.get())
    except:
           messagetxt = "Please insert value to field - 'Minimum size of symbol from
                        ArcMap'."
           message = tkMessageBox.showerror("Warning", messagetxt)
    if Pmin >=100:
       messagetxt = "Please insert lower value than 100 to field - 'Minimum size of symbol
                        from ArcMap'."
       message = tkMessageBox.showerror("Warning", messagetxt)

    values.sort()

    f = methods.get(self.methodvalue.get(), f_normalcase)
    yvalues = [ f(self, value, valmin, Pmin)
      for value in values
    ]

    fig = self.figure
    ax1 = fig.add_subplot(111)
    self.axis = ax1

    title_font = {'fontname':self.box4.get(), 'size':self.box5.get(), 'color':'black',
    'weight':'normal', 'verticalalignment':'bottom'}
    axis_font = {'fontname':self.box4.get(), 'size':self.box5.get()}
```

```
# Set the font properties (for use in legend)
font_path = 'C:\Windows\Fonts\Arial.ttf'
font_prop = font_manager.FontProperties(fname=font_path, size=14)

# Set the tick labels font
for label in (ax1.get_xticklabels() + ax1.get_yticklabels()):
    label.set_fontname(self.box4.get())
    label.set_fontsize(self.box5.get())

# Set the tick - Major and Minor
majorTicks = float(self.majorticks.get())
minorTicks = float(self.minorticks.get())
majorLocator = MultipleLocator(float(self.majorticks.get()))
majorFormatter = FormatStrFormatter('%d')
minorLocator = MultipleLocator(float(self.minorticks.get()))

if self.box2.get() in ('Min of input') and self.box3.get() in ('Max of input'):

       x = sorted(values)
       yvalues = [ f(self, value, valmin, Pmin)
            for value in values
              ]

       cx1 = majorTicks
       cy1 = None
       while cx1 < valmax:
           cy1 = f(self, cx1, valmin, Pmin)
           ax1.vlines(cx1, 0, cy1, color='black', lw=1)
           cx1 += majorTicks
       if cy1 is None:
           return

       ax1.xaxis.set_minor_locator(minorLocator)
       ticks = [valmin]
       xval = (1+int(valmin)/int(majorTicks))*majorTicks
       while xval<=valmax:
            ticks.append(xval)
            xval += majorTicks
       ticks.append(valmax)
       if valmin == 0:
           cy = 0
       else:
           cy = f(self, valmin, valmin, Pmin)
       ax1.vlines(valmin, 0, cy, color='black', lw=1)

       cy = f(self, valmax, valmin, Pmin)
       ax1.vlines(valmax-1, 0, cy, color='black', lw=1)

       ax1.set_xticks(ticks)


elif self.box2.get() in ('Custom value') and self.box3.get() in ('Max of input'):

       x = sorted(values)
       user_valmin = float(self.label1.get())
       if user_valmin > valmin:
           messagetxt = "Please insert lower value than minimum value of input
                        feature."
           message = tkMessageBox.showerror("Warning", messagetxt)

       density = (valmax-valmin)/float(len(x))
       stepx = density*(valmin-user_valmin)/(valmax-valmin)
       userx = user_valmin
       i = 0
       while userx < valmin:
           x.insert(i, userx)
           yvalues.insert(i, f(self, userx, valmin, Pmin))
           userx += stepx
           i += 1

       cx = majorTicks
       cy = None
       while cx < valmax:
           cy = f(self, cx, valmin, Pmin)
```

```
            ax1.vlines(cx, 0, cy, color='black', lw=1)
            ax1.vlines(valmax-1, 0, cy, color='black', lw=1)
            cx += majorTicks
        if cy is None:
            return

        valmin = user_valmin

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmin, Pmin)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)

elif self.box2.get() in ('Min of input') and self.box3.get() in ('Custom value'):

        x = sorted(values)

        user_valmax = float(self.label2.get())
        if user_valmax < valmax:
            messagetxt = "Please insert higher value than maximum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        density = (valmax-valmin)/float(len(x))
        stepx1 = density *(user_valmax-valmax)/(valmax-valmin)
        userx = valmax
        i = (len(x))
        while userx < user_valmax:
            x.insert(i, userx)
            yvalues.insert(i, f(self, userx, valmin, Pmin))
            userx += stepx1
            i += 1

        cx1 = majorTicks
        cy1 = None
        while cx1 < user_valmax:
            cy1 = f(self, cx1, valmin, Pmin)
            ax1.vlines(cx1, 0, cy1, color='black', lw=1)
            cx1 += majorTicks
        if cy1 is None:
            return

        valmax = user_valmax

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmin, Pmin)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        cy = f(self, valmax, valmin, Pmin)
        ax1.vlines(valmax-1, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)
```

```
elif self.box2.get() in ('Custom value') and self.box3.get() in ('Custom value'):

        x = sorted(values)
        user_valmin = float(self.label1.get())
        if user_valmin > valmin:
            messagetxt = "Please insert lower value than minimum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        user_valmax = float(self.label2.get())
        if user_valmax < valmax:
            messagetxt = "Please insert higher value than maximum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        density = (valmax-valmin)/float(len(x))
        stepx = density*(valmin-user_valmin)/(valmax-valmin)
        userx = user_valmin
        i = 0
        while userx < valmin:
            x.insert(i, userx)
            yvalues.insert(i, f(self, userx, valmin, Pmin))
            userx += stepx
            i += 1

        stepx1 = density *(user_valmax-valmax)/(valmax-valmin)
        userx = valmax
        i = (len(x))
        while userx < user_valmax:
            x.insert(i, userx)
            yvalues.insert(i, f(self, userx, valmin, Pmin))
            userx += stepx1
            i += 1

        cx = majorTicks
        cy = None
        while cx <= user_valmax:
            cy = f(self, cx, valmin, Pmin)
            ax1.vlines(cx, 0, cy, color='black', lw=1)
            ax1.vlines(user_valmax-1, 0, cy, color='black', lw=1)
            cx += majorTicks
        if cy is None:
            return

        valmin = user_valmin
        valmax = user_valmax

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmin, Pmin)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)

y = sorted(yvalues)

#Conversion between pt(ArcGIS) and inch(Matplotlib)
# 1pt = 1/72 inch  --> 10pt = 10/72 inch

ax1.set_ylim(0, 168)
ax1.set_xlim(valmin, valmax*1.01)
ax1.yaxis.set_visible(False)
label = ax1.set_xlabel(self.e8.get(), **axis_font)
ax1.xaxis.set_label_coords(1.05, -0.10)
```

```
        ax1.spines['right'].set_visible(False)
        ax1.spines['top'].set_visible(False)
        ax1.spines['left'].set_visible(False)

        ax1.tick_params(which='both', direction='out')
        ax1.xaxis.set_ticks_position('bottom')
        ax1.ticklabel_format(style='sci',scilimits=(-10,10),axis='both')
        ax1.xaxis.major.formatter_useMathText = True

        fig.tight_layout()
        ax1.plot(x, y, color='black', lw=2)

        if self.chart:
            self.chart.draw()

    def readfields(self, soubor):

        dbpath = os.path.join(self.filepath, soubor)
        shapename = dbpath.rsplit(".", 1)[0]
        shapename = os.path.split(shapename)[-1]

        # Get and print a list of tables
        fields = arcpy.ListFields(dbpath)
        values = [f.name for f in fields]
        self.fieldbox.config(values = values,
          )
        return values

    def save_figure(self):
        fig = self.figure
        ax1 = fig.add_subplot(111)
        self.axis = ax1
        outputpath = self.saveoutput

        if self.outputbox.get() in ('PNG'):
           suffix = '.png'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as png"
           messagetxt = "Image was saved as png."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('SVG'):
           suffix = '.svg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as svg"
           messagetxt = "Image was saved as svg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('JPG'):
           suffix = '.jpg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as jpg"
           messagetxt = "Image was saved as jpg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        else:
           suffix = '.tiff'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as tiff"
           messagetxt = "Image was saved as tiff."
           message = tkMessageBox.showinfo("Info", messagetxt)

    def close_window(self):
        root.destroy()


root = tk.Tk()
app = Application(master=root)
app.master.title(u'Automation of Cartodiagram scales creation in ArcGIS for Desktop')
app.mainloop()
#root.destroy()
```

PŘÍLOHA 6

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

from Tkinter import *
import Tkinter
tk = Tkinter
import tkFileDialog
from tkFileDialog import askopenfilename
import ttk
import tkMessageBox
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.font_manager as font_manager
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
import arcpy
from matplotlib.figure import Figure
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
import pylab
import os
import sys
import math

# density of points in x values
STEPS = 200

# Functions, how calculate the Bar/Column/Stacked charts
def f(self, value, valmax, Pmax):
    return (float(value)/valmax) * Pmax

class Application(tk.Frame):
    chart = None
    soubory = []
    figsize=(6.0, 3.0)
    seznam = []

    def create_chart(self, root):
        f = Figure(figsize=(6, 3), dpi=100)
        #canvas.resize(w,h)
        self.figure = f
        # a tk.DrawingArea
        canvas = FigureCanvasTkAgg(f, master=root)
        return canvas

    def __init__(self, master=None):
        tk.Frame.__init__(self, master)
        self.grid()
        self.createWidgets()

    def toggle(self, event):
        master = self
        self.label1 = Entry(master, width=24)
        self.label1.grid(row=13, column=1)

        if self.box2.get() in ('Min of input', 'Value 0'):
            self.label1.config(state=DISABLED)
        else:
            self.label1.grid()

    def toggle2(self, event):
        master=self
        self.label2 = Entry(master,  width=24)
        self.label2.grid(row=15, column=1)

        if self.box3.get() in ('Max of input'):
            self.label2.config(state=DISABLED)
        else:
            self.label2.grid()

    def fileschanged(self, event):
        self.readfields(self.filesbox.get())

    def load_file(self):
        filetypes = [
```

```
      ("Shapefile", "*.shp"),
      ("Geodatabase", "*.gdb"),

    ]
    self.files = tkFileDialog.askopenfilename(parent=root, filetypes=filetypes)
    self.filepath = os.path.dirname(self.files)

    self.soubory = [
          s
          for s
          in os.listdir(self.filepath)
          if s.endswith(".shp")
          ]

    self.filesbox.config(values = self.soubory)
    filename = os.path.split(self.files)[-1]
    self.filesbox.set(filename)
    self.readfields(filename)

def add_item_to_list(self):
    item = (self.filesbox.get(), self.fieldbox.get())
    self.seznam.append(item)
    self.seznamSHPbox.insert(END, "    ".join(item))

def clear_figure(self, widget):
    self.figure.clf()

def remove_item_button(self):
    i = self.seznamSHPbox.index(ANCHOR)
    del self.seznam[i]
    self.seznamSHPbox.delete(ANCHOR)

def savetofolder(self):
    self.saveoutput = tkFileDialog.asksaveasfilename(parent=root)

    if self.saveoutput is None:
       return

    self.label_savefolder.delete(0,END)
    self.label_savefolder.insert(0,self.saveoutput)

def createWidgets(self):
    #import pdb;pdb.set_trace()
    master = self
    Label(master, text="Default language: EN").grid(row=0, sticky=W)
    Label(master, text="Select folder with feature(s):").grid(row=1, sticky=W)
    Label(master, text="Input feature:").grid(row=2, sticky=W)
    Label(master, text="Input field:").grid(row=3, sticky=W)
    Label(master, text="List of selected features and fields:").grid(row=4, sticky=W)
    Label(master, text="Maximum length of symbol from ArcMap (in pts):").grid(row=11,
    sticky=W)
    Label(master, text="Axis X - minimum value:").grid(row=12, sticky=W)
    Label(master, text="Axis X - maximum value:").grid(row=14, sticky=W)
    Label(master, text="Axis X - label:").grid(row=16, sticky=W)
    Label(master, text="Axis X - Major division (wholenumber multiples):").grid(row=17,
    sticky=W)
    Label(master, text="Axis X - Minor division:").grid(row=18, sticky=W)
    Label(master, text="Font family used in graph:").grid(row=14, column = 2, padx = 120,
    sticky = W)
    Label(master, text="Font size used in graph:").grid(row=15, column = 2, padx = 120,
    sticky = W)
    Label(master, text="Output format:").grid(row=16, column = 2, padx = 120, sticky = W)

    self.button = Button(self, text = "Select", width=20, command=self.load_file)
    self.button.grid(row=1,column=1)

    self.filesboxvalue = StringVar()
    self.filesbox = ttk.Combobox(self, values=self.soubory,
        textvariable=self.filesboxvalue)
    self.filesbox.bind('<<ComboboxSelected>>', self.fileschanged)
    self.filesbox.state(['readonly'])
    self.filesbox.grid(row=2, column=1)
```

```
self.fieldboxvalue = StringVar()
self.fieldbox = ttk.Combobox(self, values=[],
    textvariable=self.fieldboxvalue)
self.fieldbox.state(['readonly'])
self.fieldbox.bind('<<ComboboxSelected>>')
self.fieldbox.grid(row=3, column=1)

self.seznamSHPbox = Listbox(self, selectmode=EXTENDED, width=30, height=10)
for item in self.seznam:
    self.seznamSHPbox.insert(END, "   ".join(item))
self.seznamSHPbox.grid(row=5, sticky=W, rowspan=4)

self.add_item_button = Button(self, text = "Add", width=20,
                              command=self.add_item_to_list)
self.add_item_button.grid(row=5, column=1)

self.remove_item_button = Button(self,text = "Remove", width=20,
                              command=self.remove_item_button)
self.remove_item_button.grid(row=6,column=1)

self.MaxSizeSymbol = Entry(master, width=24)
self.MaxSizeSymbol.grid(row=11, column=1)

value = StringVar()
self.box2 = ttk.Combobox(master,
  values = ['Min of input', 'Custom value']
  )
self.box2.state(['readonly'])
self.box2.bind('<<ComboboxSelected>>', self.toggle)
self.box2.current(0)
self.box2.grid(row = 12, column = 1)

value = StringVar()
self.box3 = ttk.Combobox(master,
  values = ['Max of input', 'Custom value']
  )
self.box3.bind('<<ComboboxSelected>>', self.toggle2)
self.box3.state(['readonly'])
self.box3.current(0)
self.box3.grid(row = 14, column = 1)

self.e8 = Entry(master, width=24)
self.e8.grid(row= 16, column=1)

value = StringVar()
self.box4 = ttk.Combobox(master,
  values = ['Arial','Calibri','Verdana','Times New Roman']
  )
self.box4.state(['readonly'])
self.box4.bind('<<ComboboxSelected>>')
self.box4.current(0)
self.box4.grid(row = 14, column = 2, padx = 170)

value = StringVar()
self.box5 = ttk.Combobox(master,
  values = ['8', '9', '10', '11', '12', '15']
  )
self.box5.state(['readonly'])
self.box5.bind('<<ComboboxSelected>>')
self.box5.current(0)
self.box5.grid(row = 15, column = 2, padx = 170)

self.majorticksvalue = StringVar()
self.majorticks = Entry(master,width=10)
self.majorticks.grid(row=17, column=1, sticky =W)

self.minorticksvalue = StringVar()
self.minorticks = Entry(master,width=10)
self.minorticks.grid(row=18, column=1, sticky =W)

self.chart = self.create_chart(master)
self.chart._tkcanvas.grid(row=0, column=2, rowspan=12)
```

```
        self.outputboxvalue = StringVar()
        self.outputbox = ttk.Combobox(self, values=["JPG","PNG","TIFF","SVG"],
            )
        self.outputbox.state(['readonly'])
        self.outputbox.bind('<<ComboboxSelected>>')
        self.outputbox.current(0)
        self.outputbox.grid(row=16, column=2, padx = 170)

        self.label_savefoldervalue = StringVar()
        self.label_savefolder= Entry(master, width=45, textvariable =
                                     self.label_savefoldervalue)
        self.label_savefolder.grid(row= 17, column = 2, padx = 30, sticky=W)

        self.saveto=Button(master, text="Select output folder", width=20,
                                     command=self.savetofolder)
        self.saveto.grid(row= 17, column=2, padx = 270, sticky = W)

        self.redraw = Button(master, text="Refresh graph", width=20, command = self.readshape)
        self.redraw.grid(row=12,column=2)

        self.button = Button (master, text='Save graph', width=20, command=self.save_figure)
        self.button.grid(row=19, column=2)

        self.button2 = Button(master, text='Close window', width=20, command=self.close_window)
        self.button2.grid (row=20, column=2)

    def get_maxsizesymbol(self):
        try:
            return self.MaxSizeSymbol.get()
        except:
            return self.maxsizesymbol
        pass

    def get_xlabel(self):
        try:
            return self.e8.get()
        except:
            return self.xlabel
        pass

    def get_xmin_value(self):
        try:
            return self.label1.get()
        except:
            return self.xmin_value
        pass

    def get_xmax_value(self):
        try:
            return self.label2.get()
        except:
            return self.xmax_value
        pass

    def get_majorticks(self):
        try:
            return self.majorticks.get()
        except:
            return self.majorticks
        pass

    def get_minorticks(self):
        try:
            return self.minorticks.get()
        except:
            return self.minorticks
        pass

    def get_savegraphtofolder(self):
        try:
            return self.label_savefolder.get()
        except:
            return self.savegraphtofolder
        pass
```

```
def readshape(self):
    self.figure.clf()

    values = []
    for feature, field in self.seznam:
        dbpath = os.path.join(self.filepath, feature)
        cursor = arcpy.SearchCursor(dbpath)
        values.extend(
            [
                getattr(row, field)
                for row in cursor
            ]
        )
        del cursor, row

    self.xlabel = self.get_xlabel()

    if not values:
        return
    valmin = min(values)

    valmax = max(values)
    step = (valmax-valmin)/STEPS

    values = []
    cx = valmin
    while cx < valmax:
        values.append(cx)
        cx += step
    values.append(valmax)

    Pmax = float(self.MaxSizeSymbol.get())
    try:
          Pmax = float(self.MaxSizeSymbol.get())
    except:
          messagetxt = "Please insert value to field - 'Maximum length of symbol from
                       ArcMap'."
          message = tkMessageBox.showerror("Warning", messagetxt)
    if Pmax >100:
       messagetxt = "Please insert lower value than 100 to field - 'Maximum length of
                       symbol from ArcMap'."
       message = tkMessageBox.showerror("Warning", messagetxt)

    values.sort()

    yvalues = [ f(self, value, valmax, Pmax)
      for value in values
    ]

    fig = self.figure
    ax1 = fig.add_subplot(111)
    self.axis = ax1

    title_font = {'fontname':self.box4.get(), 'size':self.box5.get(), 'color':'black',
                       'weight':'normal', 'verticalalignment':'bottom'}
    axis_font = {'fontname':self.box4.get(), 'size':self.box5.get()}

    # Set the font properties (for use in legend)
    font_path = 'C:\Windows\Fonts\Arial.ttf'
    font_prop = font_manager.FontProperties(fname=font_path, size=14)

    # Set the tick labels font
    for label in (ax1.get_xticklabels() + ax1.get_yticklabels()):
        label.set_fontname(self.box4.get())
        label.set_fontsize(self.box5.get())

    # Set the tick - Major and Minor
    majorTicks = float(self.majorticks.get())
    minorTicks = float(self.minorticks.get())
    majorLocator = MultipleLocator(float(self.majorticks.get()))
    majorFormatter = FormatStrFormatter('%d')
    minorLocator = MultipleLocator(float(self.minorticks.get()))
```

```
if self.box2.get() in ('Min of input') and self.box3.get() in ('Max of input'):

        x = sorted(values)
        yvalues = [ f(self, value, valmax, Pmax)
             for value in values
               ]

        cx1 = majorTicks
        cy1 = None
        while cx1 < valmax:
            cy1 = f(self, cx1, valmax, Pmax)
            ax1.vlines(cx1, 0, cy1, color='black', lw=1)
            cx1 += majorTicks
        if cy1 is None:
            return

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmax, Pmax)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        cy = f(self, valmax, valmax, Pmax)
        ax1.vlines(valmax-1, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)

elif self.box2.get() in ('Custom value') and self.box3.get() in ('Max of input'):

        x = sorted(values)
        user_valmin = float(self.label1.get())
        if user_valmin > valmin:
            messagetxt = "Please insert lower value than minimum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        density = (valmax-valmin)/float(len(x))
        stepx = density*(valmin-user_valmin)/(valmax-valmin)
        userx = user_valmin
        i = 0
        while userx < valmin:
            x.insert(i, userx)
            yvalues.insert(i, f(self, userx, valmax, Pmax))
            userx += stepx
            i += 1

        cx = majorTicks
        cy = None
        while cx < valmax:
            cy = f(self, cx, valmax, Pmax)
            ax1.vlines(cx, 0, cy, color='black', lw=1)
            ax1.vlines(valmax-1, 0, cy, color='black', lw=1)
            cx += majorTicks
        if cy is None:
            return

        valmin = user_valmin

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
```

```
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmax, Pmax)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)

    elif self.box2.get() in ('Min of input') and self.box3.get() in ('Custom value'):

        x = sorted(values)

        user_valmax = float(self.label2.get())
        if user_valmax < valmax:
            messagetxt = "Please insert higher value than maximum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        density = (valmax-valmin)/float(len(x))
        stepx1 = density *(user_valmax-valmax)/(valmax-valmin)
        userx = valmax
        i = (len(x))
        while userx < user_valmax:
            x.insert(i, userx)
            yvalues.insert(i, f(self, userx, valmax, Pmax))
            userx += stepx1
            i += 1

        cx1 = majorTicks
        cy1 = None
        while cx1 < user_valmax:
            cy1 = f(self, cx1, valmax, Pmax)
            ax1.vlines(cx1, 0, cy1, color='black', lw=1)
            cx1 += majorTicks
        if cy1 is None:
            return

        valmax = user_valmax

        ax1.xaxis.set_minor_locator(minorLocator)
        ticks = [valmin]
        xval = (1+int(valmin)/int(majorTicks))*majorTicks
        while xval<=valmax:
             ticks.append(xval)
             xval += majorTicks
        ticks.append(valmax)
        if valmin == 0:
            cy = 0
        else:
            cy = f(self, valmin, valmax, Pmax)
        ax1.vlines(valmin, 0, cy, color='black', lw=1)

        cy = f(self, valmax, valmax, Pmax)
        ax1.vlines(valmax-1, 0, cy, color='black', lw=1)

        ax1.set_xticks(ticks)

    elif self.box2.get() in ('Custom value') and self.box3.get() in ('Custom value'):

        x = sorted(values)
        user_valmin = float(self.label1.get())
        if user_valmin > valmin:
            messagetxt = "Please insert lower value than minimum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        user_valmax = float(self.label2.get())
        if user_valmax < valmax:
            messagetxt = "Please insert higher value than maximum value of input
                          feature."
            message = tkMessageBox.showerror("Warning", messagetxt)

        density = (valmax-valmin)/float(len(x))
        stepx = density*(valmin-user_valmin)/(valmax-valmin)
```

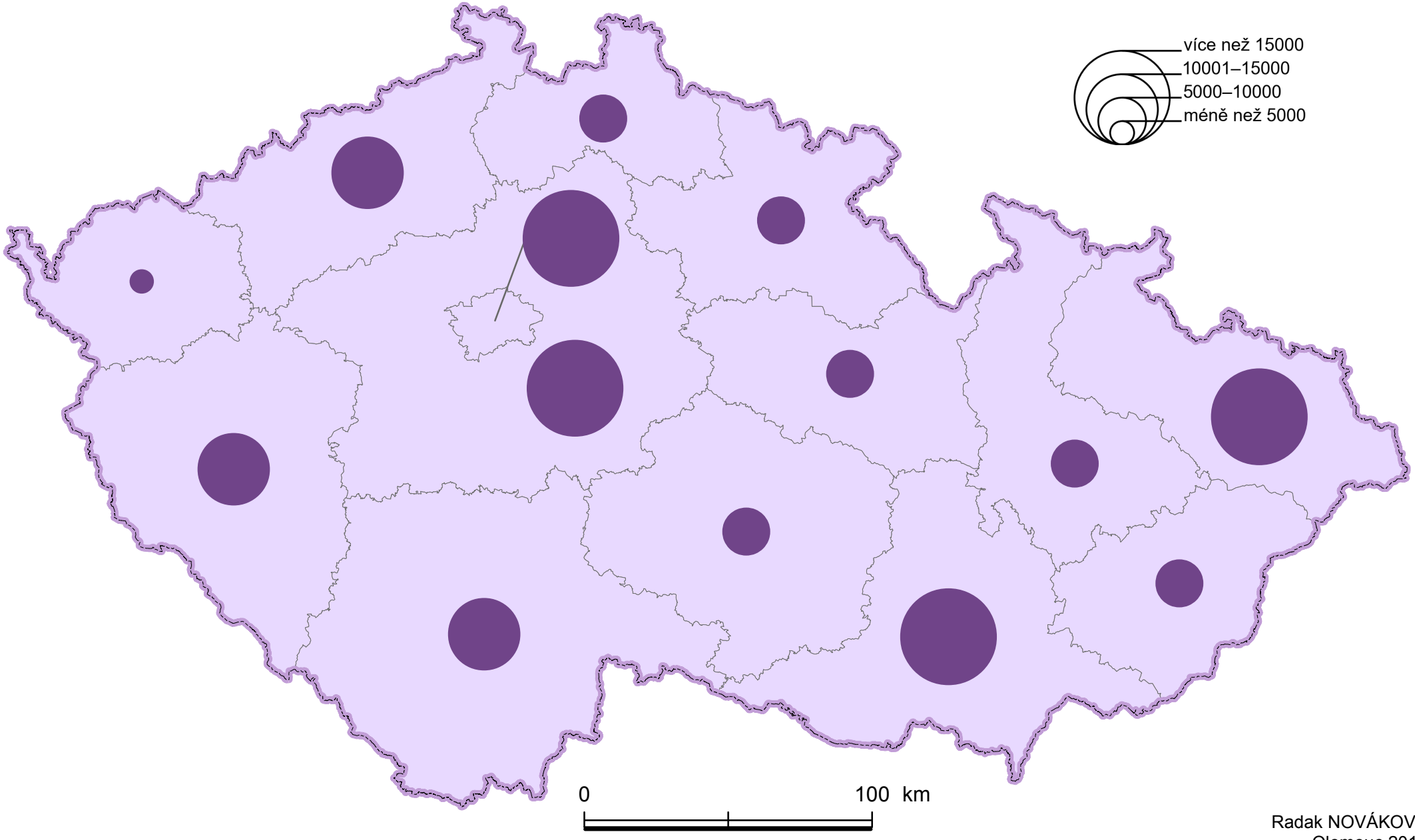
```
            userx = user_valmin
            i = 0
            while userx < valmin:
                x.insert(i, userx)
                yvalues.insert(i, f(self, userx, valmax, Pmax))
                userx += stepx
                i += 1

            stepx1 = density *(user_valmax-valmax)/(valmax-valmin)
            userx = valmax
            i = (len(x))
            while userx < user_valmax:
                x.insert(i, userx)
                yvalues.insert(i, f(self, userx, valmax, Pmax))
                userx += stepx1
                i += 1

            cx = majorTicks
            cy = None
            while cx <= user_valmax:
                cy = f(self, cx, valmax, Pmax)
                ax1.vlines(cx, 0, cy, color='black', lw=1)
                ax1.vlines(user_valmax-1, 0, cy, color='black', lw=1)
                cx += majorTicks
            if cy is None:
                return

            valmin = user_valmin
            valmax = user_valmax

            ax1.xaxis.set_minor_locator(minorLocator)
            ticks = [valmin]
            xval = (1+int(valmin)/int(majorTicks))*majorTicks
            while xval<=valmax:
                 ticks.append(xval)
                 xval += majorTicks
            ticks.append(valmax)
            if valmin == 0:
                cy = 0
            else:
                cy = f(self, valmin, valmax, Pmax)
            ax1.vlines(valmin, 0, cy, color='black', lw=1)

            ax1.set_xticks(ticks)

    y = sorted(yvalues)

    #Conversion between pt(ArcGIS) and inch(Matplotlib)
    # 1pt = 1/72 inch  --> 10pt = 10/72 inch

    ax1.set_ylim(0, 168)
    ax1.set_xlim(valmin, valmax*1.01)
    ax1.yaxis.set_visible(False)
    label = ax1.set_xlabel(self.e8.get(), **axis_font)
    ax1.xaxis.set_label_coords(1.05, -0.10)

    ax1.spines['right'].set_visible(False)
    ax1.spines['top'].set_visible(False)
    ax1.spines['left'].set_visible(False)

    ax1.tick_params(which='both', direction='out')
    ax1.xaxis.set_ticks_position('bottom')
    ax1.ticklabel_format(style='sci',scilimits=(-10,10),axis='both')
    ax1.xaxis.major.formatter_useMathText = True

    fig.tight_layout()
    ax1.plot(x, y, color='black', lw=2)

    if self.chart:
        self.chart.draw()

def readfields(self, soubor):
    dbpath = os.path.join(self.filepath, soubor)
```

```
        shapename = dbpath.rsplit(".", 1)[0]
        shapename = os.path.split(shapename)[-1]

        # Get and print a list of tables
        fields = arcpy.ListFields(dbpath)
        values = [f.name for f in fields]
        self.fieldbox.config(values = values,
          )
        return values

    def save_figure(self):
        fig = self.figure
        ax1 = fig.add_subplot(111)
        self.axis = ax1
        outputpath = self.saveoutput

        if self.outputbox.get() in ('PNG'):
           suffix = '.png'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as png"
           messagetxt = "Image was saved as png."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('SVG'):
           suffix = '.svg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as svg"
           messagetxt = "Image was saved as svg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        elif self.outputbox.get() in ('JPG'):
           suffix = '.jpg'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as jpg"
           messagetxt = "Image was saved as jpg."
           message = tkMessageBox.showinfo("Info", messagetxt)
        else:
           suffix = '.tiff'
           outputfile=os.path.join(outputpath + suffix)
           ax1.figure.savefig(outputfile, dpi=300)
           print "image was saved as tiff"
           messagetxt = "Image was saved as tiff."
           message = tkMessageBox.showinfo("Info", messagetxt)

    def close_window(self):
        root.destroy()


root = tk.Tk()
app = Application(master=root)
app.master.title(u'Automation of Cartodiagram scales creation in ArcGIS for Desktop')
app.mainloop()
#root.destroy()
```

PŘÍLOHA 7

POČET SOUKROMÝCH PODNIKATELŮ

v krajích České republiky k 31. 12. 2015

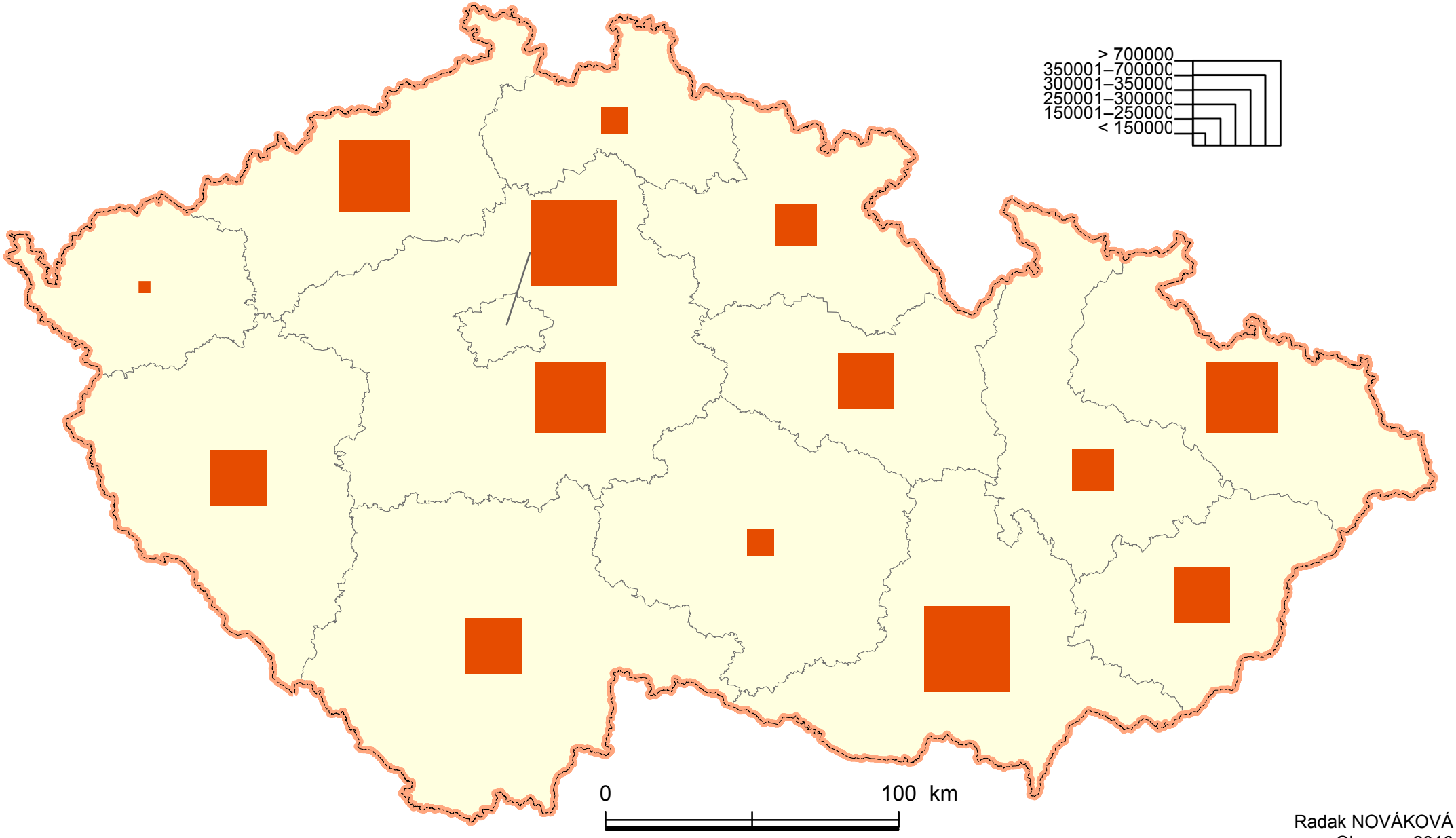


Radak NOVÁKOVÁ
Olomouc 2016

PŘÍLOHA 8

PĚNĚŽITÁ POMOC PŘI MATEŘSKÉ DOVOLENÉ

v krajích České republiky k 31. 12. 2015

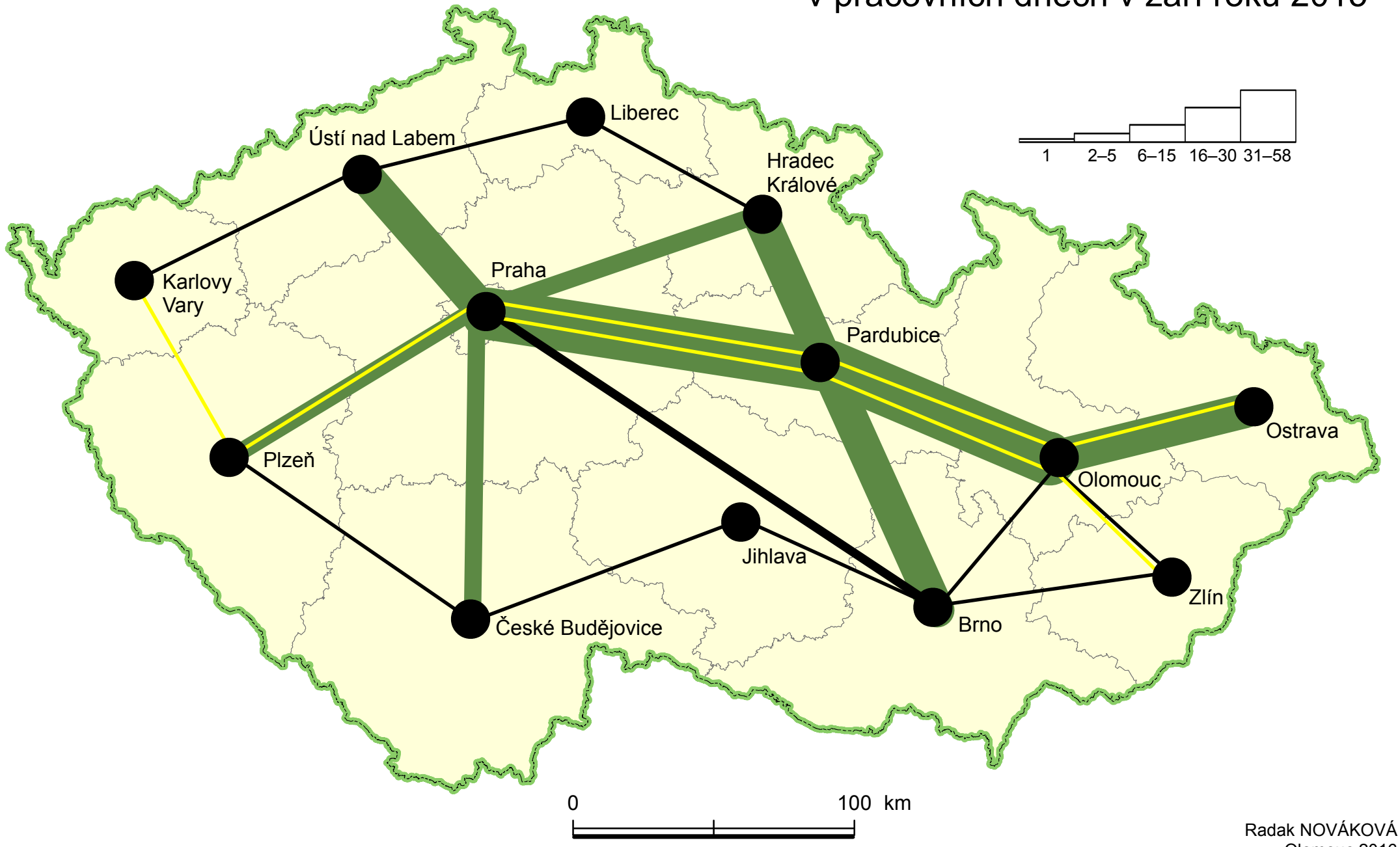


Radak NOVÁKOVÁ
Olomouc 2016

PŘÍLOHA 9

FREKVENCE PŘÍMÝCH VLAKŮ MEZI KRAJSKÝMI MĚSTY

v pracovních dnech v září roku 2015

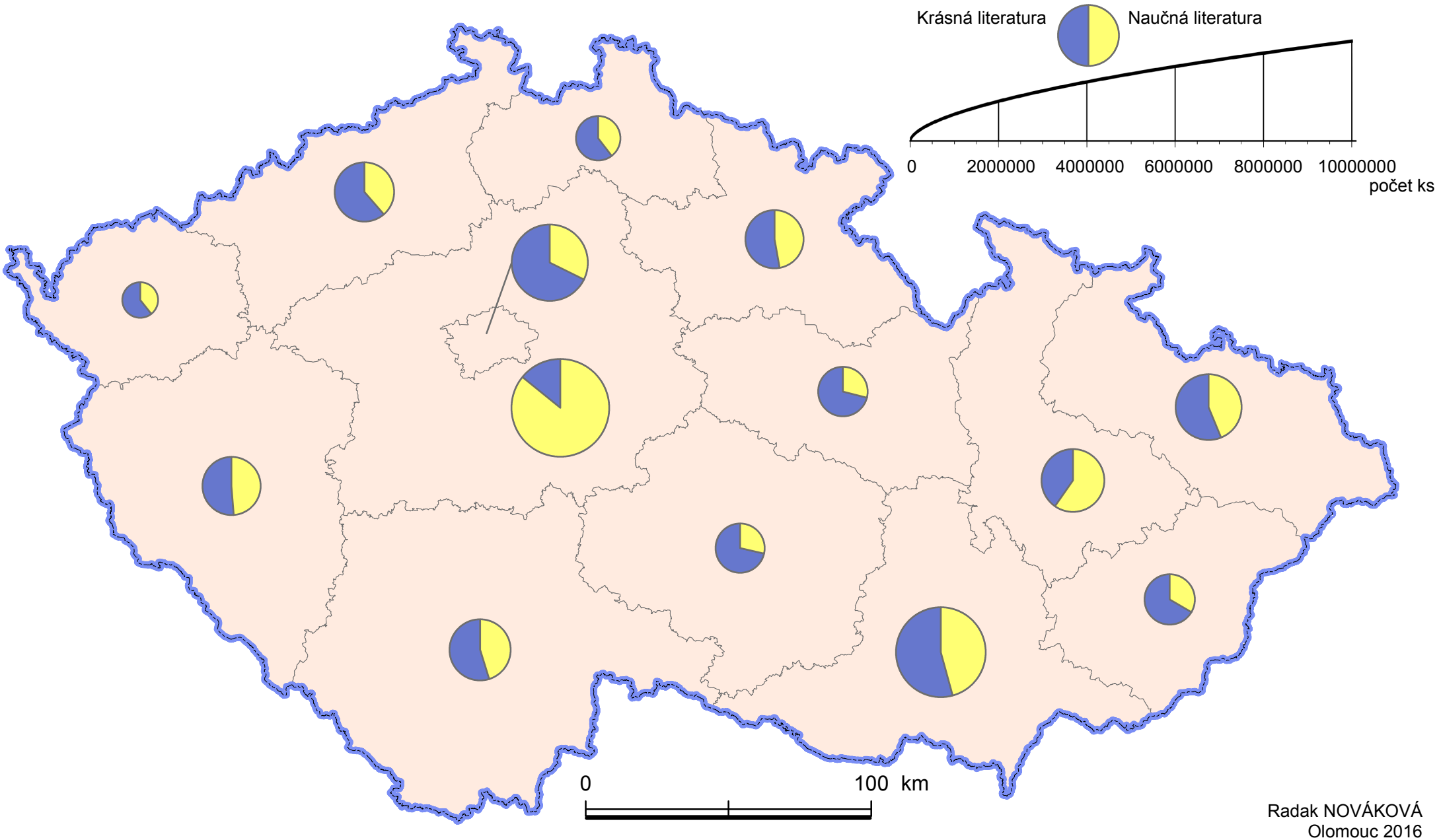


Radak NOVÁKOVÁ
Olomouc 2016

PŘÍLOHA 10

POČET KNIŽNÍCH JEDNOTEK V KNIHOVNÁCH

v krajích České republiky k 31. 12. 2015

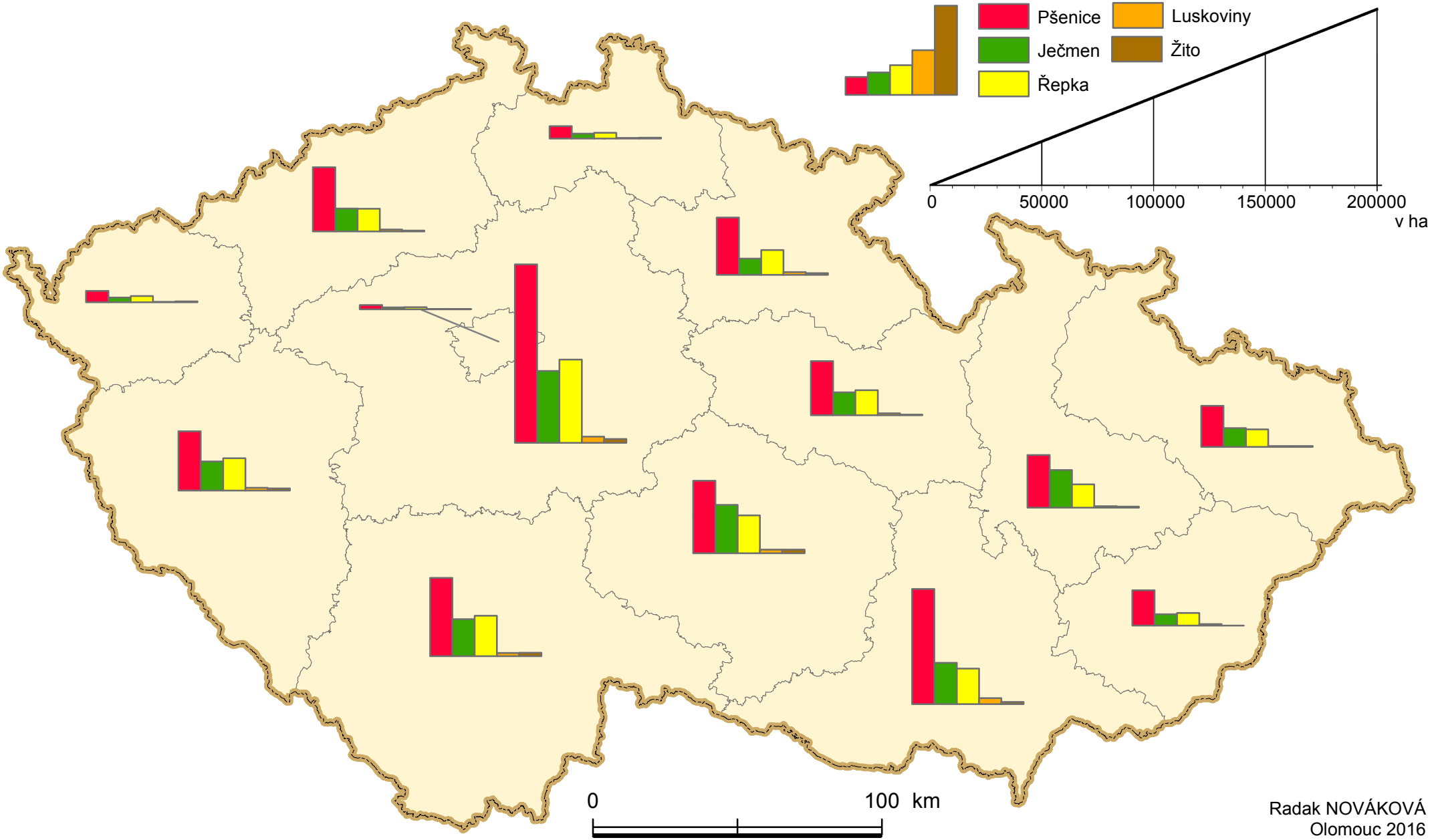


Radak NOVÁKOVÁ
Olomouc 2016

PŘÍLOHA 11

OSEVNÍ PLOCHA ZEMĚDĚLSKÝCH PLODIN

v krajích České republiky k 31. 5. 2015



Radak NOVÁKOVÁ
Olomouc 2016