universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master‘s Thesis

Developing Methods for Population Modelling and Disaggregating Census Data – A Comparison between Day and Night Population

verfasst von / submitted by

Rudolf Churanek, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017 / Vienna, 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 856
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Cartography and Geographic Information Science
Supervisor:	Ass.-Prof. Mag. Dr. Andreas Riedl

Abstract

With the increase of urbanization world wide it becomes more important for urban planners and disaster managers to have data on the small-scale distribution of population at hand. This data is usually not readily available but can be extracted by using different datasets and methods. Especially the calculation of the day-time population is complex since it requires many different auxiliary data in order to model the distribution of every population group during the day whereas the calculation of the nighttime population is comparatively simple as most of the inhabitants are expected at home.
This thesis will examine methods and datasets which are available for the modelling of the day (and night) time population and disaggregating census data, how accurate their results are and how the accuracy can be improved. In order to do this a small-scale population estimation on the level of individual buildings for Vienna (2013) was done considering daytime and nighttime population, including different age groups and evaluating different methods and data. Furthermore, the results are compared to the daytime population grid 2013 of Statistik Austria.

Mit zunehmender Urbanisierung weltweit wird es für Raumplaner und Disaster-Manager immer wichtiger, Informationen über die kleinräumige Bevölkerungsverteilung zu haben. Diese Daten sind allerdings meist nicht in der benötigten Genauigkeit verfügbar, können aber Abgeleitet werden unter Zuhilfenahme verschiedener Datensätze und Methoden. Vor allem die Berechnung der Tagesbevölkerung erweist sich als komplex, da verschiedene Hilfsdaten benötigt werden, um die Verteilung der Bevölkerungsgruppen tagsüber möglichst genau zu Modellieren. Im Gegensatz dazu ist die Nachtbevölkerung einfach zu berechnen, da die meisten Einwohner nachts in den Wohngebäuden vermutet werden.
Die vorliegende Masterarbeit untersucht Methoden und Datensätze, welche für die Bevölkerungsmodellierung (Tages- und Nachtbevölkerung) und Disaggregation von Zensusdaten eingesetzt werden können, diskutiert die Genauigkeit der Berechnung und gibt Empfehlungen zur Verbesserung der Ergebnisse. Hierzu wurde die kleinräumige Tages- und Nachtbevölkerung von Wien (2013) auf Gebäude-Level berechnet und verschiedene Methoden und Datensätze evaluiert. Schließlich wurden die Ergebnisse verglichen mit dem Tagesbevölkerungs-Raster 2013 der Statistik Austria.

Acknowledgement

The author would like to thank especially Dr. Klaus Steinnocher from the AIT Austrian Institute of Technology for his professional support throughout the modelling process and the writing of this thesis as well as DI Mario Köstl, also from the AIT, for providing some of the datasets and discussing them. Also, gratitute is owed to Dr. Gustav Lebhart from the Landesstatistik Wien for the supply of the population data of Vienna in 5 year age groups. Finally, the author wants to thank his supervisor Prof. Andreas Riedl from the University of Vienna and Aļbina Jančuka, MSc. for reviewing the thesis.

Table of Contents

Introduction

The world´s population is becoming increasingly urban with more than half (54%) living in urban areas in 2014. Projections show that this proportion is likely to increase to 66% by 2050. Mega-cities with more than 10 million people are increasing in number and smaller cities are growing rapidly. (cf. UNITED NATIONS 2014) Thus it is becoming more important for many applications like urban planning, vulnerability assessment and risk management to have data at hand about the distribution of population in urban areas in high resolution and up to date. Yet this data is in general, if available, either outdated, generalized or subject to privacy. (cf. TAUBENBÖCK 2010: 143) However, the data´s level of detail is not sufficient for urban and environmental planning, flood control and natural disasters like earthquakes where authorities and aid agencies need to know exactly how many people are affected. This number also depends on the time of day or night the disaster has happened.

The solution to the problem is to estimate the small-scale distribution of population using methods of spatial modelling often involving remote sensing and GIS. This can be done by disaggregating population data from administrative units such as districts to raster cells or, even more accurate, individual buildings. In order to do so, there are several methods available. Basically, what is needed are data on buildings or built-up densities and census data. The analyst can choose between various input data, depending on availability, the method which is used and the scale of the study. This input data can be remote sensing data such as high resolution satellite imagery (e.g. IKONOS) and data from ALS (airborne laser scanning) which can be used to derive land use classes or create a 3D-city model which will be later used to identify individual buildings (cf. TAUBENBÖCK 2010: 67, 68). Alternatively, data on land use / land cover can be directly implemented using e.g. Urban Atlas (cf. FREIRE 2015: 3-5) or CORINE Land Cover data. (cf. STEINNOCHER 2014: 911)

The result of the disaggregation can be seen as the night population due to the fact that the population is expected to be at home during the night. However, during the day the population distribution will be very different as people are at work, school or are commuting resulting in a significant variation in exposure regarding natural hazards compared to the nighttime population. (cf. FREIRE 2015: 1) Therefore, population distribution is not only contingent upon time (work-day/weekend, time of year, time of day) but also upon age – school children are expected to be at school, the working population is distributed among the working places and the elderly are located in nursery homes. Thus, additional data such as company data or data on commuting is needed to sufficiently project the day time population. Since this data is not always available, modelling the daytime population is more complex. (cf. STEINNOCHER 2014: 912, 913)

According to the described predicament the research question of this thesis is: “Which methods and data are available for modelling the day (and night) time population and disaggregating census data, how accurate are their results and how can the accuracy be improved?” In order to show this a small-scale population estimation on the level of individual buildings for Vienna (2013) was done considering daytime and nighttime population and including different age groups and evaluating different methods and data. Furthermore, the results are compared to the daytime population grid 2013 of Statistik Austria.

The following chapters give an overview of existing papers concerning population modelling (1), the datasets needed in population modelling (2), the used method in this thesis and its implementation (3), the results (4) and the comparison with reference data (5).

1 State of the Art

There are already numerous studies about population modelling using different methods and datasets. Besides statistical data about population (e.g. census, balance of commuters) these datasets also include information on land use, building density and building height. Such information is mostly derived from remote sensing imagery and is in part freely available in the European Union. Examples are CORINE Landcover, Urban Atlas, the Imperviousness Layer of Copernicus Land Cover Services or in case of Vienna the Land Use Vienna Layer ("Realnutzungskartierung"). Apart from remote sensing data, auxiliary data like address points and socioeconomic data can also be used for the modelling.

An approach from Aubrecht et al. (2009) describes an object-oriented analysis of different remote sensing data for the modelling of land cover and urban structures, whereas the latter is generalized in a semi-automatic process, delivering the information on individual buildings (use, footprint, height) which are needed in the analysis. Afterwards the population is disaggregated to the buildings weighted by their volume.

Similarly, Taubenböck et al. (2010) uses for identifying individual buildings an automated, transferable and object-oriented approach which consists of the two modules segmentation and classification. Object-oriented means that instead of observing the individual pixels and their spectral properties, clusters of neighbouring pixels (= segments) are being observed. The segmentation is a hierarchical method which generates an overriding spatial unit of apartment blocks which are often delimited by streets and are constituted of similar building structures. The result is a building mask which allows to easily distinguish between urban and non-urban structures. With the help of the digital surface model the height and subsequently the volume of the buildings can be derived. (cf. Taubenböck 2010: 68-69) When this information is available, the population can be disaggregated to the individual buildings. Hereby, the population, e.g. by district (= source zone) is distributed among the buildings (= target zone). This can be done by areal weighting where each source zone contributes to the target zone (e.g. area or volume of the building) a portion of its data proportional to the percentage of its area that the target zone occupies, (cf. Mennis 2006: 180) assuming that the population a) is distributed evenly inside the districts, b) is living in residential buildings and c) the number of residents per building is dependent on the size of the housing space. (cf. TAUBENBÖCK 2010: 145, 146)

Another approach was chosen by Steinnocher et al. (2011) who disaggregated the population on the European level based on building density. The population numbers were available on NUTS 3 units which are small regions for specific diagnoses (the NUTS classification (Nomenclature of territorial units for statistics) is a hierarchical system for dividing up the economic territory of the EU). The building density on the other hand was derived from the Copernicus Imperviousness Layer which captures the spatial distribution of artificially sealed areas, including the level of sealing of the soil per area unit. (cf. Copernicus n.d.) This was done by masking out all sealed surfaces which did not include residential buildings such as streets or industrial buildings. The identification of these areas was done with the help of CORINE Landcover data. Target area of the disaggregation were 500 m raster cells. The relationship between the population density and housing density can be described as

$$Pdens = k \times Hdens \tag{1.1}$$

with $Pdens$ being the population density, $Hdens$ the housing density and k being a factor representing the relationship between population and housing density. The total population of the region is calculated with the formula

$$Pop = \sum_i A_i \times k \times Hdens_i \tag{1.2}$$

where A_i is the area of the housing density i. The underlying assumptions for the approach are a) the population density is proportional to housing density, b) no population resides outside housing areas and c) dependency between population and housing density is constant within a region. (cf. Steinnocher 2011: 3-4)
Martin et al. (2014) suggests a spatio-temporal population modelling where the population is calculated on four different times over a typical work day. Target area of the disaggregation are 200m raster cells. The population is grouped into three categories: residential population, non-residential population and commuters. These groups can be further divided into subgroups depending on age and economic activity whereas the subgroups are over time in constant movement between the three main groups and different areas. The temporal modelling is based on time profiles which exist for different areas (e.g. schools) and represent the activity pattern of the population. They can be replaced anytime in order to simulate different seasons or special occasions such as holidays.
Another method for dynamic population modelling comes from Deville et al. (2014) who use mobile phone data to calculate population densities. Benefits of this approach are the possibility to provide short-term population estimations in case of a disaster and it makes population calculations in poorer countries easier where in many cases exact census data do not exist. Also, mobile phones are common world wide and populations can be calculated over country borders. The anonymised data is provided by mobile phone companies and capture the positions of the involved cell towers as well as time and date per customer and call.
From the references it can be seen that different types of information can be used in order to model the population for a complex system like a city. Table 1.1 gives an overview of such datasets and how they can be grouped. Basically, it can be distinguished between spatial data and socioeconomic data. The former includes data which is mostly acquired through remote sensing like data on buildings/built-up structures (e.g. satellite imagery, ALS) and data on land use/land cover (e.g. CORINE Land Cover, Urban Atlas) whereas the latter originates from statistical sources like census or space-related databases and refer to individual actors like persons or companies. (cf. AUBRECHT et al. 2009: 15)

Data on buildings/built-up structures		spatial data
Very high resolution satellite images	3D-city model	
Airborne laser scanning		
Data on land use/land cover		
CORINE Land Cover		
Urban Atlas		
Imperviousness layer		
Generalized zoning data		
Geospatial data		
Building address data		
Location of hospitals, schools, etc.		
Administrative data (e.g. districts, registration districts)		
Socioeconomic data		socioeconomic data
Census data (population numbers, e.g 5 year age groups)		
data on total employees / unemployed (e.g. from census)		
commuter numbers		
company data (register of companies)		
student numbers		
patient numbers		
data on retirement homes		

Table 1.1: *Overview of data used in population modelling.*

2 Data

This chapter describes the input datasets of the analysis which uses a data driven method for calculating the daytime population meaning that the modelling is compelled by data, rather than by intuition. Therefore, the method relies heavily on the availability and accuracy of the input data. In order to receive results with the best possible accuracy, many different datasets and auxiliary data are needed. Table 2.1 lists the datasets which were chosen as input data for the analysis along with their source, year and type.

Dataset	***Source***	***Year***	***Type***
Data on land use			
Land Use Vienna *(Realnutzungslayer)*	City of Vienna (OGD)	2012	polygon-shp
Data on buildings			
Digital Building Model *(Digitales Baukörpermodell)*	City of Vienna (OGD)/ AIT	2015	polygon-shp
Geospatial data			
Building address data	BEV/AIT	2015	point-shp
Location of hospitals	City of Vienna (OGD)	2017	point-shp
Location of schools	City of Vienna (OGD)	2017	point-shp
Location of universities	City of Vienna (OGD)	2017	point-shp
Location of adult education centres	City of Vienna (OGD)	2017	point-shp
Location of kindergartens	City of Vienna (OGD)	2017	point-shp
Location of retirement homes	City of Vienna (OGD)	2017	point-shp
Public transportation network	City of Vienna (OGD)	2014	line-shp
Districts & Registration Districts of Vienna	City of Vienna (OGD)	2014	polygon-shp
Socioeconomic data			
Population of Vienna per registration district in 5 year age groups	Vienna population register	2013	table
Balance of commuters	Statistik Austria	2012	table
Employees per hospital (from register of companies)	Bisnode Austria/AIT	2015	table
Employees per university & adult education centre (from register of companies)	Bisnode Austria/AIT	2015	table
Register of companies (number of employees)	Bisnode Austria/AIT	2015	table
Employees of Wiener Linien	Wiener Linien	2013	single value
Total number of employees per district	Statistik Austria	2013	table
Total number of unemployed	Job Centre Vienna	2013	single value
Students per school type	Statistik Austria	2012/13	table
Students per university	Statistik Austria	2013/14	table
Retirees in retirement homes (derived from percentage of people in need of care per age group and the amount of them which are in retirement homes)	Federal Statistical Office (Germany)	2013	table
Number of patients (derived from number of beds per hospital and average occupancy rate)	Ministry of Health	2013	table

Table 2.1: *Use case Vienna: input datasets.*

2.1 Data on Land Use

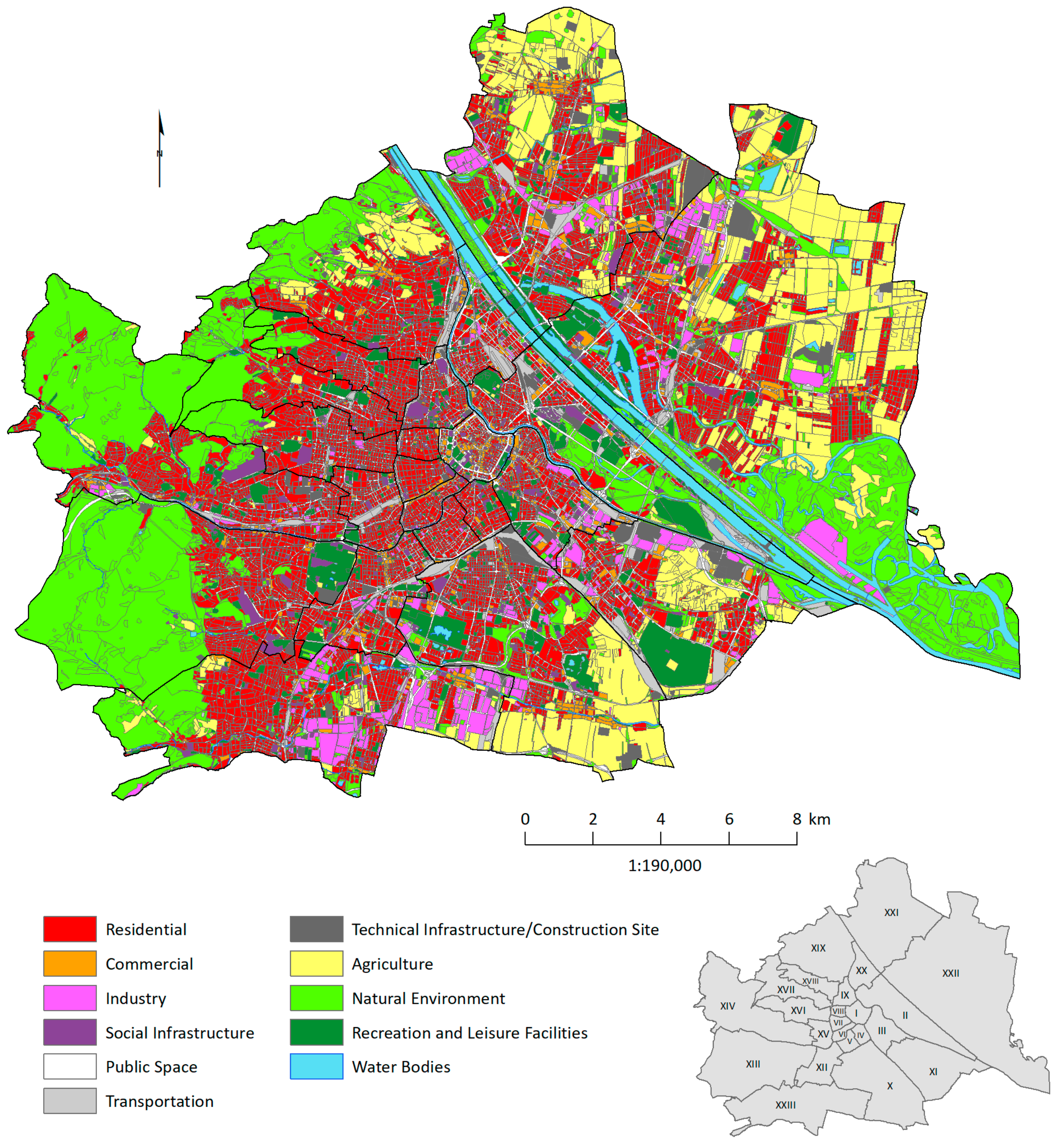

Figure 2.1: *Land Use Vienna 2012. Source: MA 18 (OGD), 2012. Own illustration. Coordinate System: ETRS 1989 LAEA*

The dataset on land use of Vienna in 2012 is based on aerial images which are available since 1981 and shows the effective land use of the city region at the date of the photo flight. Responsible for the assessment is the magistrate division MA 18 (urban development and planning) of the city of Vienna. The dataset is provided by the government on its web portal data.gv.at under creative commons licensing. The generalized data with a scale of 1:10.000 to 1:150.000 is based on aerial photo interpretation of the magistrate division MA 41 (city mapping) using ancillary data. Quality assessment is conducted by divisions of MA 21 (district planning and land use) and MA 18 with a continuation cycle of about two years depending on the availability of new ortho-photographs. The focus of the dataset lies in the land use rather than land cover. The latter (like grassland, buildings) always needs context in order to make assumptions of the land use (like park, sports facilities, single-family housing area). This context is obvious for some classes from the areal image like single-family housing areas while others such as commercial or mixed use areas require ancillary data like data on building heights, small area population and employment statistics in order to identify them. (cf. City of Vienna n.d. a) There is a total of 32 land use categories in level 3, eleven categories in level 2 and three in level 3 (building land, pasture land and transport). Figure 2.1 shows the land use of Vienna for 2012 adopting these categories while Table 2.2 shows the total area each category consumes and its share compared to the total area of the city. It can be seen that both natural and residential areas each make up 25% of Vienna´s surface followed by agriculture with 14%. The share of public space and transportation surfaces combined is also 14%, commercial and industrial areas amount to 6%. This equals the share of recreation and leisure facilities which mainly consist of parks and sports facilities. Regions of agriculture are dominant especially in the northern (Kahlenberg) and eastern (Marchfeld) parts of the city as well as in the south whereas natural environments are prevalent in the east (Lobau) and west (Lainzer Tierpark, Wiener Wald) but can also be found in the north (Hermannskogel, Kahlenberg, Leopoldsberg) and near the city centre (Prater). Districts with the highest share of agricultural surfaces compared to their total area are districts 10 (29%), 22 (27%) and 21 (26%). Parks are spread over the whole city with the biggest ones being rather near its edges (e.g. Vienna Central Cemetery, Schlosspark Schönbrunn, Wienerberg). Industrial areas can be found mainly in the south (especially in the 23rd district but also in the 3rd and 11th district) and the northeast (21st and 22nd district). Finally, a small industrial area is located in the west (Auhof). Mixed use commercial areas are often located at shopping streets. Especially Vienna´s 1st district has a high share of commercial areas (15%) out of the total area (compared to 22% residential areas), also partly due to the high density of shopping streets (e.g. Kärntner Straße, Graben, Rotenturmstraße, Kohlmarkt). Outside of the 1st district commercial areas can be found at shopping streets such as Mariahilfer Straße and Meidlinger Hauptstraße but also at Kaisermühlen (Uno City) or Siemensstraße with many R&D-facilities.

Land Use	***Area [km²]***	***Area [%]***
Natural Environment	104.77	25.25
Residential	104.01	25.07
Agriculture	59.10	14.25
Public Space	44.82	10.80
Recreation and Leisure Facilities	25.02	6.03
Water Bodies	19.13	4.61
Industry	15.24	3.67
Transportation	14.54	3.50
Commercial	9.71	2.34
Social Infrastructure	9.43	2.27
Technical Infrastructure/Construction Site	9.11	2.20
Total	**414.87**	**100**

Table 2.2: *Total area per land use class. Source: MA 18 (OGD), 2012.*

Residential areas are constituted of buildings whose primary use is housing but they can also include shops and businesses. For this reason some shopping streets like Währinger Straße can´t be identified as such in fig. 2.1 because the primarily use of its buildings is residential. The biggest percentage of the residential areas in a district compared to its total area can be found in district 8 with 61% followed by districts 7 and 5 with 57% and 56%. The lowest proportion on the other hand can be found in district 2 with 14%. Buildings whose use is “Social infrastructure” can include, among others, hospitals, universities, schools, churches and theatres. The class “transportation“ mainly consists of stations and rail tracks with the largest being the central marshalling yard of Vienna (36 ha), the central workshop of Wiener Linien (29 ha) and the freight railway station Nordwestbahnhof (24 ha). More diverse is the class “Technical Infrastructure/Construction Site“, ranging from landfill and disposal sites (with the biggest having an area of 59 ha) over Vienna´s main sewage treatment facility (32 ha) to facilities of the Austrian military and building sites. The main water bodies in Vienna are the Danube river which divides the city, the Old Danube which is separated from the river and the Danube canal which branches from the main river and runs near the city centre.

2.2 Data on Buildings - The Digital Building Model

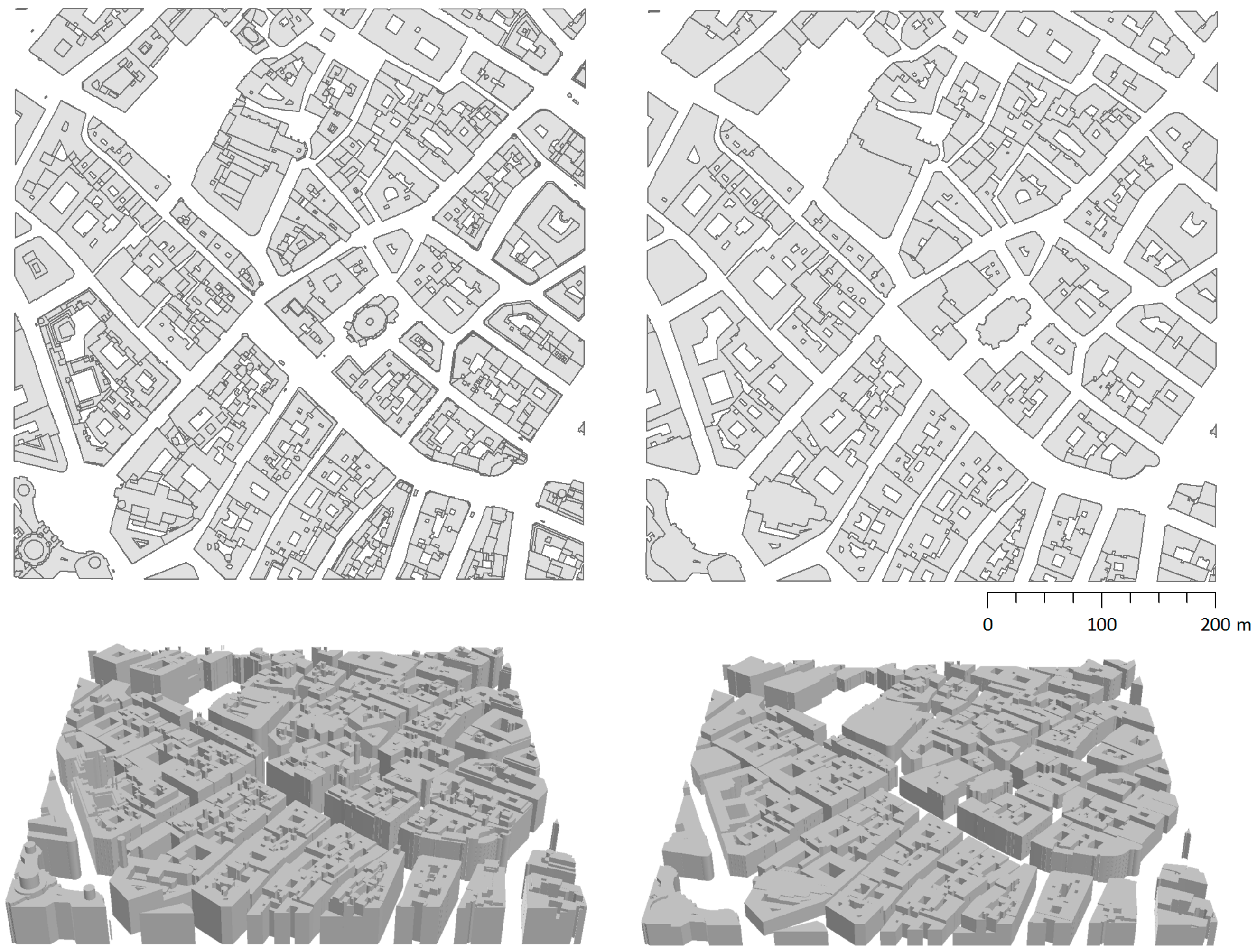

Figure 2.2: *Digital Building Model - a comparison.* ***Left****: Original dataset (2D&3D),* ***Right****: Generalized dataset (2D&3D) which was used for the analysis. Source: MA 41, 2015. Own illustration. Coordinate System: ETRS 1989 LAEA*

The digital building model depicts all buildings of Vienna as prisms (also called building structures). The original dataset contains about 200.000 buildings and more than 650.000 building structures since one building can be comprised of several structures (see Figure 2.2, left). However, for the population modelling it is necessary to have a more simplified model where every building consists of only one structure in order to allocate people specifically to one building using a weighting factor (in this case the volume of the building). The problem of the detailed dataset lies in the fact that every prism of a building has its origin at the ground (digital terrain model) so the sum of the volumes of each prism is higher than the actual volume of the building. Therefore, the dataset which was used in the analysis had to be generalized (see Figure 2.2, right) so that there is only one prism per building. This generalized dataset was provided by the AIT Austrian Institute of Technology and was also linked with the land use data from chapter 2.1. The original dataset can be found on data.gv.at under creative commons licensing. Responsible for the assessment is the magistrate division MA 41 (city mapping). It is updated continuously. (cf. City of Vienna n.d. b)

Generation of the original dataset

For every building there exists height information which is extracted through aerial photo evaluation measuring each prism. For superstructures and big passages additional height data is gathered through terrestrial data capturing. The accuracy of the measurements is +/- 25 centimetres. For the generation of the dataset the buildings are separated in individual parts according to their height structure. As mentioned before each of the building parts is represented through a prism (building structure) which is placed on the digital terrain model. The layout for each building space comes from the area-multi-purpose map (Flächen-Mehrzweckkarte) which is a digital city map of Vienna containing small-scale land use information about the whole city. (cf. City of Vienna n.d. b, c)

A description of the generalized dataset

The generalized dataset consists of 247,560 prisms, each representing one building. Figure 2.3 and Figure 2.5 show the use class of every structure of the digital building model for the whole city and also a more detailed extract from the centre. Because of its history, like most of Middle European cities, Vienna is structured radial-concentric with geographic features having influenced the development of the city. For example, today´s Danube canal was a branch of the Danube which was flowing directly along the city centre in the Middle Ages. Moreover, streams of the Vienna Woods which were flowing into the Danube shaped axes along which streets and further settlements developed. (cf. MA18 2005: 41) As mentioned in chapter 2.1 the high density of buildings with the class "Commercial" in the 1st district can also be seen in the digital building model. Figure 2.4 shows that out of the 247,560 buildings there are 214,270 whose main use is residential which is 87% of all buildings. 26% (56,033) of that amount consist of very small buildings with an area of up to 30m^2 and a height of up to 5m. Then, far behind come the use classes "Commercial" (9,021 buildings, 4%) and "Industry" (5,569 buildings, 2%). Buildings without a use class (No data) are those that are located (with most of their surface) outside of the city boundary and therefore outside of the Land Use 2012 layer from where those classes are derived. This is a characteristic of the dataset which does not interfere with the analysis in a serious way since it affects only 44 buildings (0.02%). The same goes for buildings with the use class "Water Bodies". They are located closely to real water bodies and are mistakenly classified as such by the Land Use 2012 dataset. When looking at the mean volume per building of each use class (see Table 2.3) it can be seen that buildings with the class "Social Infrastructure" have by average the biggest volume followed by the class "Industry" with the largest buildings by volume being the General Hospital of Vienna (AKH Wien, main building - social infrastructure), the Vienna University of Economics and Business (WU Wien main building until 2013, incl. bio centre of University of Vienna - social infrastructure) and the Opel Wien GmbH (industry).

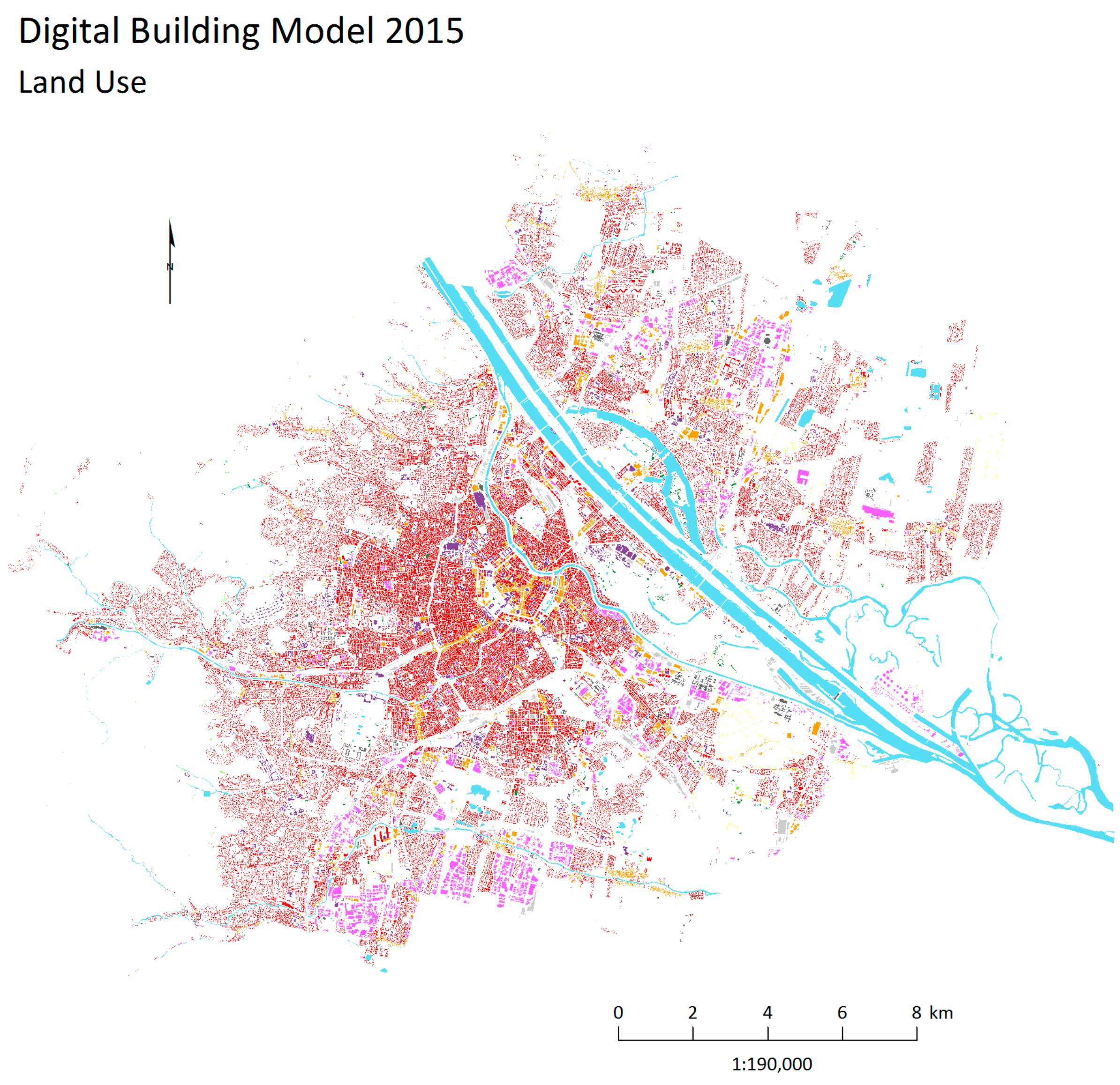

Figure 2.3: *Digital Building Model of Vienna - Land use. The depicted water bodies are from the dataset Land Use Vienna 2012. Source: MA 41, 2015; water bodies: MA 18 (OGD), 2012. Own illustration. Coordinate System: ETRS 1989 LAEA. Legend see Figure 2.5.*

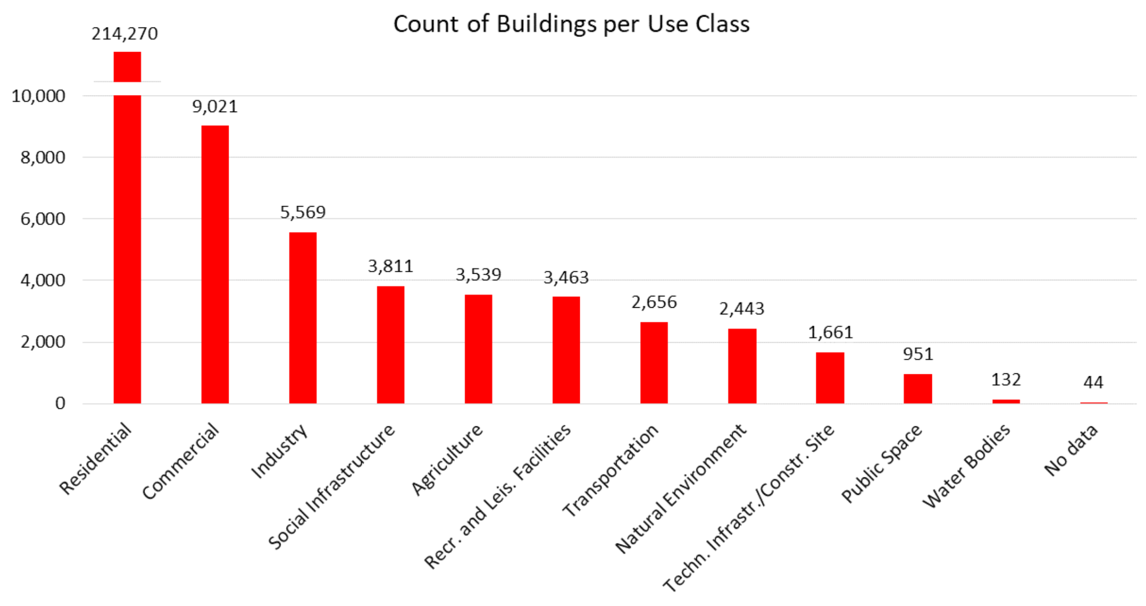

Figure 2.4: *Count of buildings per use class. Source: MA 41, 2015. Own illustration.*

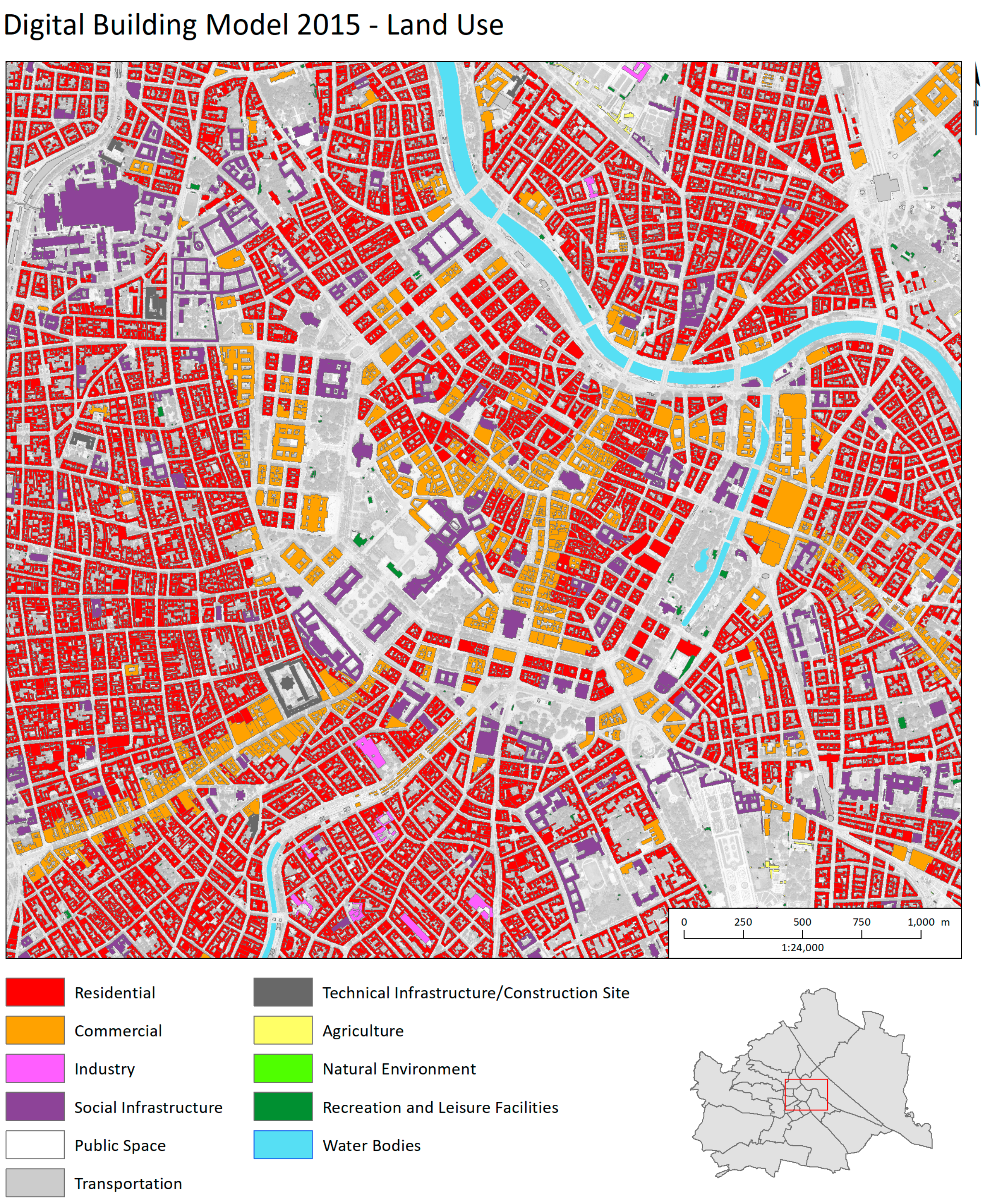

Figure 2.5: *Digital Building Model of Vienna (detailed) - Land use. The depicted water bodies are from the dataset Land Use Vienna 2012. Source: MA 41, 2015; ortho photo: MA 41, 2016; water bodies: MA 18 (OGD), 2012. Own illustration. Coordinate System: ETRS 1989 LAEA.*

Residential buildings which are the most frequent have only an average volume of 1,933 m³ per building. Finally, Figure 2.6 shows the percentage of the area covered by buildings per district compared to the total area of each district. According to the data the most dense districts are district 7 (Neubau) and 8 (Josefstadt) with 53% resp. 50% of its area covered by buildings whereas the least dense districts are district 13 (Hietzing, 6%), 22 (Donaustadt, 7%) and 14 (Penzing, 9%).

Use of Buildings	***Volume / Building [m³]***	***Count***
Social Infrastructure	12,052	3,811
Industry	9,273	5,569
Technical Infrastructure/Construction Site	6,955	1,661
Commercial	6,828	9,021
Transportation	6,001	2,656
Residential	1,933	214,270
Recreation and Leisure Facilities	968	3,463
Agriculture	589	3,539
No data	577	44
Transportation	446	951
Natural Environment	270	2,443
Water Bodies	156	132
Total	**46,049**	**247,560**

Table 2.3: *Average volume per building. Source: MA 41, 2015.*

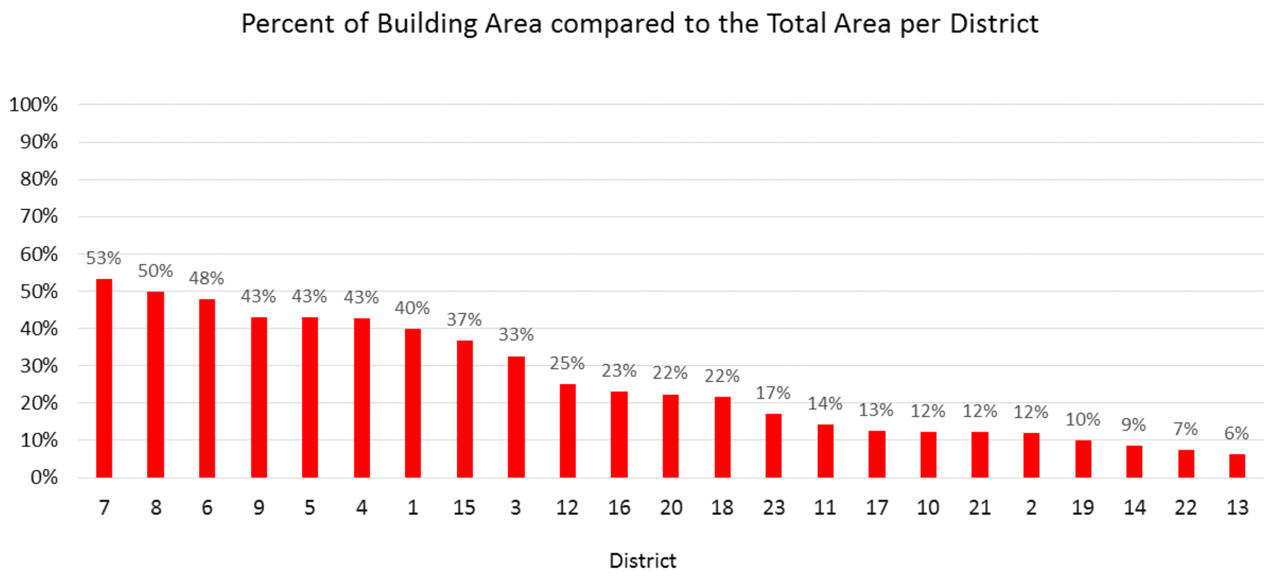

Figure 2.6: *Percent of Building area compared to the Total Area per District. Source: MA 41, 2015. Own illustration.*

2.3 Geospatial Data

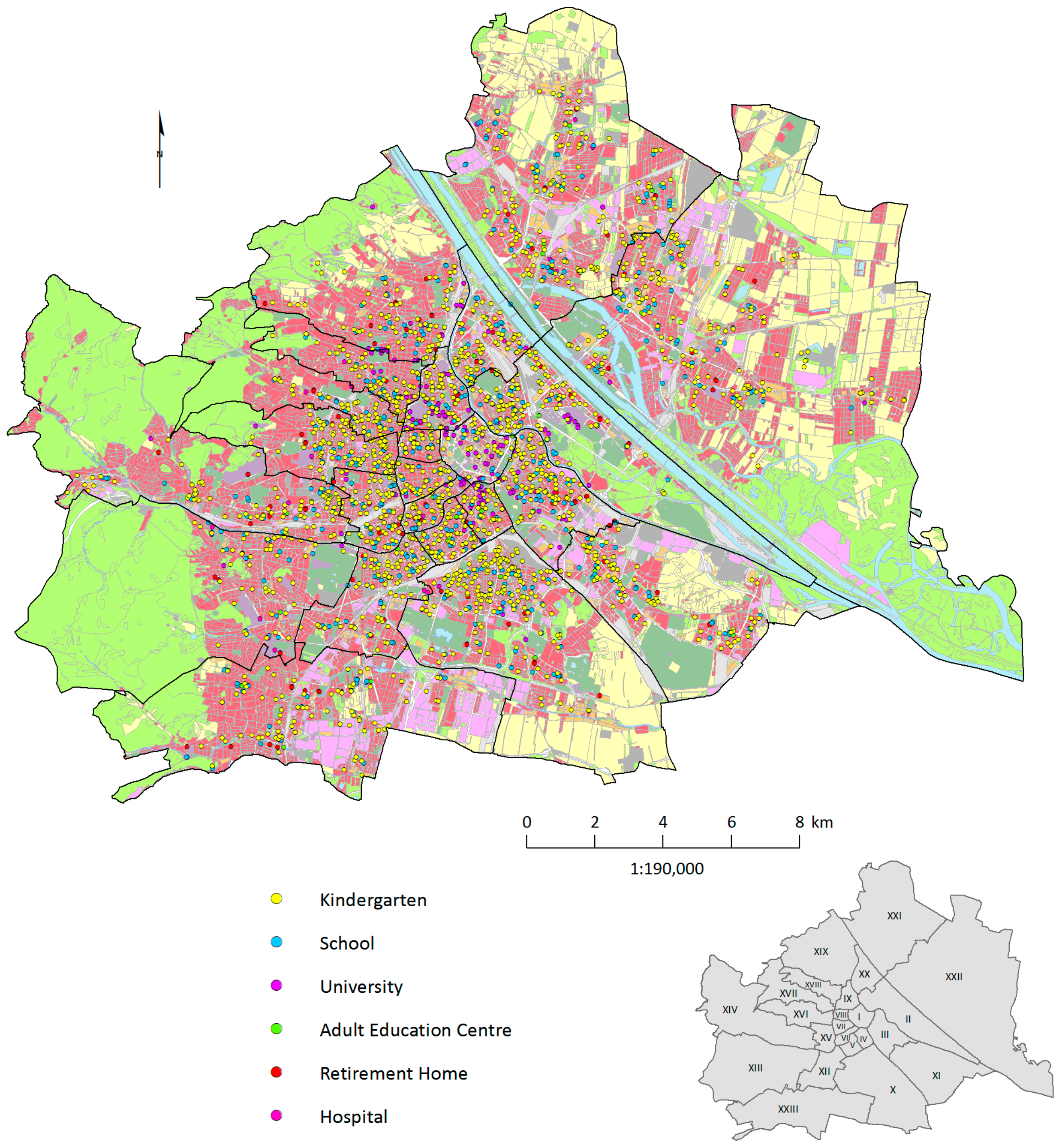

Figure 2.7: *Geospatial datasets (excluding address points and public transportation). Source: kindergarten: MA 11 (OGD), 2017; school: MA 56 (OGD), 2017; retirement home: Fonds Soziales Wien (OGD), 2017; university: Universities of Vienna (OGD), 2017; adult education centre: MA 13 (OGD), 2017; hospital: KAV (OGD), 2017; land use: MA 18 (OGD), 2012. Own illustration. Coordinate System: ETRS 1989 LAEA.*

The geospatial datasets depict the location of facilities which are relevant to the analysis as well as the administrative boundaries. The facilities are mainly represented through point-shapefiles and include kindergartens, schools, universities, adult education centres, retirement homes, hospitals and building address points as well as the public transportation network which is represented through line-shapefiles. All data is provided by the government on its web portal data.gv.at under creative commons licensing. Figure 2.7 shows all datasets except address points and the transportation network.

Point-Shapefiles - Location of Facilities

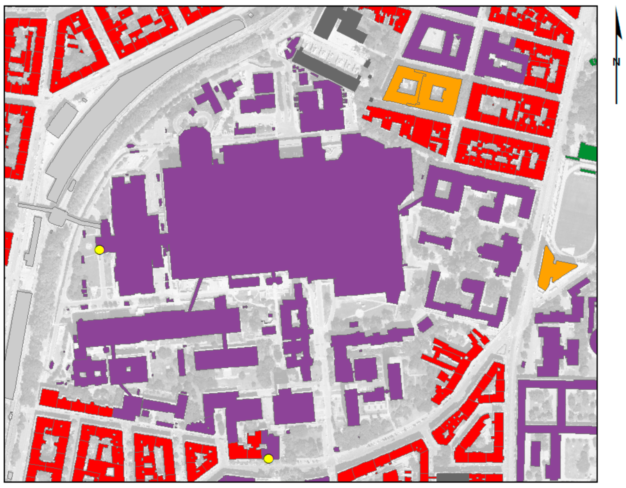
Figure 2.8: *Example: Point-shapefile (hospitals) & the digital building model. Scale: 1:10,000*

The point-shapefiles with the information of the location of certain types of facilities play an important role in the population modelling since they are the places where certain age groups of people reside during the day. Each point lies within a building structure of the digital building model and its information can be assigned to the building with the spatial join tool. Usually one point represents e.g. one school or kindergarten. Universities differ from this pattern because one point represents one institute of a university. It includes also the type of university so for example all students of a university can be distributed to its institutes weighted by their volume. The university types were summarized as following: University of Vienna, Technical University of Vienna, Vienna University of Economics and Business, Medical University of Vienna and Other University. The available dataset with the hospitals however is suboptimal since every point represents one hospital but does not account for its buildings with the problem being that most hospitals are comprised of several structures. This can be seen in Figure 2.8 which shows the General Hospital of Vienna and its surrounding buildings as well as the location of its designated point from the point-shapefile (being the top left point; the bottom point in the middle belongs to another hospital which consists of only one building). From looking at the figure it should be clear that it is impossible to select all the buildings of a hospital automatically by using the point-shapefile for the reason that sometimes hospitals can be in close proximity to one another as well as other buildings with the use “Social Infrastructure“. Because of this the hospital buildings had to be selected before the analysis manually using the point-shapefile as reference and the name of their hospital resp. a distinct hospital ID had been assigned to them.
Another noteworthy editing had been done with the point-shapefile containing schools which were summarized into four types: compulsory school, grammar school, trade school and other upper secondary school so that the students can be distributed according to the socioeconomic dataset “Students per school type“ (see page 25).

When looking at the count of each facility type it shows that the most frequent are kindergartens with an average of 79.3 per district followed by schools (27.4 per distr.) and university-institutes (7 per distr.). The least frequent are hospitals with 1.5 per district (see Table 2.4). Most university-institutes can be found in districts 1 (Innere Stadt, 28), 9 (Alsergrund, 27) and 4 (Wieden, 25) which are comparatively small and near the centre of Vienna. Together they house 49% of all institutes.

	Count	***Average per District***
Kindergarten	1,825	79.3
School	629	27.4
University (Institutes)	162	7.0
Retirement Home	138	6.0
Adult Education Centre	46	2.0
Hospital	34	1.5

Table 2.4: *Count of Facilities. Source: City of Vienna, 2017.*

Building Address Points

The building address points are a auxiliary dataset and are provided by the Federal Office for Metrology and Surveying (*Bundesamt für Eich- und Vermessungswesen*) in the form of a csv-table with the source being the Austrian Address Register; reporting date: 15.07.2015. Like the digital building model a new dataset was derived from the original data and provided for the analysis by the Austrian Institute of Technology. This dataset (the point-shapefile) was generated from the coordinates of the original csv-table and the addresses were summarized in one column containing the postcode, street name and street number (e.g. 1100 Alaudagasse 40). In this form the addresses can be distinctively identified and joined with the register of companies to get the number of employees. Through the position of the address points that number can then be assigned to the buildings.

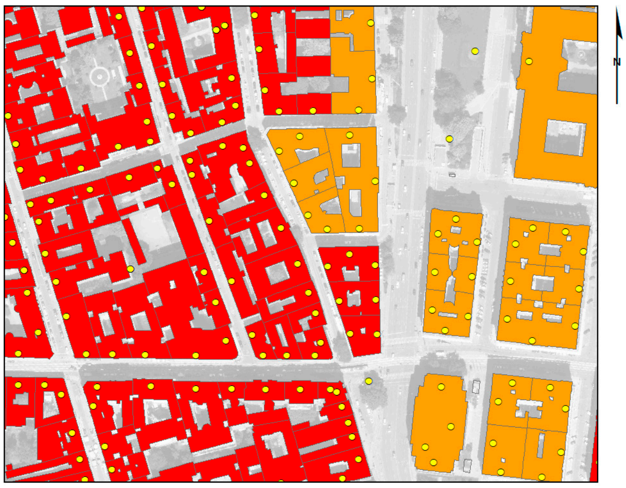
Figure 2.9: *Building Address Points. Scale: 1:5,000*

The address point shapefile consists of 265,111 address points (compared to 247,560 buildings from the generalized digital building model) with an average of 11,527 points per district. Nonetheless, there are only 140,368 distinct addresses with some being represented up to 745 times which means there can exist up to 745 points for one address. The most abundant ones are located usually in single-family house areas as shown in Figure 2.10. This is the most extreme example where the two green points represent in total 745 individual points with the same address which are stacked up one another. This mainly concerns buildings with the use “Residential“ but also some buildings with the use “Commercial“ or “Industry“ which contain several points with the same address as well as points with different addresses. The latter occurs mostly in corner houses which can be seen in Figure 2.9. This was handled by dissolving the points with the same address before joining the data from the company register and afterwards using the join operation “join one to many“ of the spatial join tool. Another feature of the dataset is the fact that some points are located outside of the buildings in various distances which can also be seen in Figure 2.9. In this case if an address from the company register matches such an address point the data will not be included in the model. In total there are 60,861 (25%) points which lie outside of buildings meaning that only 186,699 points are potentially joined with the company register. From this quantity there exist 1,263 matches with the company register and lastly 1,153 matches with the digital building model, distributing a total of 114,795 employees. The biggest problem with the dataset however, is the circumstance that there are some points which share the same address but lie in different buildings. This concerns a total of 13,334 points which are sharing 5,676 different addresses. Instead of checking manually which address point lies in the correct building the approach was to delete the matching addresses from the company register in order to avoid distributing employees of one company repeatedly to different buildings where they are not belonging to. Since this affected only 49 (1.4%) of originally 3,417 entries the loss of data is acceptable.

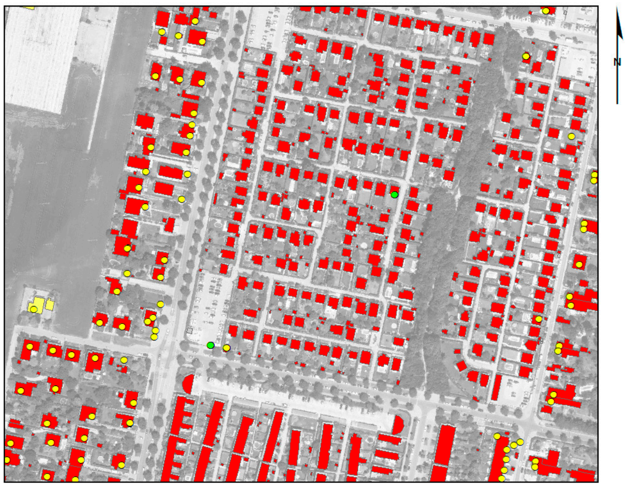
Figure 2.10: *Building Address Points. Scale: 1:5,000*

Public Transportation Network

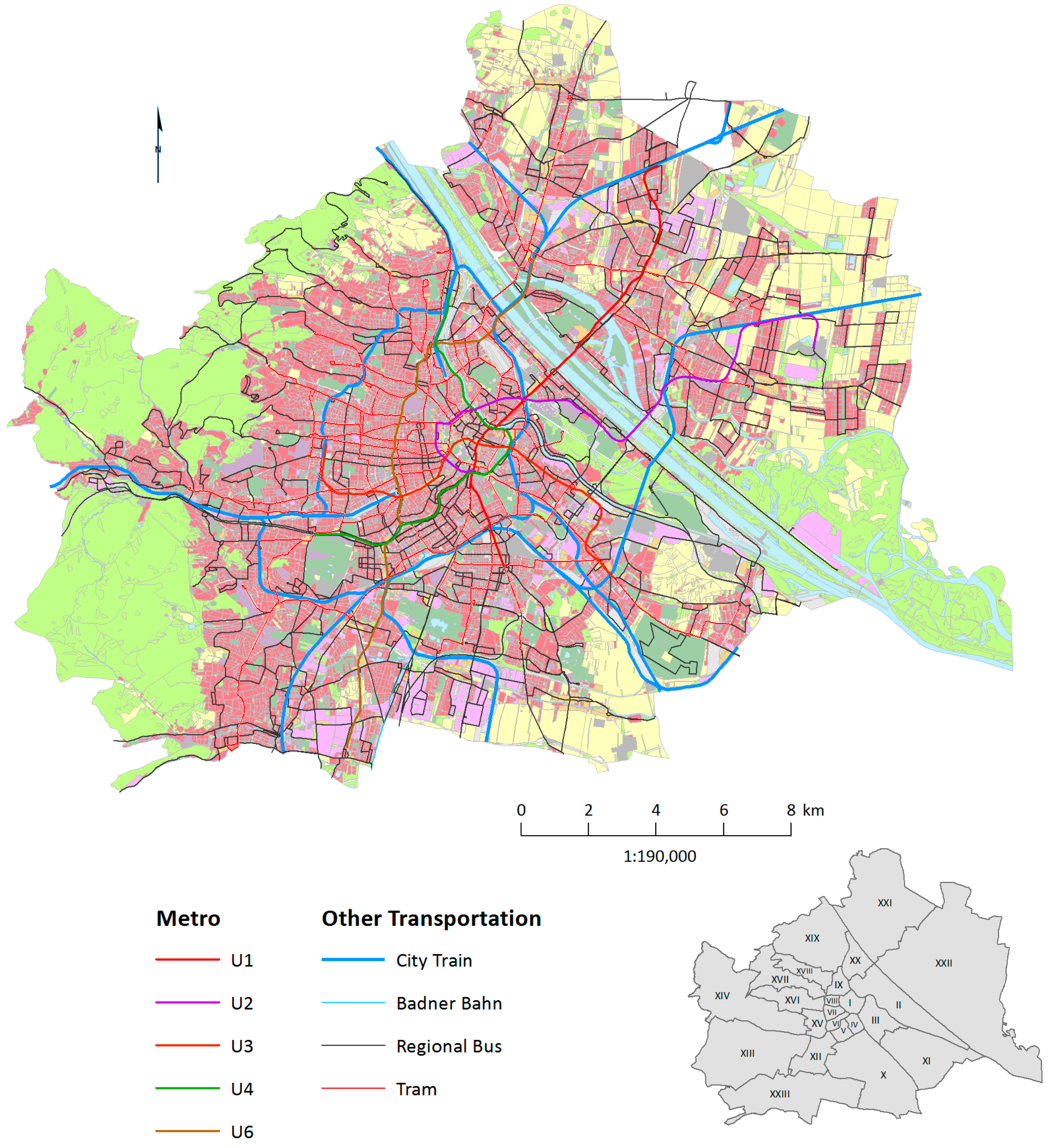

Figure 2.11: *Public Transportation Network of Vienna. Source: network: Wiener Linien (OGD), 2014; land use: MA 18 (OGD), 2012. Own illustration. Coordinate System: ETRS 1989 LAEA.*

The public transportation network of Vienna is also provided by the city of Vienna with the responsible entity being the Wiener Linien GmbH & Co KG. The dataset is constituted of 1,306 km of public transportation lines which are divided into different types (see Figure 2.11). It was used in the calculation of the daytime population to distribute the employees of the Wiener Linien (which are approx. 4,289 during the day considering shift work) to their "work place" which is basically all over Vienna. This was done by calculating a buffer of 2.5m around the lines and then erasing all parts of the buffer which lie outside of Vienna due to the fact that 20 km of the lines lie outside of the city boundaries.

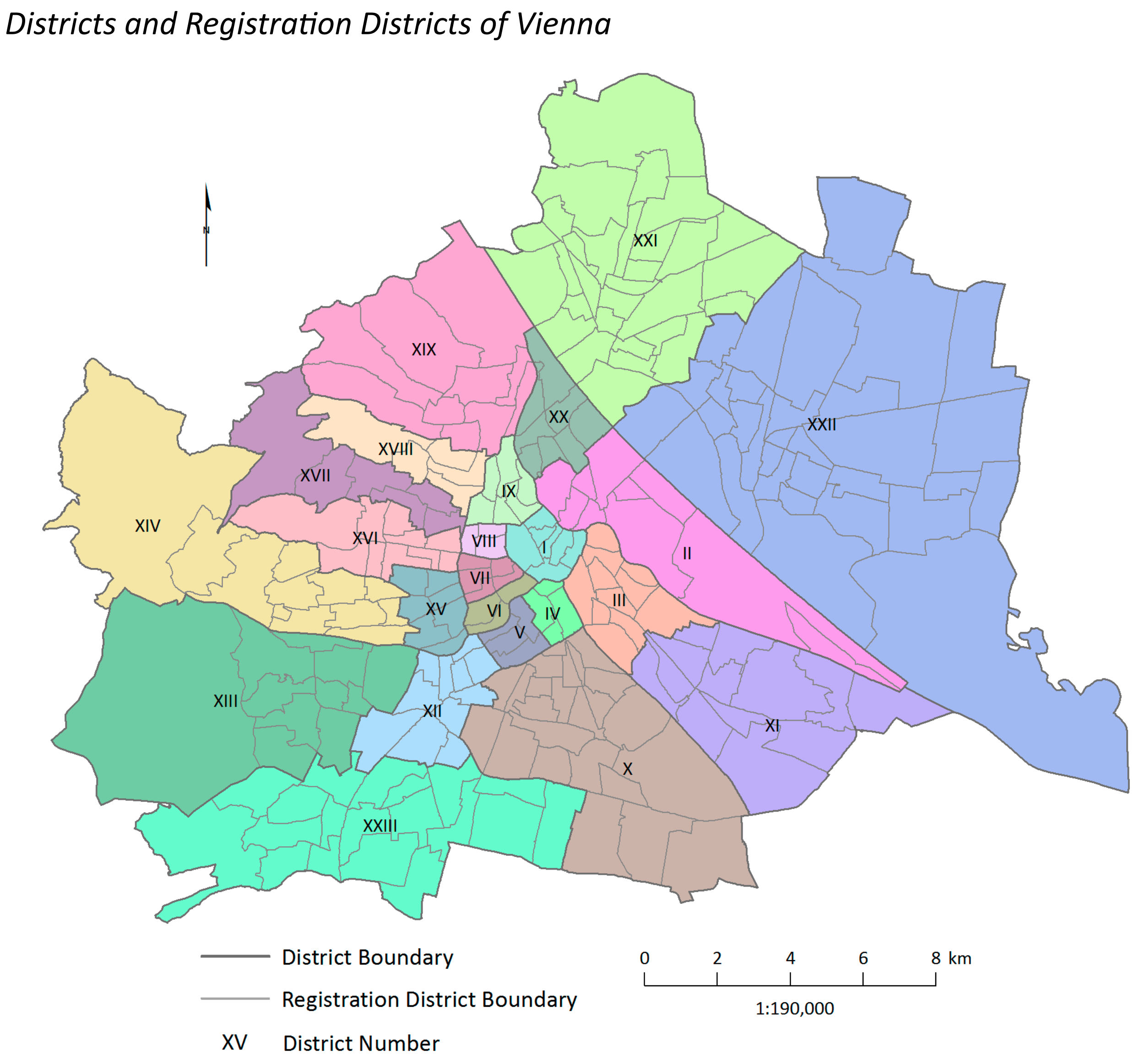

Figure 2.12: *Districts and Registration Districts of Vienna. Source: MA 21, 2014. Own illustration. Coordinate System: ETRS 1989 LAEA.*

The polygon-shapefile of the districts and registration districts of Vienna from 2014 builds the basis of the population modelling since it provides the spatial units which are used to select the buildings to which the population is distributed. Whether the districts, registration districts or the whole of Vienna are used depends on the age group which is distributed. The dataset contains 250 registration districts which have a distinctive ID like 9230115 where the 9 stands for Vienna, 23 for the district and 0115 for the registration district. Through this ID it was possible to derive the 23 districts. The source of the dataset is the magistrate division MA 21 (district planning and land use) while it is provided by the city of Vienna under creative commons licensing.

2.4 Socioeconomic Data

The Socioeconomic datasets are a diverse group of auxiliary data which are used to improve the results of the population modelling. They include, among others, residential population numbers, commuting, employment and students. In contrast to the previously described datasets the socioeconomic ones are mainly tables and two single values with the exception being the data on districts and registration districts of Vienna which are a polygon-shapefile.

Population per Registration District

The data on the population per registration district in 5 year age groups with the reporting date being the 1.1.2013 on the other hand comes from the Vienna population register and is provided by the magistrate division MA 23 (economics, work and statistics). Normally, this dataset is subject to a charge but it was kindly allocated by Mag. Dr. Gustav Lebhart from the MA 23 for this thesis in the form of a csv-table. During the data preparation for the analysis the data was correlated to the districts and registration districts shapefile via the registration district ID. Afterwards, the population from the registration districts could be distributed among the buildings of the digital building model. However, this is only half true as only the age groups 0-4 and 65+ were distributed from the registration districts whereas the population numbers of the other age groups were taken from the other auxiliary data described later in this chapter. Table 2.5 shows the area, the summarized population of Vienna over all age groups and the number of registration districts per district.

District	***Area [km²]***	***Population 2013***	***Number of Registration Districts***
1., Innere Stadt	3.01	16,779	7
2., Leopoldstadt	19.27	98,207	10
3., Landstraße	7.45	87,131	11
4., Wieden	1.80	31,744	4
5., Margareten	2.03	53,937	4
6., Mariahilf	1.48	30,336	3
7., Neubau	1.61	31,040	5
8., Josefstadt	1.08	24,469	3
9., Alsergrund	2.99	40,956	6
10., Favoriten	31.80	182,119	23
11., Simmering	23.21	92,858	13
12., Meidling	8.16	90,420	11
13., Hietzing	37.69	51,253	11
14., Penzing	33.82	86,817	12
15., Rudolfsheim-Fünfhaus	3.86	74,936	7
16., Ottakring	8.65	98,677	10
17., Hernals	11.33	54,296	6
18., Währing	6.30	49,218	5
19., Döbling	24.90	70,267	10
20., Brigittenau	5.67	85,099	8
21., Floridsdorf	44.51	146,889	30
22., Donaustadt	102.24	165,808	32
23., Liesing	32.02	95,632	19
Total	**414.88**	**1,758,888**	**250**

Table 2.5: *Population of Vienna in 2013. Source: Vienna population register, reporting date 1.1.2013, Calculation: MA 23.*

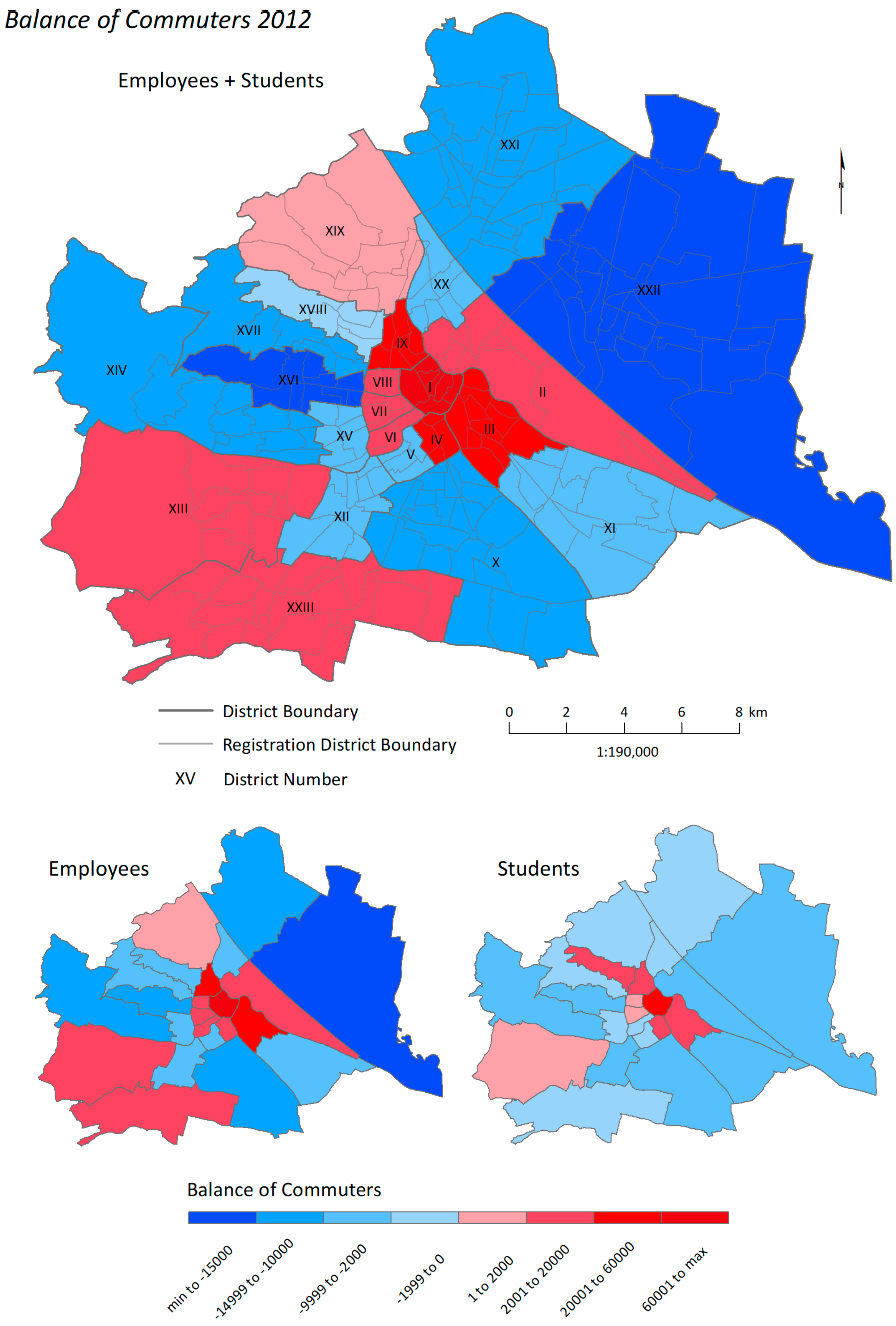

Figure 2.13: *Balance of Commuters 2012. Source: Statistik Austria; reporting date: 31.10.2012; territory information: 2014; created on 6.11.2014. Own illustration. Coordinate System: ETRS 1989 LAEA.*

In order to calculate the daytime population of Vienna correctly it is necessary to know the total number of people who stay in Vienna during the day. This number is constituted of the residents of Vienna (1,758,888) from the data on the population per registration district in 5 year age groups plus the balance of commuters (207,662) which amounts to 1,966,550. The source of the original dataset from the year 2012 is Statistik Austria. It includes the inbound commuters from both other districts and outside of Vienna and the outbound commuters to both other districts and outside of Vienna per district for employees and students. From these numbers the balance of commuters could be calculated by subtracting the outbound commuters from the inbound commuters per district. However, the only relevant number from this dataset is the balance of commuters over all districts (207,662) which was used to get the total daytime population of Vienna. Since the datasets on employees and students do not include every person (e.g. not all unemployed persons might be registered as unemployed) it was necessary to calculate and distribute the rest of the population after all other distributions were done. Figure 2.13 and Table 2.6 give an overview on the balance of commuters of Vienna in the year 2012. The data has been linked to the registration district shapefile.

District	***Balance of Commuters Employees***	***Balance of Commuters Students***	***Balance of Commuters Total***
1., Innere Stadt	103,556	38,726	142,282
2., Leopoldstadt	17,362	-3,068	14,294
3., Landstraße	55,511	3,688	59,199
4., Wieden	13,450	11,134	24,584
5., Margareten	-4,884	-1,385	-6,269
6., Mariahilf	13,386	-211	13,175
7., Neubau	15,749	593	16,342
8., Josefstadt	4,565	579	5,144
9., Alsergrund	34,243	8,387	42,630
10., Favoriten	-11,145	-2,640	-13,785
11., Simmering	-5,634	-4,153	-9,787
12., Meidling	-3,214	-3,653	-6,867
13., Hietzing	4,897	222	5,119
14., Penzing	-10,005	-3,076	-13,081
15., Rudolfsheim-Fünfhaus	-3,033	-1,118	-4,151
16., Ottakring	-13,871	-4,428	-18,299
17., Hernals	-9,628	-1,790	-11,418
18., Währing	-6,789	5,017	-1,772
19., Döbling	1,934	-213	1,721
20., Brigittenau	-7,379	-1,871	-9,250
21., Floridsdorf	-10,949	-1,014	-11,963
22., Donaustadt	-17,200	-4,999	-22,199
23., Liesing	12,928	-915	12,013
Total	**173,850**	**33,812**	**207,662**

Table 2.6: *Balance of Commuters 2012. (= inbound commuters - outbound commuters) Source: Statistik Austria; reporting date: 31.10.2012; territory information: 2014; created on 6.11.2014.*

Employees per Hospital 2015

The numbers of employees per hospital come from the Bisnode Austria register of companies which was provided by the Austrian Institute of Technology for this analysis. Table 2.7 which had also been used as input for the modelling was created by selecting all hospitals from the register. From 34 hospitals in Vienna the register contains only 17 with a total of 25,230 employees. Because of the shift operation in hospitals only 50% of the employees were distributed among the clinics while the other 50% were distributed among the residential buildings.

Hospital ID	***Name***	***Employees***
Hospital 1	Allgemeines Krankenhaus der Stadt Wien - Universitätskliniken	9,321
Hospital 13	Krankenhaus Hietzing mit neurologischem Zentrum Rosenhügel	3,351
Hospital 33	Wilhelminenspital der Stadt Wien	3,000
Hospital 25	Sozialmedizinisches Zentrum Ost	2,856
Hospital 5	Hanusch Krankenhaus der Wiener Gebietskrankenkasse	1,299
Hospital 10	Krankenhaus der Barmherzigen Brüder Wien	801
Hospital 12	Krankenhaus Göttlicher Heiland GmbH	579
Hospital 17	Orthopädisches Spital Speising GmbH	558
Hospital 31	Unfallkrankenhaus Meidling	501
Hospital 3	Evangelisches Krankenhaus Wien gemeinnützige Betriebsgesellschaft m.b.H.	474
Hospital 28	St. Anna Kinderspital GmbH	438
Hospital 30	Unfallkrankenhaus Wien Lorenz Böhler	432
Hospital 11	Krankenhaus der Barmherzigen Schwestern Wien Betriebsgesellschaft m.b.H.	414
Hospital 29	St. Josef Krankenhaus GmbH	381
Hospital 8	Herz Jesu Krankenhaus GmbH	321
Hospital 32	Wiener Privatklinik Betriebs-GmbH & Co KG	261
Hospital 14	Krankenhaus St. Elisabeth GmbH	243
Total		**25,230**

Table 2.7: *Employees per Hospital 2015. Source: Bisnode Austria/AIT, 2015.*

Employees per University & Adult Education Centre 2015

Like the previous dataset the numbers of employees per university and adult education centre were taken from the register of companies. Table 2.8 shows that the biggest employers among the universities which are listed in the register are the University of Vienna, the Medical University and the Technical University of Vienna (the register of companies does not include all universities).

Name	***Type***	***Employees***
Universitaet Wien	University	6,492
Medizinische Universitaet Wien	University	5,343
Technische Universitaet Wien	University	4,515
Universitaet fuer angewandte Kunst	University	1,571
Wirtschaftsuniversitaet Wien	University	1,187
Veterinaermedizinische Universitaet Wien	University	1,086
Universitaet fuer Musik und darstellende Kunst Wien	University	800
Die Wiener Volkshochschulen GmbH	VHS	671
Total		**21,665**

Table 2.8: *Employees per University & Adult Education Centre 2015. Source: Bisnode Austria/AIT, 2015.*

Register of Companies 2015

The source of the register of companies is the Bisnode Austria Holding GmbH. The dataset had been acquired by the Austrian Institute of Technology which provided an already edited set for the analysis. This set contained 3,560 company entries with 1,187,570 employees which is more than the total amount of employees of Vienna in 2013 which is 955,839. The problem with the data is that especially large and/or international corporations with the corporate headquarters in Vienna list all of their domestic and/or international employees in their headquarters with some of the most extreme examples being the Raiffeisen-Landesbanken-Holding GmbH (60,356 empl.), the UniCredit Bank Austria AG (55,443 empl.) and the Österreichische Bundesbahnen-Holding Aktiengesellschaft (41,543 empl.). This was solved by eliminating all companies with more than 1,000 employees with some exceptions like Siemens Aktiengesellschaft Österreich (1,817 empl.) leaving 3,368 companies with 328,790 employees. Of course this solution is not perfect as the elimination will not cover all companies with incorrect employment data for the given address while it will exclude some companies with correct employment data but the approach is sufficiently accurate for the modelling since most of the entries (2,846 or 85%) are constituted of small companies with 150 employees or less.

Employees Wiener Linien 2013

In 2013 the Wiener Linien GmbH & Co KG (public transportation) had 8.577 employees (cf. Wiener Linien 2013: 4). Nevertheless, the share of the employees which work directly in the public transport vehicles is unknown.

Total Number of Employees per District 2013 & Total Number of Unemployed 2013

Similar to the dataset on the balance of commuters the total number of employees per district is needed because the employment data of the other datasets (employees per hospital, university, adult education centre, register of companies and employees of Wiener Linien) does not include the whole working population of Vienna. For that reason, after all employees of the other datasets were distributed their sum was calculated per district and subtracted from the total number of employees per district. The difference could then be distributed per district among all buildings with the use "Industry" and "Commercial" as well as the hospitals with no employees based on the percentage of their weighting factor (e.g. volume)

District	***Total Number of Employees***	***District***	***Total Number of Employees***
1., Innere Stadt	110,104	13., Hietzing	25,791
2., Leopoldstadt	69,739	14., Penzing	28,651
3., Landstraße	95,793	15., Rudolfsheim-Fünfhaus	28,808
4., Wieden	28,237	16., Ottakring	29,492
5., Margareten	19,656	17., Hernals	14,323
6., Mariahilf	28,451	18., Währing	14,407
7., Neubau	35,183	19., Döbling	30,341
8., Josefstadt	16,096	20., Brigittenau	28,389
9., Alsergrund	50,457	21., Floridsdorf	53,911
10., Favoriten	63,907	22., Donaustadt	59,616
11., Simmering	34,667	23., Liesing	54,065
12., Meidling	35,755		
Total 955,839			

Table 2.9: *Total number of Employees per District 2013. Source: Statistik Austria, 2013.*

compared to the total weighting factor of all selected buildings in that district. When looking at Table 2.9 it shows that the first district has the most employees followed by the third district. Now, in regard to their area they are not the largest districts but compared to the land use/ building use (see Figure 2.1, Figure 2.3 & Figure 2.5) it can be seen that they have a high share of buildings with the use "Commercial" or "Industry" which is 33% in the first district (no "Industry") and 30% in the third district (percentage based on building area). The district with the lowest number of employees is the 17th district (14,323 empl.). Here it shows that the share of buildings with the use "Commercial" or "Industry" is only 5% with 48% of the land use (from the land use 2012 dataset, Figure 2.1) being natural environment.
The total number of unemployed in Vienna in 2013 was 120,815 with the source being the Job Centre Vienna; the calculation was done by the magistrate division MA 23 (economics, work and statistics). The value was used to distribute the unemployed among the residential buildings.

Students per School Type 2012/13

The dataset on the students per school type comes from the education statistics of the Austrian Federal Ministry of Education 2012/13 and is provided by Statistik Austria. The original data contained 8 different school types which were summarized into 4 types for the analysis (see Figure 2.14). The data shows that most of Vienna's pupils in 2012/13 went to compulsory school while the least amount went to trade school. The sum of all pupils is 225,645.

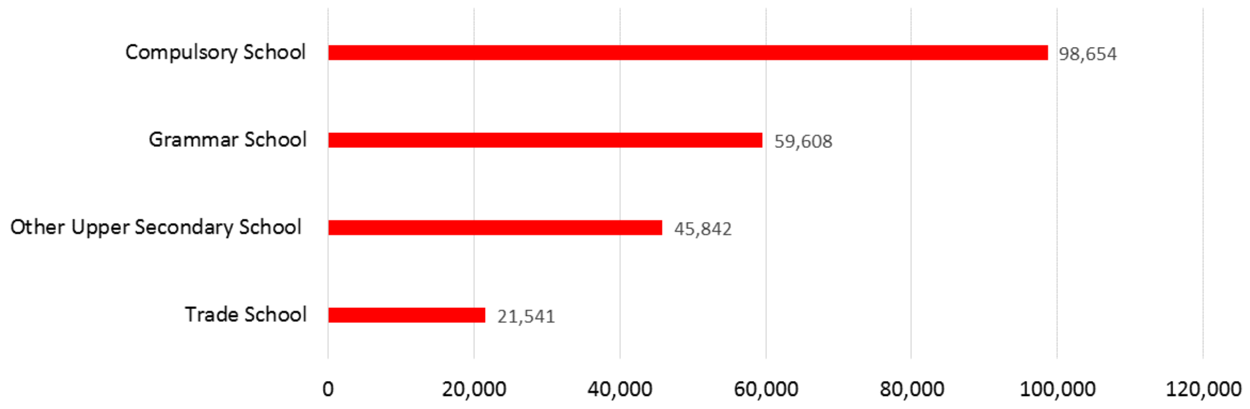

Figure 2.14: *Students per School Type 2012/13. Source: Statistik Austria – education statistics,* Austrian Federal Ministry of Education, 2012/13.

Students per University 2013/14

This dataset contains 4 types of universities – colleges, teacher training colleges, public universities and private universities with the student numbers from the academic year 2013/14. It also includes the student numbers of Vienna´s biggest universities (University of Vienna, Technical University of Vienna and Vienna University of Economics and Business). In preparation for the modelling these numbers were summarized into 5 classes, introducing an additional class for the Medical University of Vienna (see Figure 2.15). The source of the dataset which is from the year 2013/14 is Statistik Austria while the student number of the Medical University of Vienna comes from its annual report 2015. (cf. Medizinische Universität Wien 2015: 77)

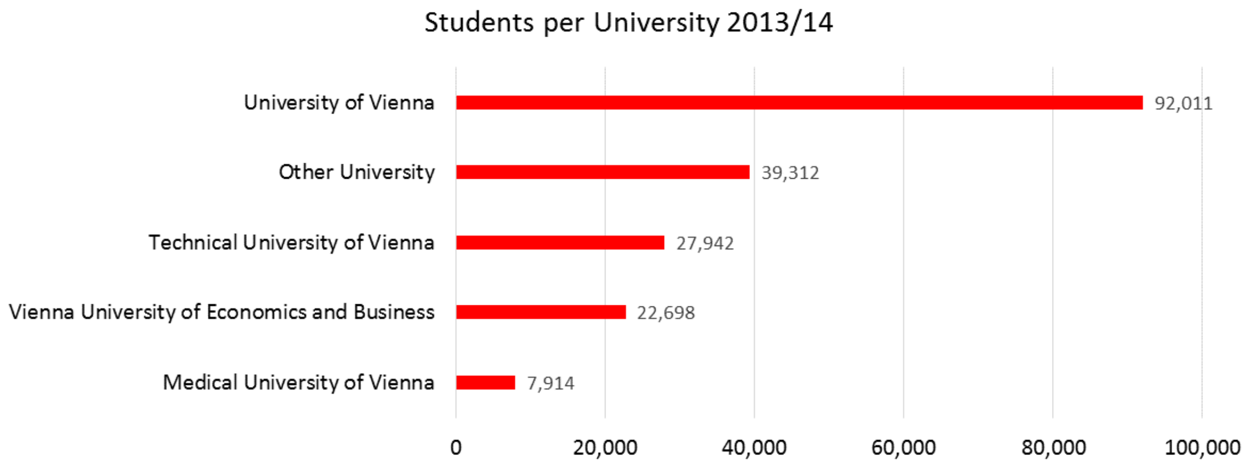

Figure 2.15: *Students per University 2013/14. Source: Statistik Austria*, 2013/14.

Retirees in retirement homes 2013

Age Group	***Population***	***Share of Elderly Dependents***	***Share of Elderly Dependents in Retirement Homes***	***Retirees in Retirement Homes***
65-69	88,699	0.030	0.291	774
70-74	86,114	0.050	0.291	1,253
75-79	42,520	0.098	0.291	1,213
80-84	38,111	0.210	0.291	2,329
85-89	27,990	0.382	0.291	3,111
90+	13,881	0.644	0.291	2,601
Total	**297,315**			***11,282***

Table 2.10: *Retirees in Retirement Homes 2013. Source: Federal Statistical Office (Germany).*

The number of retirees in retirement homes for 6 different age groups was calculated by multiplying the population of the particular age group with the share of elderly dependents of that age group and the share of elderly dependents in retirement homes (since most of them are nursed at home). Origin of the data on the share of elderly dependents is the Federal Statistical Office of Germany (Statistisches Bundesamt) with the values referring to Germany. The same data was not available for Austria so the German values were used as input for the analysis. Since the two countries are similar in many ways the results should be sufficient. The similarity can be seen by comparing the share of elderly dependents who are nursed at home where the value for Germany is 70.9% (cf. STATISTISCHES BUNDESAMT 2013: 9) and the value for Upper Austria, which is available, is 70%. (cf. AMT DER OÖ. LANDESREGIERUNG 2014: 5) From this number the share of elderly dependents in retirement homes was calculated as 0.291. The values for the modelling were directly calculated in the python script. While the retirees in retirement homes were distributed to the retirement homes over all Vienna the remainder of the retirees were distributed to the residential buildings per registration district.

Number of patients

Name	*Number of Beds*	*Average Occupancy Rate*	*Number of Patients*
AKH	1,824	0.85	1,550
Krankenhaus Hietzing	1,005	0.85	854
Wilhelminenspital	988	0.85	840
SMZ Baumgartner Hoehe	952	0.85	809
SMZ Ost Donauspital	946	0.85	804
Krankenanstalt Rudolfstiftung	707	0.85	601
SMZ Sued Kaiser Franz Josef Spital	668	0.85	568
Hanusch Krankenhaus	411	0.85	349
Krankenhaus der Barmherzigen Brueder	395	0.85	336
Krankenhaus Goettlicher Heiland	284	0.85	241
Orthopaedisches Spital Speising	252	0.85	214
Evangelisches Krankenhaus Wien	232	0.85	197
Krankenhaus der Barmherzigen Schwestern	210	0.85	179
SMZ Floridsdorf	166	0.85	141
Hartmannspital	161	0.85	137
Privatklinik Doebling	160	0.85	136
Herz Jesu Krankenhaus	155	0.85	132
Rudolfinerhaus Privatklinik	155	0.85	132
St Josef Krankenhaus	155	0.85	132
Neurologisches Rehabilitationszentrum Rosenhuegel	147	0.85	125
Wiener Privatklinik	145	0.85	123
Unfallkrankenhaus Meidling	142	0.85	121
Sanatorium Liebhartstal	137	0.85	116
Krankenhaus St Elisabeth	129	0.85	110
Unfallkrankenhaus Lorenz Boehler	128	0.85	109
St Anna Kinderspital	119	0.85	101
Heeresspital Wien	113	0.85	96
Goldenes Kreuz Privatklinik	102	0.85	87
Sanatorium Hera	100	0.85	85
Confraternitaet Privatklinik Josefstadt	96	0.85	82
Orthopaedisches Krankenhaus Gersthof	92	0.85	78
SMZ Sophienspital	89	0.85	76
Rehabilitationszentrum Meidling	52	0.85	44
Total	**11,417**	**0.85**	**9,704**

Table 2.11: *Number of Patients per Hospital 2013. Source: Ministry of Health, 2013.*

Unfortunately, there exist no patient numbers of the hospitals of Vienna but similar to the previous described dataset the values could be estimated by multiplying the number of beds per hospital with the average occupancy rate. The former is provided by the City of Vienna with the source being the Ministry of Health while the latter was taken from the Regionaler Strukturplan Gesundheit (RSG) Wien 2009 (regional structure plan) which is also provided by the City of Vienna. Outpatients were not considered.

3 Method

The proposed method of this thesis is based on areal weighting as introduced in chapter 1. It is assumed that the population a) is distributed evenly inside the districts, b) is living in residential buildings and c) the number of residents per building is dependent on the size of the housing space. On the basis of these assumptions the distribution of the population can be done according to the formula

$$P_h = \frac{Hs_h}{Hs_t} \times P_t \tag{3.1}$$

with P_h being the population per house, Hs_h the living space per house, Hs_t the total living space per administrative unit (e.g. district) and P_t the total population per administrative unit. The living space per house is defined as the mathematical product of the building footprint A_h and its height h_h (alternatively its number of floors) minus the vacancy V, if known.

$$Hs_h = (A_h \times h_h) \times (1 - V) \tag{3.2}$$

Finally, the total living space per administrative unit is the sum of the living space of all buildings:

$$Hs_t = \sum_{h=l}^{l} Hs_h \tag{3.3}$$

Figure 3.1 visualizes the concept of the modelling approach where the population of a source zone (e.g. district, registration district or the whole of Vienna) is disaggregated to the target zone (e.g. buildings in that district) weighted by their area (or volume) so that larger buildings will get a bigger proportion of the population. Afterwards, the population of the buildings is aggregated to the 100x100m grid where each cell gets a share of the population of its intersecting buildings depending on how much of the building footprint lies within the cell.

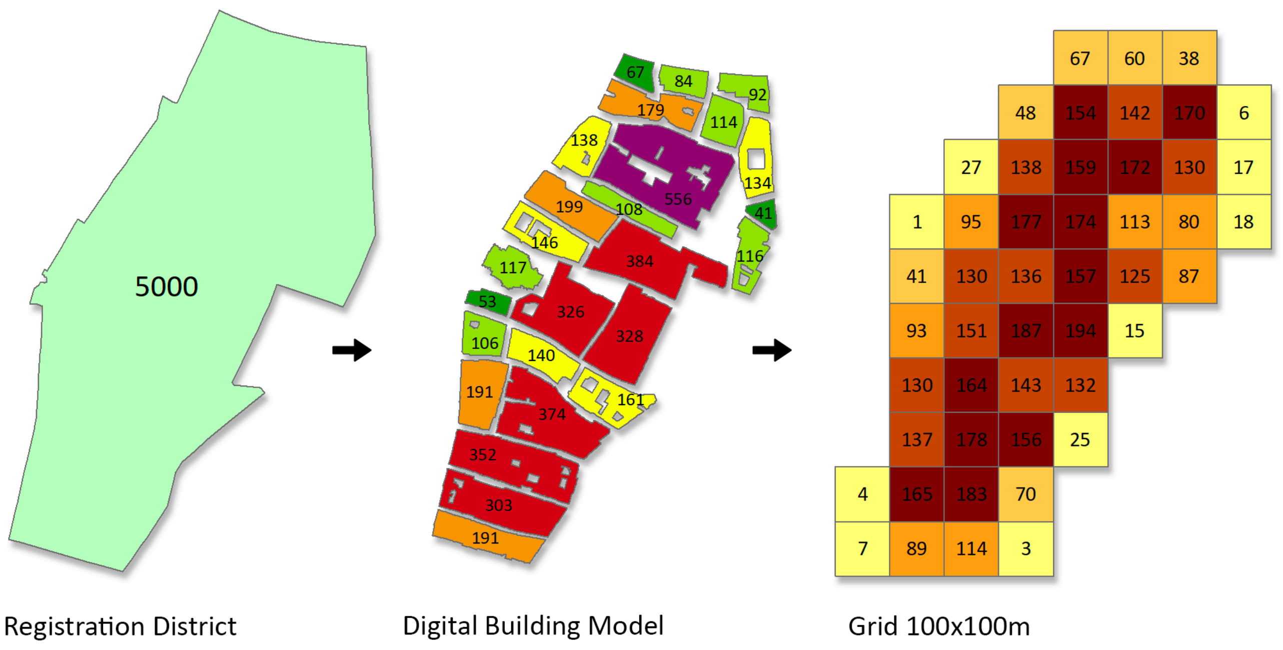

Figure 3.1: *Principle of Population Disaggregation and Aggregation. The population e.g. of a registration district (**left**) is distributed to the buildings that lie in that district based on a weighting factor (**middle**) and is then aggregated to a 100x100m raster (**right**). Source: own illustration.*

In total 5 models were developed for this thesis numbered from 01 to 05.

- Model 01 delivers presumably the most accurate results because it utilizes all available datasets and distributes the population to the buildings weighted by their volume.
- The second model also utilizes all datasets but distributes the population weighted only by the area of the building footprints. This is done to see how the distributed population changes when no height data of the buildings is available.
- The third model excludes the register of companies and uses the volume as weighting factor while the
- fourth model does not use auxiliary data at all (except the balance of commuters so that the total distributed population is for all models the same) and distributes the population per registration district to the digital building model weighted by volume.
- Finally, the fifth model does not utilize auxiliary data as well but distributes the population to the surfaces of the Land Use 2012 layer, weighted by area. After the disaggregation is done, the population is then aggregated to a raster with 100x100m (=1 hectare), making it easier to compare the results. Also, the results can be compared to Statistik Austria, which disaggregated the daytime population of Vienna 2013 on raster level.

3.1 Modelling Daytime Population

In this section model 01 will be described in more detail and the principle of the modelling will be shown in Figure 3.2 - Figure 3.11. On the left side of the figures the source and type of the population numbers can be seen (which is either from the shapefile of the registration districts or from a table). Furthermore, it is explained if the values exist per spatial unit (registration district, school type etc.) or are a single value (e.g. total number of unemployed). If the source of population is a sum sign it means that the value(s) is/are an output of the previous calculations of the script. The sum sign (without text underneath), to which the first arrow is leading means that the value(s) from the input dataset is/are taken directly for the distribution. If there is a text underneath then the population from the source was manipulated and the population mentioned in the text is distributed. How the manipulation was done can be seen by the dataset/sum sign which is located over the first sum sign (e.g. the already distributed employees are subtracted from the total working population in order to distribute the remaining working population). Next, an arrow will lead to the distribution. This can be a single arrow or two branching ones with a percentage meaning that the respective share of the population will be distributed to its target. Now comes the symbol for the distribution of the population which can be for the whole of Vienna, per district, per registration district or to the public transportation buffer. If there are yellow dots inside the symbol then the information on the target comes from the geospatial data (point-shapefiles, e.g. address points), otherwise it comes from the digital building model. There are also three special cases (hospital, university & register of companies) where the symbol for the distribution (whole of Vienna) is accompanied by a building and a text below (e.g. “per hospital”). This means that the employees or patients are distributed to the buildings per hospital (which mostly have more than one building) over the whole of Vienna (since the hospitals are located all over the city). The same is valid for the universities. In the third case the employees of a company are distributed to the respective building. Finally, an arrow is leading to the target which are always buildings except for the public transportation buffer. The numbers next to it represent the age group of the population which was distributed. When population is allocated to buildings then it is always weighted by a factor (volume or area depending on the model) except for the working population of the register of companies which is directly transferred to the respective building where the join address matches. Another exception is when the patients are subtracted from the distributed population where the size of the subtrahend is weighted by the population number of the building. The order of the figures is based on the age group of the distributed population, not on the order of the script.

Distribution of Infants

The infant population (age 0-4) comes from the polygon-shapefile with the registration districts and is distributed in equal parts to the kindergartens (including daycare centres) and residential buildings (see Figure 3.2). This is because the population includes very young children who are with their parents and not in daycare centres and since it is not known where they are during the day they are allocated to the residential buildings. There are no data available on the percentage of young children in daycare centres so it was estimated for the model to be 50%. The distribution to the kindergartens is done per district because it is assumed that the children will go to a facility which is not too far away but at the same time not necessarily in the same registration district. On the other hand the infants who are allocated to the residential buildings are distributed per registration district because the source of the population has the same resolution and there are no commuter numbers involved since infants do not commute (the commuter numbers only cover the student and working population).

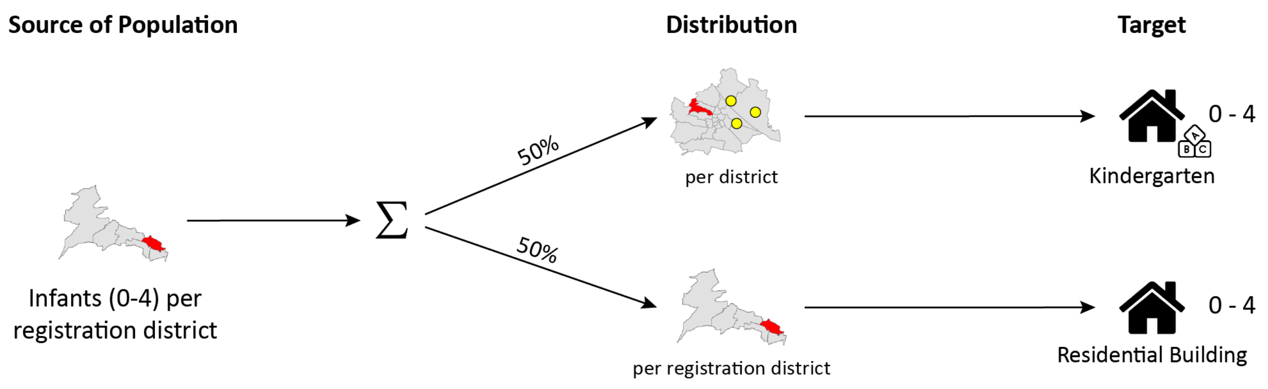

Figure 3.2: *Distribution of Infants. Source: own illustration.*

Distribution of Pupils

Pupils are distributed per school type covering all of the school buildings in Vienna because there are no student numbers per school type and district (see Figure 3.3). The pupils of the trade school *(Berufsschule)* are a special case because they spend most of their time, namely 80% working and only 20% in school, according to WKO 2006:6 so they are distributed to the respective building type accordingly. The population from the data source has no age group assigned to it but since the pupils age group of each school type is known the distributed population of the particular school type gets that age group. However, this is more important for labelling purposes and does not influence the calculations in the script.

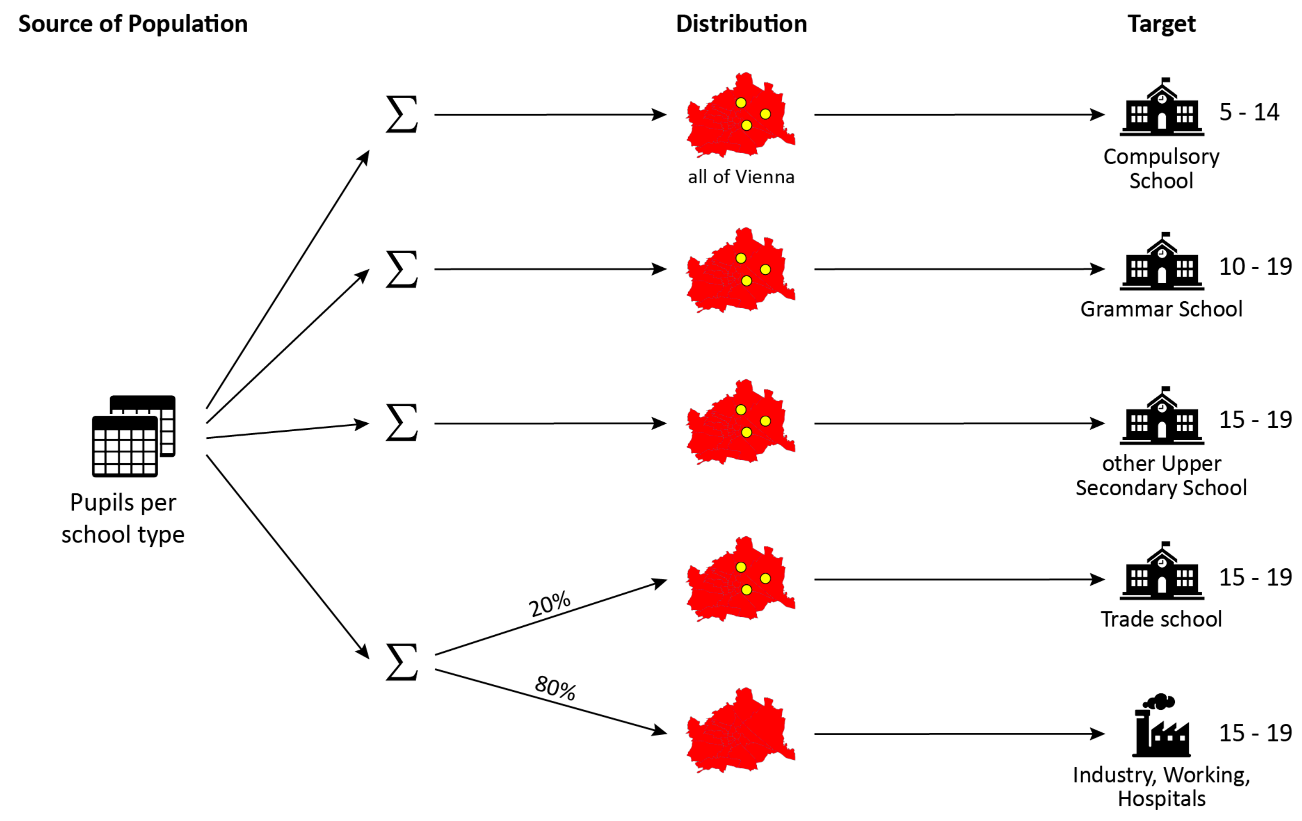

Figure 3.3: *Distribution of Pupils. Source: own illustration.*

Distribution of Students and Employees (per University, Adult Education Centre)

Figure 3.4 and Figure 3.5 show the distribution of the students per university and employees per university and adult education centre. The student population per university is split similar to the infants so that 50% of the students are distributed among the residential buildings over Vienna and 50% are distributed per university over Vienna. Per university means that the students are distributed to the faculties of each university. There is no data available on the percentage of time a student in Austria spends at university so it was estimated to be 50% since the data also covers advanced students. Similar to the students the employees per university were also distributed per university. However, the population comes from the register of companies where the employment data is in contrast to the student data not available for all universities. In the same dataset (employees per university) are also the employees of the adult education centres. However, not per centre but for all centres meaning that the population is distributed to all facilities over Vienna.

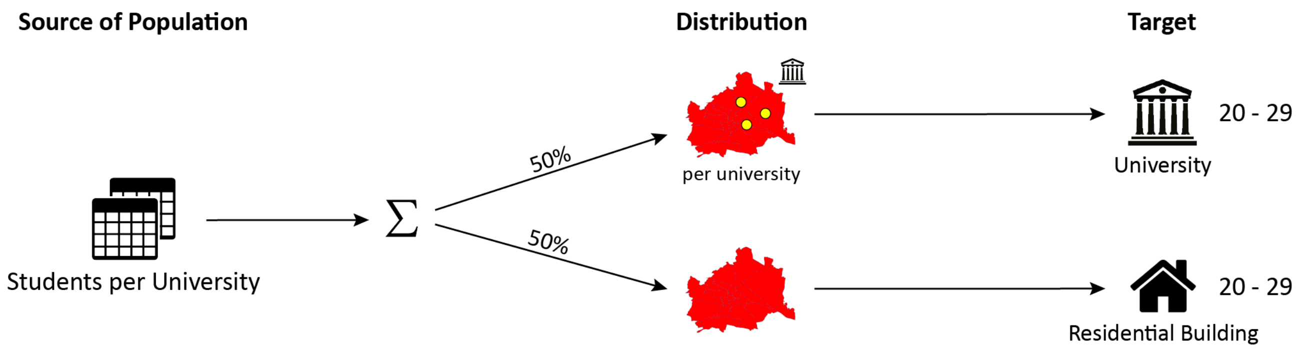

Figure 3.4: *Distribution of Students per University. Source: own illustration.*

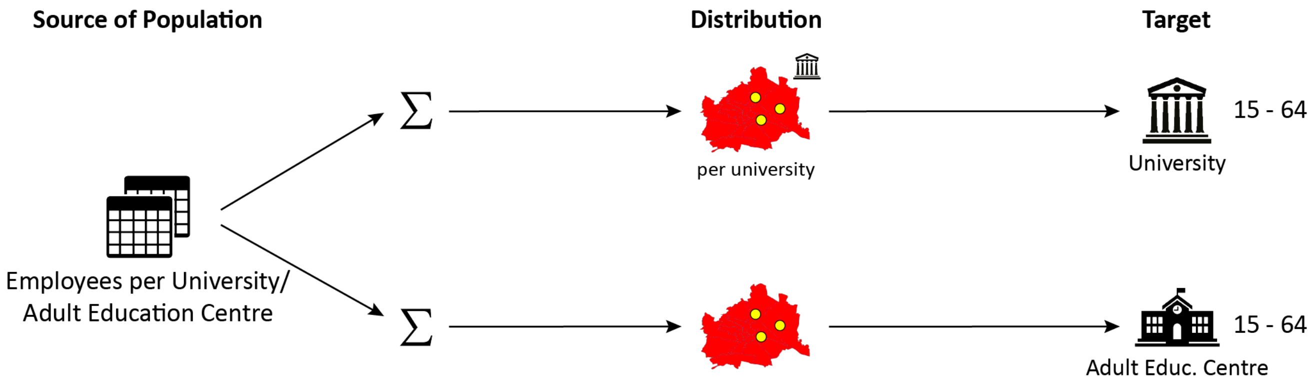

Figure 3.5: *Distribution of Employees per University/vhs. Source: own illustration.*

Distribution of Employees (Hospitals, Wiener Linien & Companies)

The employees of the hospitals are like the university students distributed to the hospital buildings per hospital (see Figure 3.6 and Figure 3.7). Because of the shift operation in hospitals 50% of the population is distributed to residential buildings (because the location of the employees in their free time is not known). Furthermore, 50% of the employees of Wiener Linien GmbH & Co KG are distributed to the transportation network buffer (without weighting) while the other 50% are also distributed to the residential buildings also because of the shift work. Meanwhile, the employees from the register of companies are allocated to the respective building where the addresses of the two datasets match. The address data of the buildings comes from a spatial join with the address point-shapefile. Theoretically, the population is distributed to the buildings with the use class "Industry" and "Commercial" but this is not always true for the reason that residential buildings can also house businesses. Moreover, address points can also lie within buildings of any other use class such as "Social Infrastructure" or "Transportation".

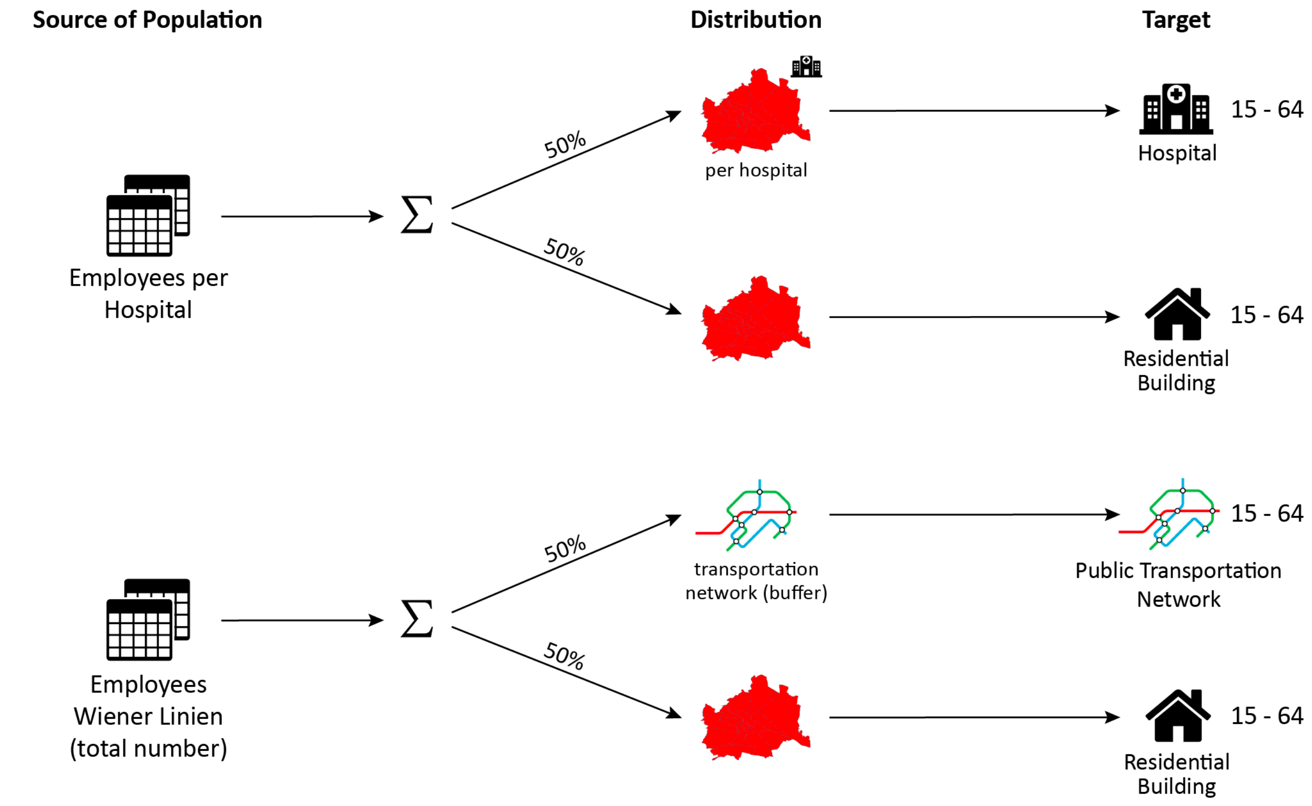

Figure 3.6: *Distribution of Employees - Hospitals & Wiener Linien. Source: own illustration.*

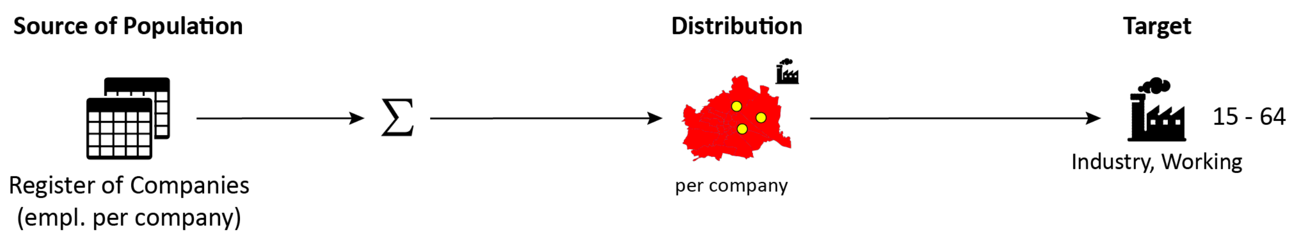

Figure 3.7: *Distribution of Employees - Register of Companies. Source: own illustration.*

Distribution of Remaining Employees and Unemployed

Since not all employees have been distributed yet the remaining working population is calculated by subtracting the already distributed employees from the total employees per district (see Figure 3.8). The difference is then distributed to the buildings with the use "Industry", "Commercial" or "Hospital" if the respective building does not have employees already allocated to it. The unemployed are distributed to the residential buildings over Vienna.

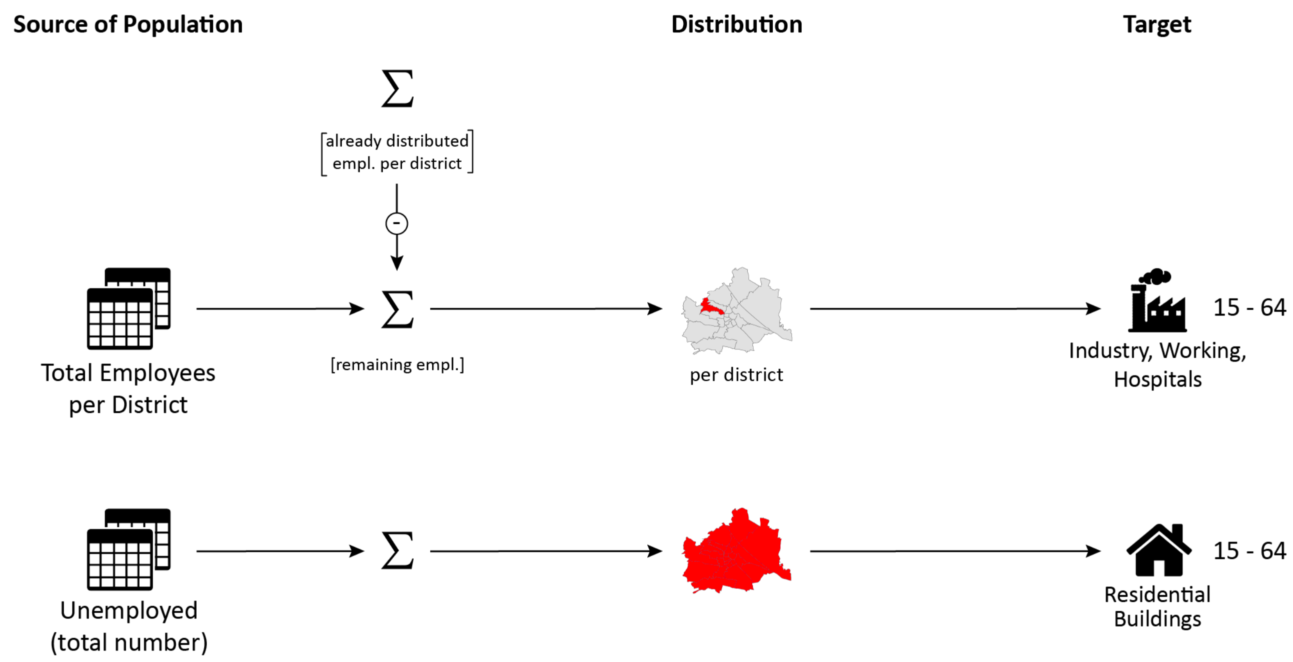
Source of Population
Distribution
Target
Σ
[already distributed empl. per district]
-
Total Employees per District
Σ
[remaining empl.]
per district
15 - 64
Industry, Working, Hospitals
Unemployed (total number)
Σ
15 - 64
Residential Buildings

Figure 3.8: *Distribution of Remaining Employees and Unemployed. Source: own illustration.*

Distribution of Retirees

For the distribution of the retirees the population is split into elderly dependents who are nursed in retirement homes/care homes and persons who are not in need of care and include strictly speaking also the dependents who are nursed at home (see Figure 3.9). The distribution is then done accordingly to the retirement homes for all of Vienna (since the elderly dependents can theoretically live in any retirement home in Vienna) and to the residential buildings per registration district (as the source of the population has the same resolution). The splitting is based on the values for the share of elderly dependents and the share of elderly dependents in retirement homes which exist for six different age groups within the retirees (see Table 2.10 on page 26).

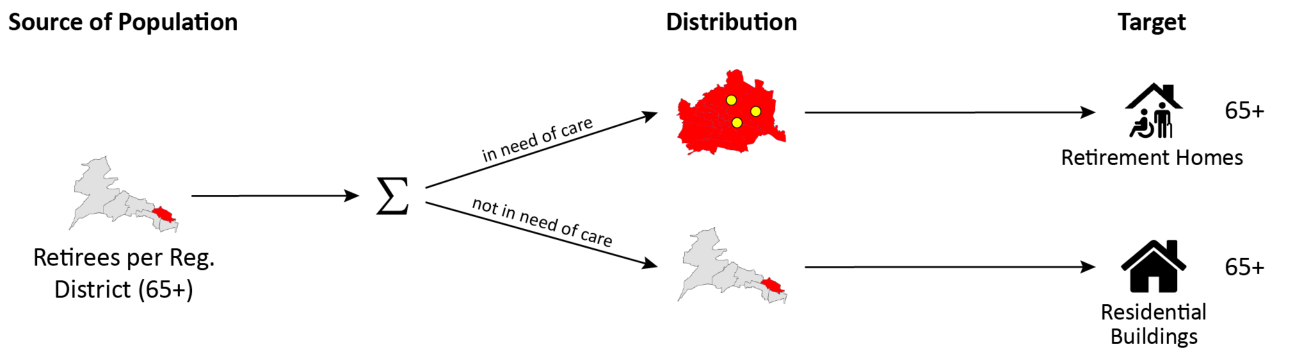

Figure 3.9: *Distribution of Retirees. Source: own illustration.*

Distribution of Remaining Population

As with the employees not the whole daytime population of Vienna is distributed by using the datasets of Figure 3.2 - Figure 3.9 because they do not include every person (e.g. not all unemployed might be reported as unemployed). For that reason the total daytime population is calculated by adding up the total population by registration district and the total balance of commuters and subtracting that value from the already distributed population. This leads to the remaining population which is distributed to the residential buildings for the whole of Vienna (see Figure 3.10).

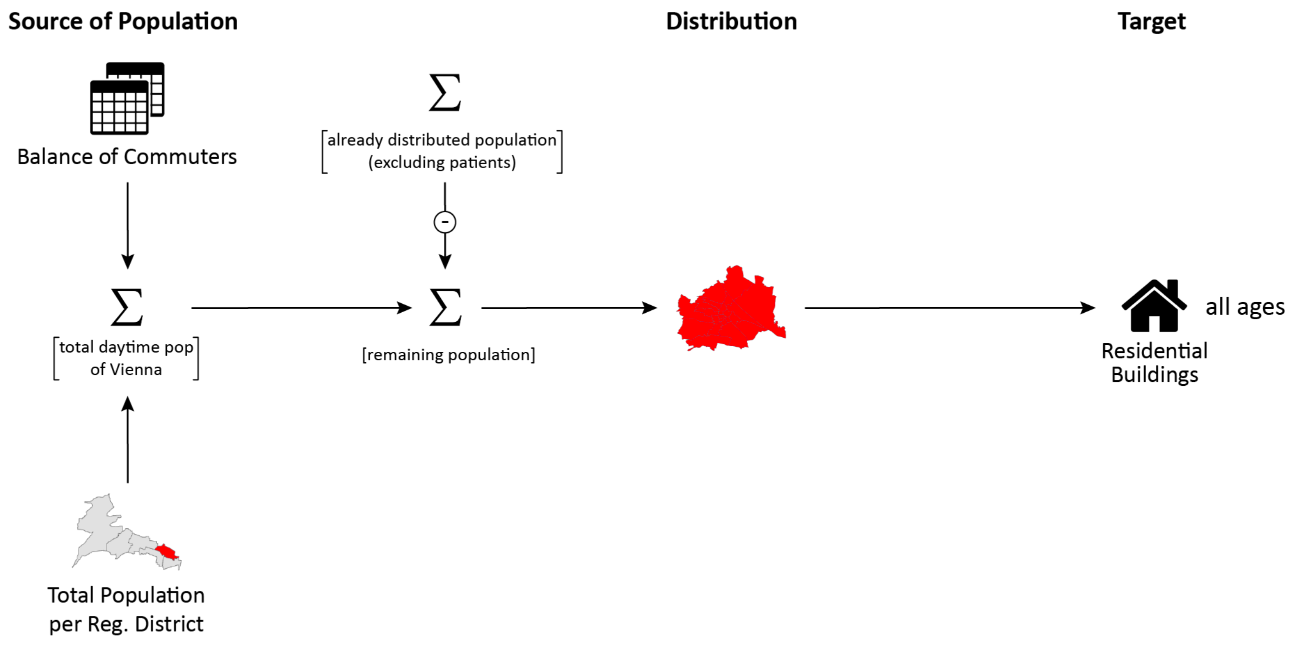

Figure 3.10: *Distribution of Remaining PopulationSource: own illustration.*

Distribution of Patients and Subtraction from Buildings

Another consideration in the model concerns the patients (see Figure 3.11) who are distributed per hospital over the whole city. The number of patients each hospital gets is derived from the number of beds multiplied by the average occupancy rate. However, this population is basically constituted of the other population groups who have been distributed so far thus making it necessary to subtract the patients from those groups. As mentioned before the size of the subtrahend is weighted by the population of the respective building meaning that buildings with a large population count loose more persons than buildings with a small count as those buildings have a potentially higher amount of sick people. Also, buildings with no population assigned to it will not get negative values.

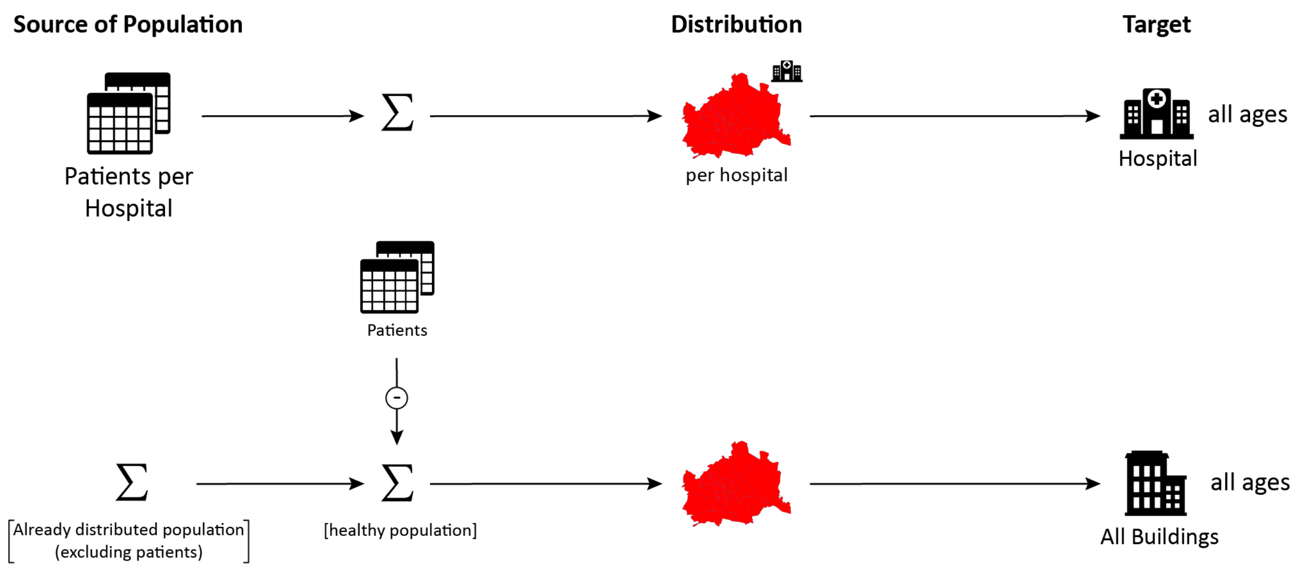

Figure 3.11: *Distribution of Patients and Subtraction from Buildings. Source: own illustration.*

Nighttime Population

The nighttime population of Vienna for the year 2013 was also calculated but since this thesis focuses on the daytime population the description will be short. Calculating the nighttime population is much more simple than the daytime population because most of the residents are at home during the night. Also, commuter numbers are not accounted for. So basically, the whole population from the registration districts shapefile can be distributed to the residential buildings per registration district weighted by their volume. Exceptions are the employees of the hospitals who work during the night and the residents of retirement homes. Those two groups are of course distributed to the respective facility first. Then the remaining population can be allocated by distributing the population per age group as well as the retirees who are not in retirement homes to the residential buildings per registration district. Before that the employees of the hospitals who belong to the age group 15-64 are subtracted from the same age group of the population per registration district weighted by the population number of that registration district compared to the total population of Vienna. This means that registration districts with more inhabitants have potentially more hospital employees who are working and therefore a larger amount of population is subtracted from those districts. This approach is necessary because the residence of the employees is not known. The nighttime population has also been aggregated to the 100x100m grid.

3.2 Implementation

Calculating the daytime population of a city like Vienna using 23 different input datasets is a complex task which involves various steps and data manipulation. In order to do this efficiently a Python script using the ArcPy site package of the mapping and spatial analytics software ArcGIS was written. ArcPy can be used for geographic data analysis, data conversion, data management and map automation with Python, a general-purpose programming language. (cf. Esri n.d.) The script can be found in full length in the appendix and shows in detail how the daytime population was modelled.

What follows is an overview of the five models as well as the structure of the script with a short description of its most important calculations (order according to the code lines). The script is divided into three parts: in the first part the model is selected and the variables are defined, in the second part models 04 or 05 are calculated and in the third part models 01, 02 or 03 are calculated. Table 3.1 on page 40 shows which input datasets were used for each model.

model 01 - utilize all available data, weighting by building volume
model 02 - utilize all available data, weighting by building footprint
model 03 - utilize all available data except register of companies, weighting by building volume
model 04 - do not use auxiliary data (except balance of commuters), weighting by building volume
model 05 - do not use auxiliary data (except balance of commuters), distribute to Land Use 2012 layer, weighting by area

ZBEZ...registration district	univ...university	bkm...digital building model
BEZ.....district	vhs....adult education centre	

PART A - define variables

- set model number
- define workspace
- define weighting factor
- define output names
- define input datasets (soft-coded)
- define input field names of datasets

As mentioned before in the first part the model number is selected because the script calculates only one model at a time. The weighting factor is already defined depending on the entered model number. Also important in this part is the definition of the output names, input datasets and the input field names. The values of the input datasets (e.g. number of students) are soft-coded, meaning they are obtained from an external source (in this case database tables) and are not coded in the source code.

PART B - model 04, 05

- dissolve ZBEZ to get aggregated population per BEZ
- add balance of commuters to population per BEZ
- distribute age0_4 to residential buildings/areas per ZBEZ (because there are no data on kindergartens and they are often located in residential buildings)
- distribute age5_9 to bkm/realnut (social infrastr.) per BEZ
- distribute age10_64 to bkm/realnut (commercial, industry, social infrastructure) per BEZ
- distribute age65_plus to bkm/realnut (residential) per ZBEZ
- aggregate population to raster

If the model number is set to 04 or 05, the code in part B will be executed and the respective model is calculated (meaning that the whole part is inside an if-statement). This part comes before part C because its length is shorter (due to the fact that models 04 and 05 do not use auxiliary data and are therefore much more simple to program) which means that less code has to be inside the if-statement. At the end of part B there is an exit command, preventing the program to run the code of part C if model 04 or 05 were calculated. The difference between model 04 and 05 is that the population in model 04 is distributed to the buildings of the digital building model (weighted by volume) while in model 05 it is distributed to the Land Use 2012 layer (weighted by area as there are no buildings).

PART C - model 01, 02, 03

- dissolve ZBEZ to get aggregated population (per BEZ)
- add pendlersaldo to population (per BEZ)

EMPLOYEES HOSPITAL, UNIVERSITY, ADULT EDUC. CENTRE
- distribute working pop of hospitals (age15_64) - 50% to hospitals, 50% to residential build. (whole Vienna)
- spatial join univ, vhs to bkm
- distribute working pop of univ/vhs (age15_64) to universities (per univ, whole Vienna)/vhs (whole Vienna)
- get distributed population (univ/vhs) per BEZ
- get distributed population (hospital) per BEZ

REGISTER OF COMPANIES (only model 01, 02)
- dissolve adr_points, single_part after join_adr (eliminate identical addresses)
- join population of register of companies to adr_points
- spatial join adr_points to bkm (one_to_many)
- dissolve bkm and summarize population of register of companies
- get distributed population (register of companies) per BEZ

EMPLOYEES WIENER LINIEN
- buffer public transportation network
- clip buffer (eliminate areas outside of Vienna)
- distribute 50% of pop_wrlinien to buffer
- distribute 50% of pop_wrlinien to residential buildings (whole Vienna)
- get distributed population (wrlinien) per BEZ

REMAINING WORKING POPULATION
- get total distributed working population per BEZ
- calculate remaining working population per BEZ ([total working pop]-[distributed working pop])
- distribute remaining working population per BEZ where:
  USE_bkm in (trade, industry) and population of register of companies IS NULL or USE_bkm = social infrastructure and ID_hospital IS NOT NULL and working population of hospital/univ/vhs = 0

UNEMPLOYED
- distribute unemployed to bkm (residential buildings, whole Vienna)

INFANTS
- spatial join kindergarten to bkm
- distribute 50% of infants to kindergarten per BEZ
- distribute 50% of infants to residential buildings per ZBEZ

PUPILS PER SCHOOL
- spatial join schools to bkm
- distribute pupils (compulsory school, grammar school, other upper sec. school) to schools (whole Vienna)
- distribute 20% of pupils (trade school) to schools
- distribute 80% of pupils (trade school) to commercial/industry/hospital buildings

----

STUDENTS PER UNIVERSITY
- distribute 50% of students to university buildings (per university, whole Vienna)
- distribute 50% of students to residential buildings (whole Vienna)

RETIREES
- calculate retirees in need of care
- spatial join retirement homes to bkm
- distribute retirees in need of care to retirement homes (whole Vienna)
- distribute retirees in need of care to residential buildings per ZBEZ

PATIENTS
- distribute patients to hospitals (whole Vienna)

REMAINING POPULATION
- calculate distributed population (patients excluded) & total population of Vienna
- calculate remaining population ([distributed pop]-[total pop])
- distribute remaining population to residential buildings (whole Vienna)
- subtract patients from buildings weighted by their number of residents (whole Vienna)

- aggregate population to raster

Part C will be executed if model number 01, 02 or 03 is selected. Model 01 is the most accurate by including all available datasets and distributing the population based on the building volumes. If model 02 is selected the only change is that the weighting factor is defined as the building footprint instead of the volume. In case of model 03 where the register of companies is excluded more of the remaining employees (=[total working population]-[distributed working population from hospital, univ, vhs, wrlinien]) are distributed among the buildings based on their volume instead of their address point.

Dataset	***Model***	***Year***	***Type***
Data on land use			
Land Use Vienna *(Realnutzungslayer)*	01, 02, 03, 04, 05	2012	polygon-shp
Data on buildings			
Digital Building Model *(Digitales Baukörpermodell)*	01, 02, 03, 04	2015	polygon-shp
Geospatial data			
Building address data	01, 02	2015	point-shp
Location of hospitals	01, 02, 03	2017	point-shp
Location of schools	01, 02, 03	2017	point-shp
Location of universities	01, 02, 03	2017	point-shp
Location of adult education centres	01, 02, 03	2017	point-shp
Location of kindergartens	01, 02, 03	2017	point-shp
Location of retirement homes	01, 02, 03	2017	point-shp
Public transportation network	01, 02, 03	2014	line-shp
Districts & Registration Districts of Vienna	01, 02, 03, 04, 05	2014	polygon-shp
Socioeconomic data			
Population of Vienna per registration district in 5 year age groups	01, 02, 03, 04, 05	2013	table
Balance of commuters	01, 02, 03, 04, 05	2012	table
Employees per hospital (from register of companies)	01, 02, 03	2015	table
Employees per university & adult education centre (from register of companies)	01, 02, 03	2015	table
Register of companies (number of employees)	01, 02	2015	table
Employees of Wiener Linien	01, 02, 03	2013	single value
Total number of employees per district	01, 02, 03	2013	table
Total number of unemployed	01, 02, 03	2013	single value
Students per school type	01, 02, 03	2012/13	table
Students per university	01, 02, 03	2013/14	table
Retirees in retirement homes (derived from percentage of people in need of care per age group and the amount of them which are in retirement homes)	01, 02, 03	2013	table
Number of patients (derived from number of beds per hospital and average occupancy rate)	01, 02, 03	2013	table

Table 3.1: *Input datasets of model 01-05.*

4 Results

Result of the population disaggregation described in the previous chapter is the daytime (and nighttime) population of Vienna in 2013. Depending on the model the outcome of the daytime population differs by the level of detail with model 01 being the most detailed.

4.1 Comparing Daytime and Nighttime Population

This chapter describes the results of the daytime (model 01) and nighttime disaggregation. When looking at Figure 4.1 on page 44 which depicts both populations as well as the building use per structure it can be seen that during the day buildings with the use “Commercial“ or “Industry“ (also “Social Infrastructure“) as well as large buildings have more population allocated to them than residential buildings. This is expected since those are the types of buildings where people are working or attending school. During the night this ratio is reversed as people are mostly at home. Also, there are 207,662 less persons in the city because of the commuters.

Building Use	***Average Population per Building (Daytime)***	***Average Population per Building (Nighttime)***	***Building Count***
Social Infrastructure	182.31	13.27	1788
Commercial	163.36	0.33	3380
Industry	81.42	0.00	2988
Residential	10.23	22.01	72890
Recr. and Leis. Facilities	2.10	0.00	686
Techn. Infrastr./Constr. Site	1.76	0.29	746
Transportation	1.09	0.00	908
Natural Environment	0.48	0.00	162
Agriculture	0.39	0.00	297
Public Space	0.00	0.00	77
Water Bodies	0.00	0.00	8
No data	0.00	0.00	3
Total			***83,933***

Table 4.1: *Average Population per Building (daytime & nighttime). Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

Table 4.1 shows the average population per building during the day and night. Excluded from the statistic are very small buildings with a height less than 5m and an area less than 30m². These make up 26% of the buildings of the digital building model but they have only a mean daytime population of 0.06 (0.13 nighttime) and in total 3,730 (0.5% of total daytime population) inhabitants during the day and 8,389 (0.5% of total nighttime population) inhabitants during the night. The average population per building during the day is as expected high for commercial and social infrastructure buildings with the maximum being 182 persons per building (social infrastructure). During the night most of the population is concentrated in residential buildings with an average population of 22 per building. This is more than twice the amount of the daytime population which has a value of 10. At first this seems still like a large number for the daytime but it should be considered that residential buildings get most of the infant population (most kindergartens are located

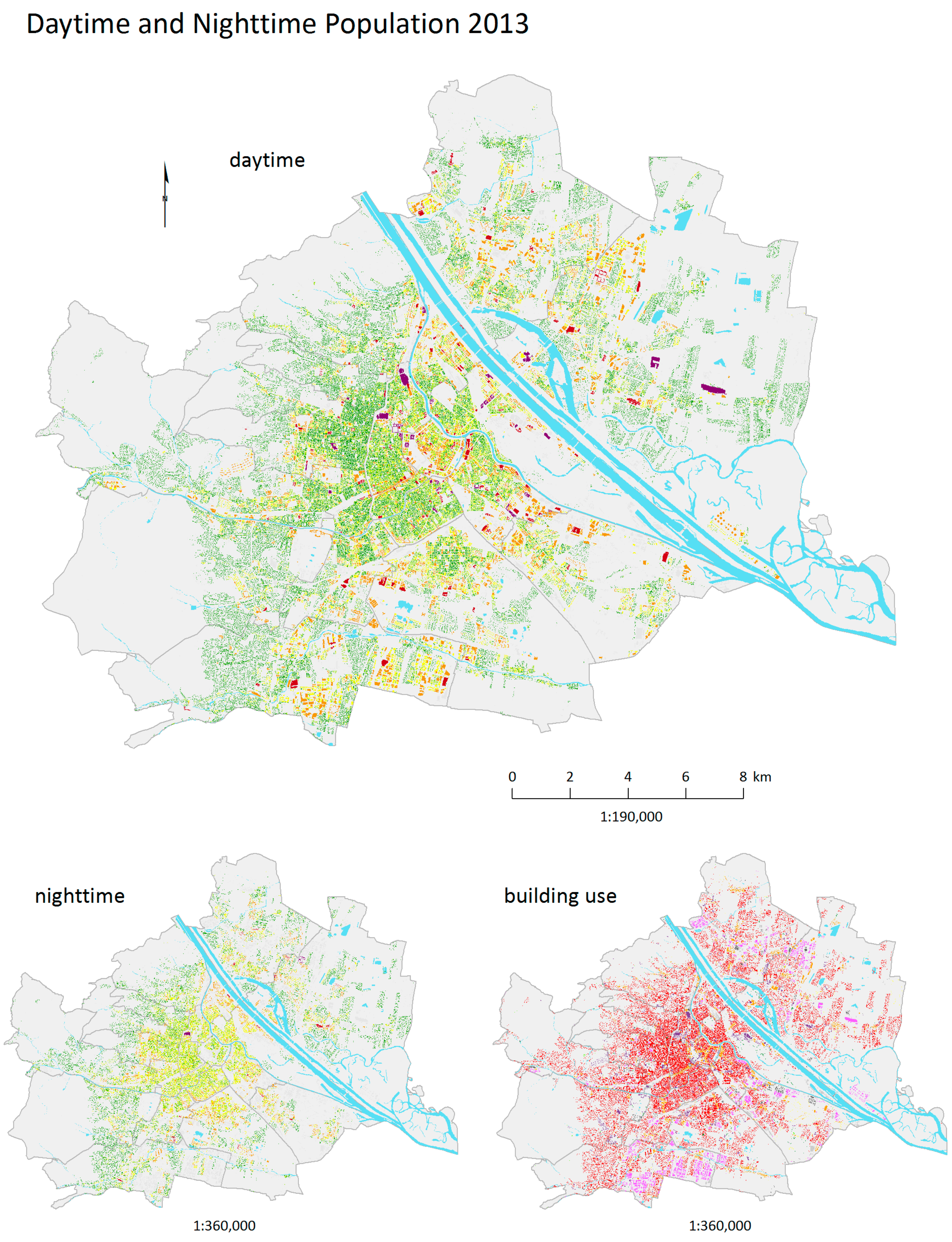

Figure 4.1: *Daytime and Nighttime Population 2013 - Digital Building Model. Legend of daytime and nighttime population see Figure 4.2; legend of building use see Figure 2.5 on page 13. Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

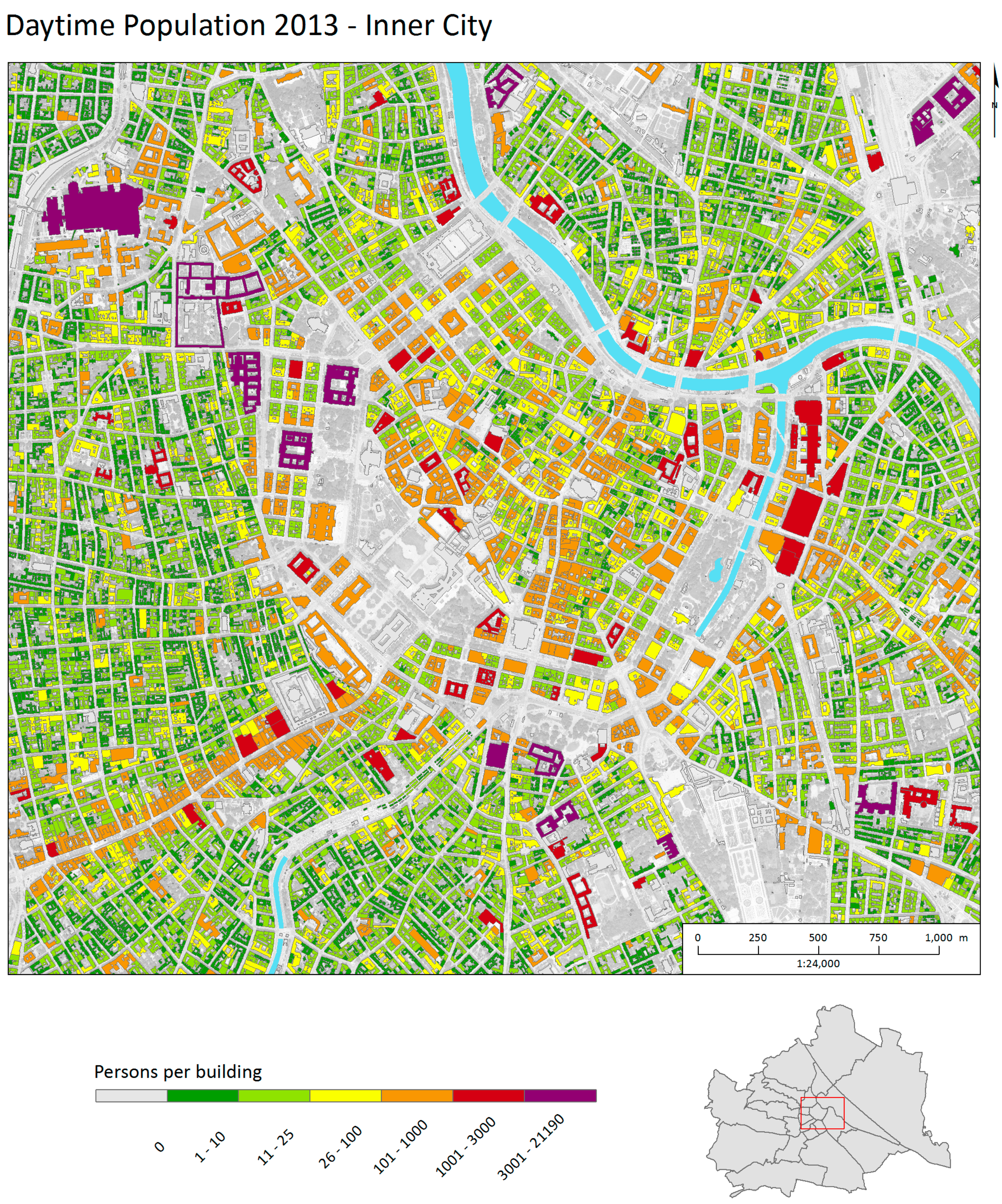

Figure 4.2: *Daytime Population 2013 - Digital Building Model. Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

in residential buildings), 50% of the hospital and Wiener Linien employees, 50% of the university students, 100% of the unemployed, most of the retirees as well as 100% of the remaining population. A more detailed view of the daytime population show Figure 4.2 (2D) and Figure 4.3 (3D) which depict the inner city of Vienna. Here it can be seen that many of the commercial buildings have a population of between 101 and 1,000 allocated to them. Museums, opera houses and churches usually don´t get population assigned because they are often large buildings with only a few persons inside compared to their size. For that reason they are excluded when distributing population weighted by building volume or area. However, it is possible that they get population from the company register if the join address matches. This is for example the case with the Natural History Museum (200 empl.) and the Museumsquartier (263 empl.). If this is the exact amount or if the number includes employees from other locations is unknown.

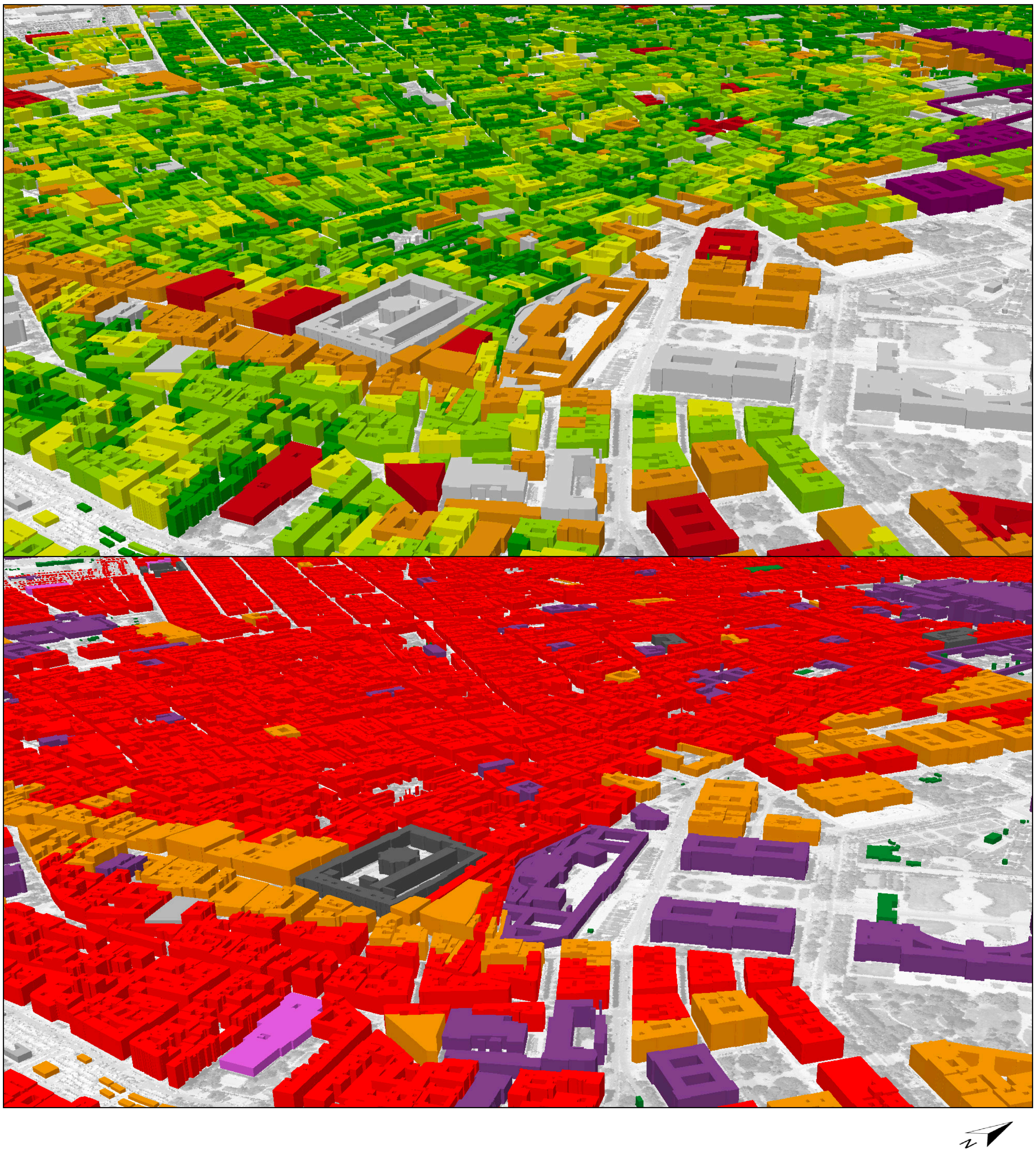

Figure 4.3: *Daytime Population 2013 - Digital Building Model (3D). Shown is a part of the inner city with the Hofburg, museums and the shopping street Mariahilfer Straße.* ***Top****: daytime population;* ***bottom****: building use. Legend see Figure 4.2. Source: own illustration.*

Table 4.2 shows the descriptive statistics of the daytime and nighttime population. The building with the most inhabitants of the daytime population (21,189, see Figure 4.4, left) is the Vienna University of Economics and Business (location of 2013) and the bio centre Althanstraße which belongs to the University of Vienna (both buildings are grouped together in one complex). It contains both students (18,661) and university employees (2,633). This high number can be traced to the high volume of the building as the model obviously distributes more people to larger buildings. The amount is even larger if the other buildings of a group (e.g. institutes of a university) are much smaller. Second comes the General Hospital of Vienna with 12,138 inhabitants (main building). This group is much more diverse as it includes trade school pupils (244), university employees (4,132 of the Medical University of Vienna), hospital employees (2,914), patients (969), students (3,060) and employees from the company register (873). The amount of university employees and students seems very high – this is due to the fact that the Medical University of Vienna also lectures its students at this hospital. Since it has a very high volume compared to the other institutes of that university, a large number of students is allocated there. On the other hand, the building with the most inhabitants during the night is a residential building complex at Rennbahnweg (see Figure 4.4, right). The high amount of 5,434 is also because of the large volume compared to other residential buildings.

Descriptive Statistics

	N	*Range*	*Min*	*Max*	*Sum*	*Mean*	*Std. Dev*	*Var*	*Skewness*		*Kurtosis*	
	Statistic	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Std.Error*	*Statistic*	*Std. Error*
population_ daytime	247560	21189	0	21189	1962262	7.93	91.596	8389.792	84.957	.005	14266	.010
population_ nighttime	247560	5434	0	5434	1758870	7.10	30.965	958.817	43.721	.005	5229.836	.010

Table 4.2: *Descriptive Statistics (daytime & nighttime). Analysis of the digital building model. Source: own illustration.*

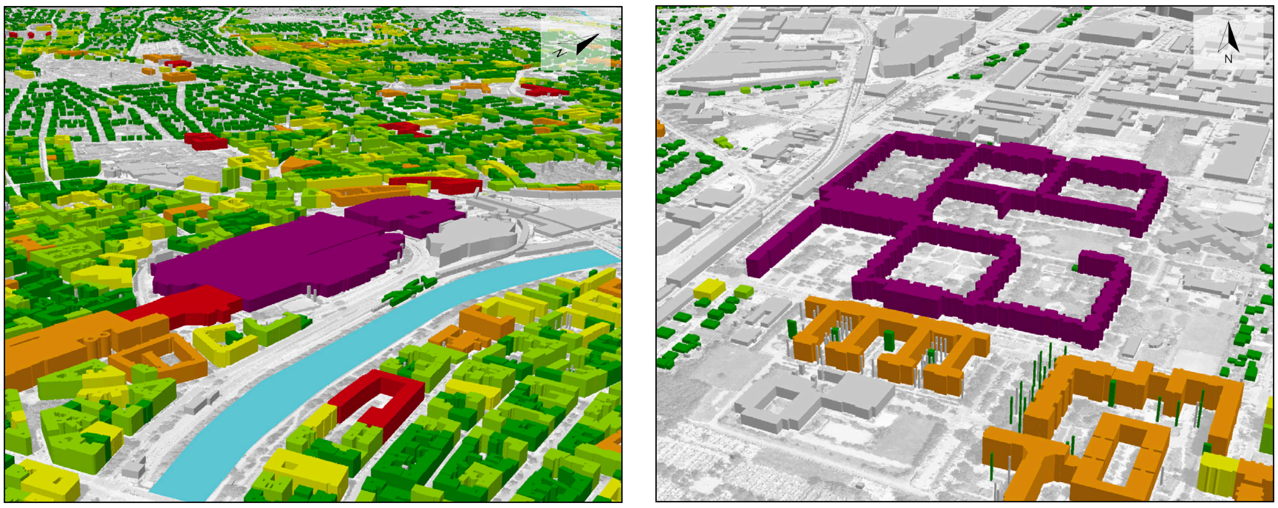

Figure 4.4: *Daytime & Nighttime Population - Examples.* ***Left****: daytime population: Vienna University of Economics and Business & Bio Centre Althahnstraße (centre);* ***right****: nighttime population: residential buildings at Rennbahnweg. Source: own illustration*

When looking at the total daytime and nighttime population (Table 4.2) the difference between the values can be explained by the balance of commuters, as mentioned before. The mean amount of inhabitants per building is for both populations similar. However, the standard deviation as well as the variance are for the daytime population much higher. This is because during the day there are much more people in structures such as industrial buildings or universities compared to residential buildings. The skewness for both populations is positive meaning that the frequency of positive values is higher than negative ones. Lastly, the kurtosis is especially high for the daytime population which means that the values spread close around the mean forming a steep distribution. The reason is that during the day people are working or studying and are therefore outside of their homes resulting in a much higher number of buildings with very little inhabitants compared to the night time when residential buildings are more populated.

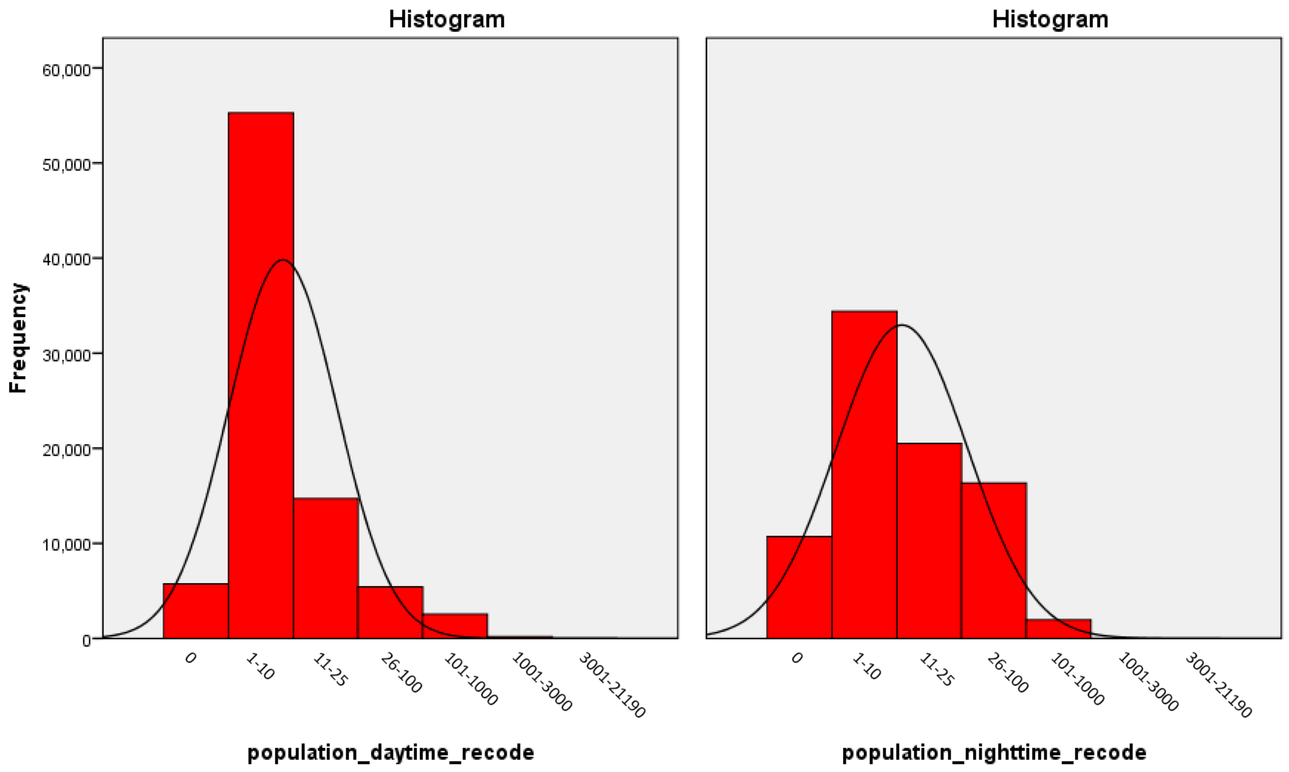

Figure 4.5: *Histogram Daytime & Nighttime Population. Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

population digital building model (daytime)

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	0	5747	6.8	6.8	6.8
	1 - 10	55275	65.9	65.9	72.7
	11 - 25	14705	17.5	17.5	90.2
	26 - 100	5432	6.5	6.5	96.7
	101 - 1000	2561	3.1	3.1	99.7
	1001 - 3000	182	.2	.2	100.0
	3001 - 21190	31	.0	.0	100.0
	Total	83933	100.0	100.0	

Table 4.3: *Frequency Table (daytime population). Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

Figure 4.5 as well as Table 4.3 and Table 4.4 show the frequency (= number of buildings) of the distributed populations which were grouped into 7 different classes. Those classes match with the ones from the figures already shown in this chapter. Because of the high number of very small structures (see page 43), buildings

population digital building model (nighttime)

Inhabitants		***Frequency***	***Percent***	***Valid Percent***	***Cumulative Percent***
Valid	0	10708	12.8	12.8	12.8
	1 - 10	34392	41.0	41.0	53.7
	11 - 25	20509	24.4	24.4	78.2
	26 - 100	16344	19.5	19.5	97.6
	101 - 1000	1959	2.3	2.3	100.0
	1001 - 3000	19	.0	.0	100.0
	3001 - 21190	2	.0	.0	100.0
	Total	83933	100.0	100.0	

Table 4.4: *Frequency Table (nighttime population). Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

with a height of less than 5m and an area of less than 30m² were excluded from these statistics. Despite the grouping of the values and the exclusion of cases the higher kurtosis of the daytime population can still be seen as well as a slightly positive skewness for both distributions. During the day most of the buildings (65.9%) have between 1 and 10 inhabitants whereas buildings with very high populations of between 1,001 and 3,000 are rare (0.2%). Even higher values can be found in 31 buildings. In case of the nighttime population the distribution is more even with 41% of the buildings housing between 1 and 10 residents. In the group with a high amount of inhabitants (1001-3000) there are just 19 buildings while in the group 3,001+ there are only 2 buildings (see Table 4.3).

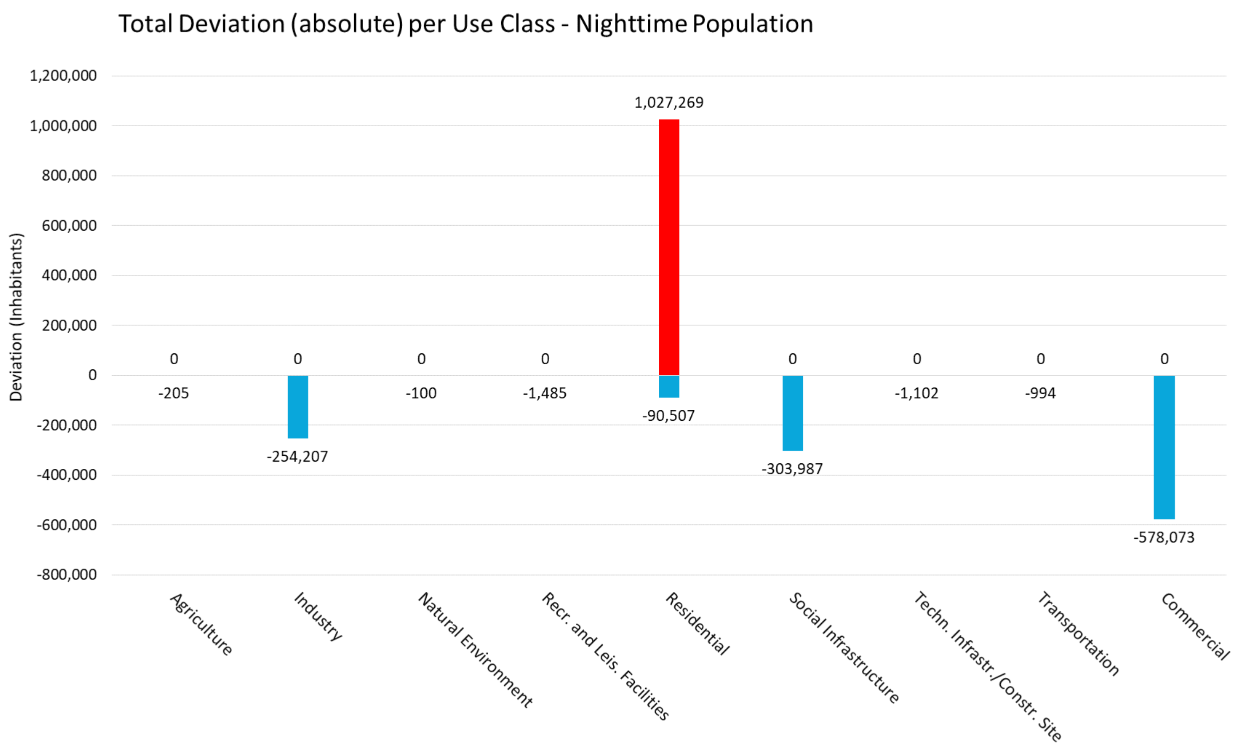

Figure 4.6: *Total Deviation (absolute) per Use Class - Nighttime Population. All cases selected. Source: own illustration.*

A look at the total (absolute) deviation between the daytime and nighttime population (Figure 4.6) further confirms the observation made before that during the night residential buildings are much more populated

while commercial and social infrastructure buildings lose population. The deviation was calculated by subtracting the nighttime from the daytime population per building and summarizing both the positive (red) and negative (blue) deviations. This means that positive deviations represent a surplus of the nighttime population while negative deviations are the opposite. It can be seen that in residential buildings there are cases where the daytime population is actually larger than the nighttime population. The reason for this lies mainly in the fact that when calculating the daytime population some of the residential buildings get inhabitants from address points (university institutes, schools, register of companies) allocated to regardless of its use class from the digital building model/land use layer. This indicates that the use classes from the source dataset (land use layer) are not always correct.
The disadvantage of depicting the daytime and nighttime population per building is that the building structures vary in size thus making a comparison difficult. Because of this the disaggregation results were aggregated to a 100x100m raster (see Figure 4.8). Here each cell depicts the number of residents per hectare as the area of one cell equals one hectare. During the day the concentration of population in the inner city stands out with shopping streets such as the Mariahilfer Straße and Kärntner Straße being visible. Also visible are large buildings like the Vienna University of Economics and Business or the General Hospital of Vienna. In contrast, the nighttime population has less hotspots and is distributed more evenly over Vienna. This can also be seen when depicting the grids in 3 dimensions (Figure 4.7).

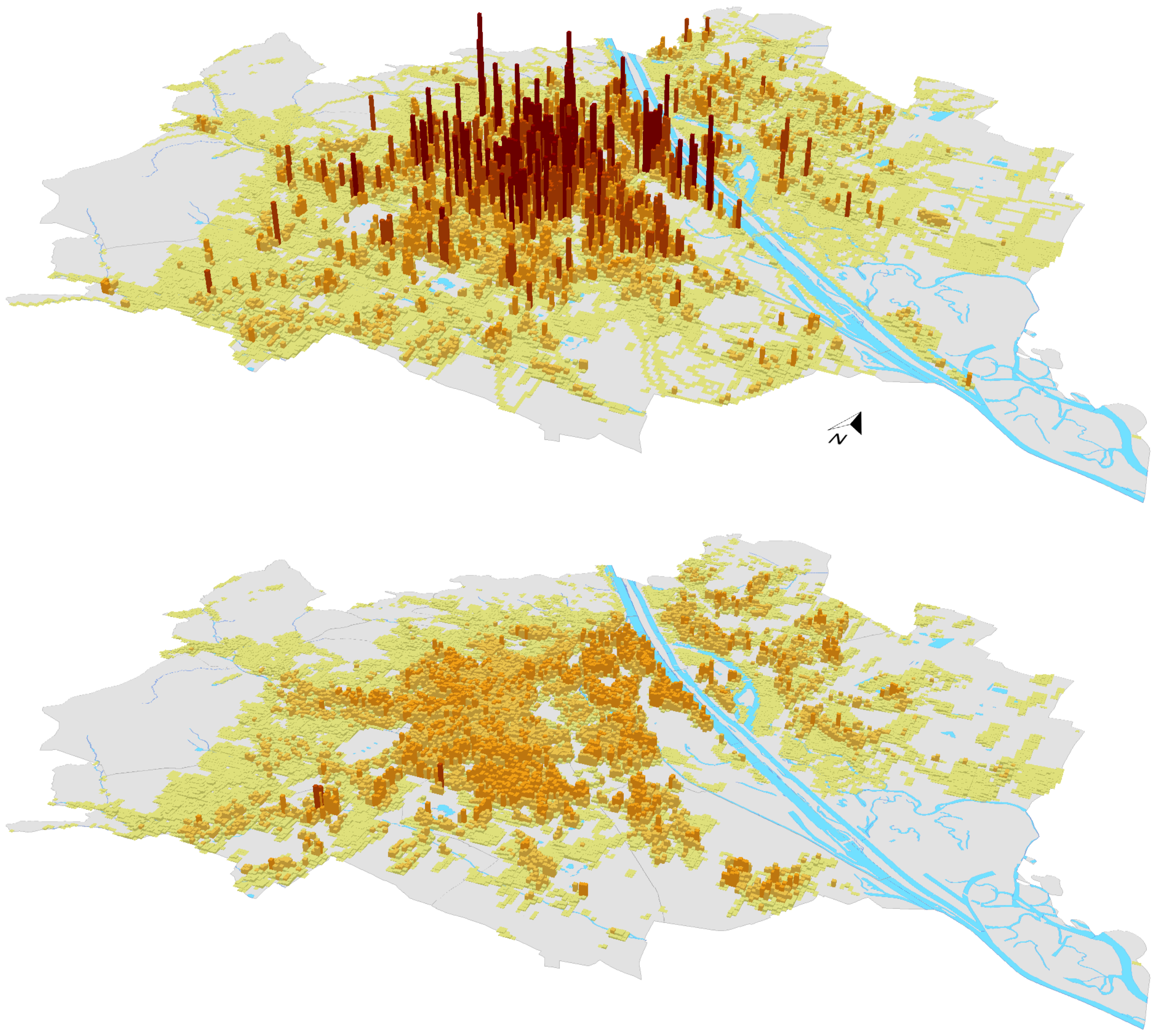

Figure 4.7: *Population Grid 2013 - Daytime & Nighttime (3D). Source: own illustration.*

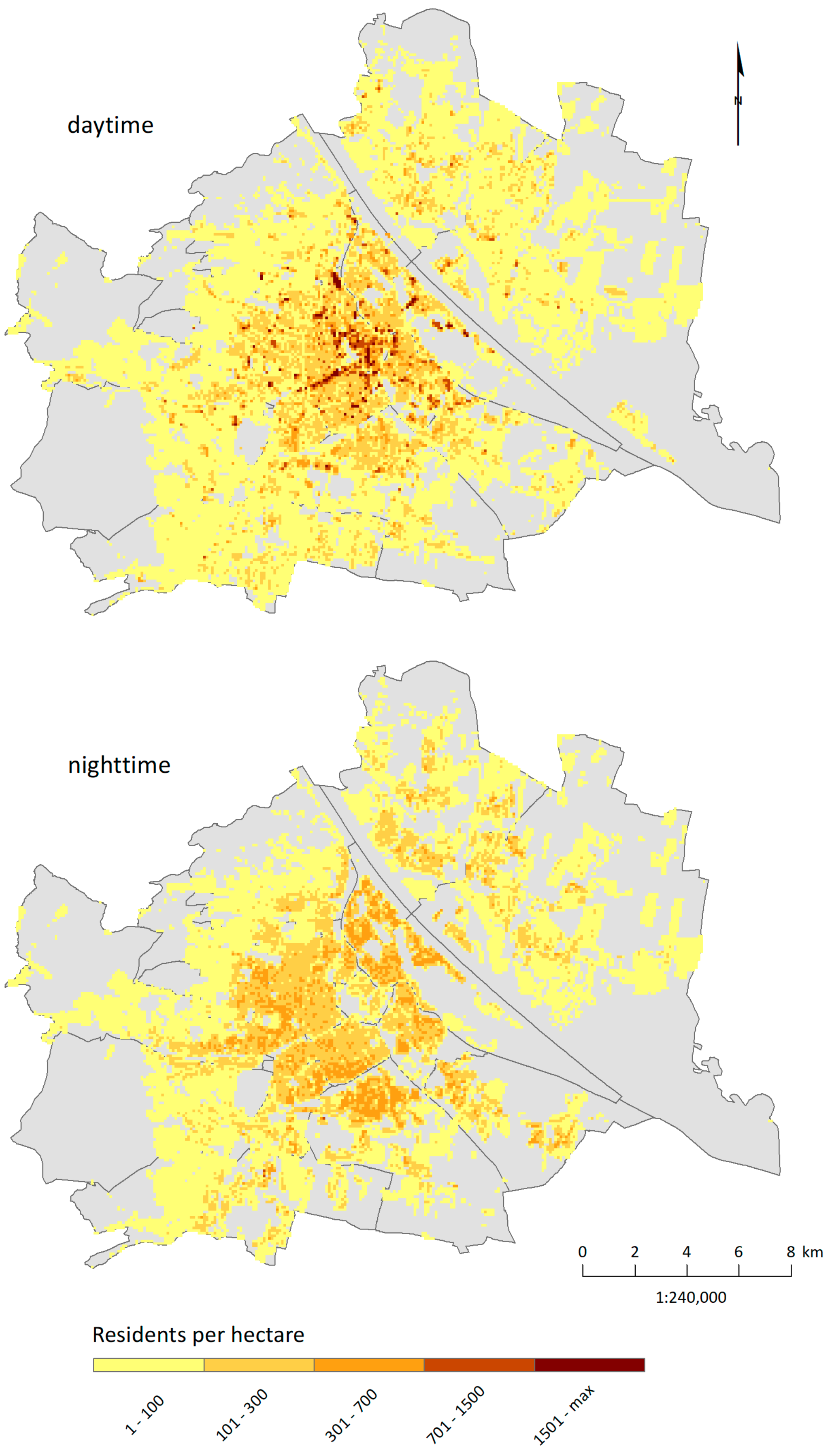

Figure 4.8: *Population Grid 2013 - Daytime & Nighttime. Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

4.2 Comparison with Model 02

When looking at the descriptive statistics of the deviations between model 02 (all data, weighting by area) and model 01 it can be seen that the standard deviations are relatively small with 22.9 for the digital building model and 40.8 for the population grid (Table 4.5). The skewness is negative indicating that the frequency of the negative values is higher than the positive ones, meaning that underestimations of inhabitants occur more often. However, the skewness (-5.6) of the grid is not distinctive. A look at the sum of the deviations shows that it equals 0. This is due to the fact that the same total population is distributed in both models. Finally, the high kurtosis is a sign that most of the values are near the mean which is 0. A look at Table 4.6 and Table 4.7 confirms this where 96.4% of the deviations (digital building model) lie only between -19 and 20. The percentage for the grid is still 87.8%.

Descriptive Statistics

	N	*Range*	*Min*	*Max*	*Sum*	*Mean*	*Std. Dev*	*Var*	*Skewness*		*Kurtosis*	
	Statistic	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Std.Error*	*Statistic*	*Std. Error*
deviation_02	247560	4831	-2830	2001	0	.00	22.907	524.743	-21.238	.005	2991.950	.010
dev_02_grid	45375	2168	-1217	951	0	.00	40.800	1664.636	-5.554	.011	175.261	.023

Table 4.5: *Descriptive Statistics (deviation model 02 - digital building model & grid). Source: own illustration.*

deviations digital building model

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	-2830 to -1500	6	.0	.0	.0
	-1499 to -200	201	.2	.2	.2
	-199 to -100	280	.3	.3	.6
	-99 to -20	1454	1.7	1.7	2.3
	-19 to 20	80934	96.4	96.4	98.7
	21 to 100	790	.9	.9	99.7
	101 to 200	139	.2	.2	99.8
	201 to 1500	126	.2	.2	100.0
	1501 to 2001	3	.0	.0	100.0
	Total	83933	100.0	100.0	

Table 4.6: *Frequency Table (deviation model 02 - digital building model). Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

deviations grid

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	-1217 to -200	176	.4	.4	.4
	-199 to -100	309	.7	.7	1.1
	-99 to -20	2584	5.7	5.7	6.8
	-19 to 20	39820	87.8	87.8	94.5
	21 to 100	2074	4.6	4.6	99.1
	101 to 200	279	.6	.6	99.7
	201 to 951	133	.3	.3	100.0
	Total	45375	100.0	100.0	

Table 4.7: *Frequency Table (deviation model 02 - grid). All cases selected. Source: own illustration.*

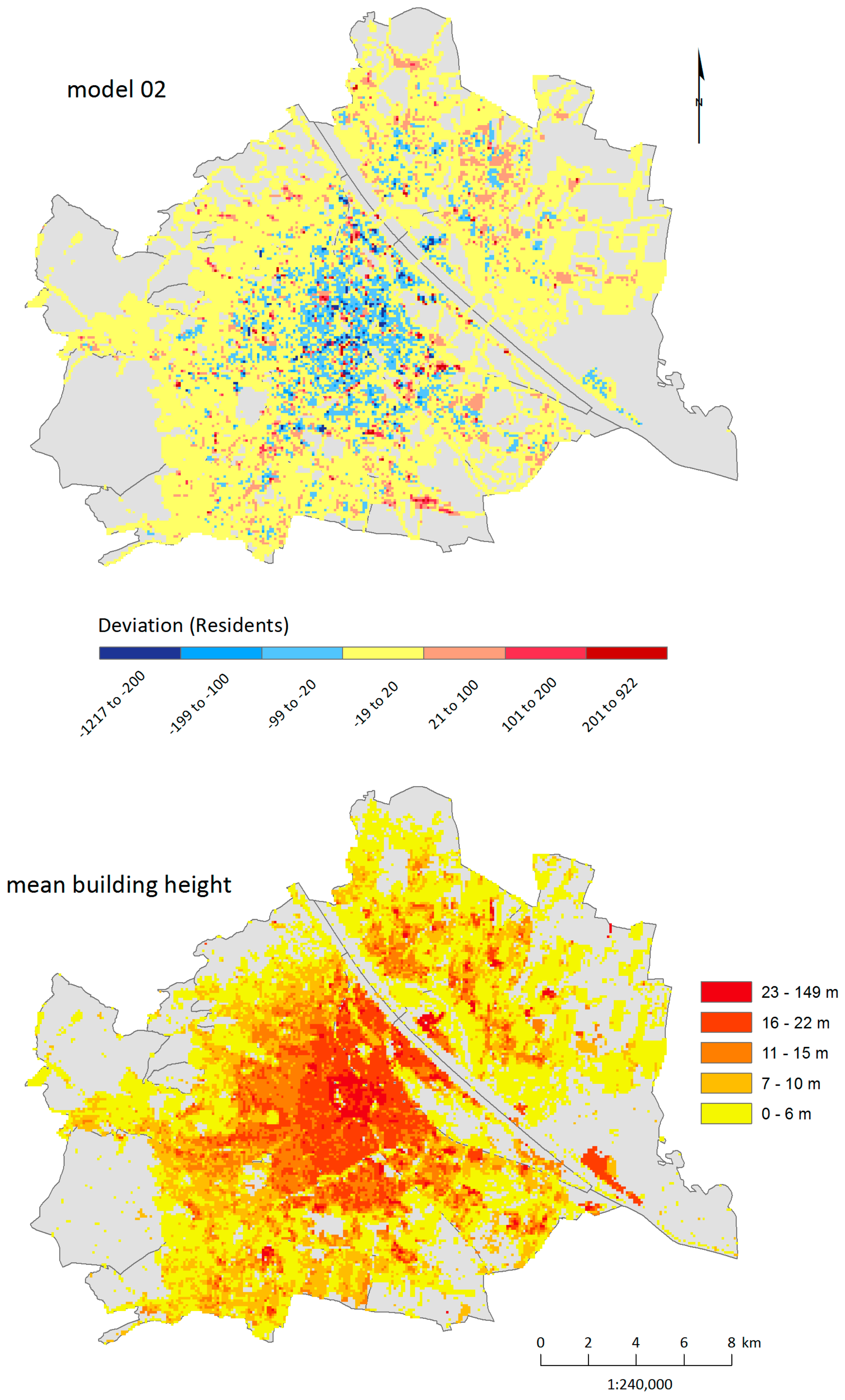

Figure 4.9: *Model 02 - Deviation & Mean Building Height (grid). Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

When looking at the deviation grid which was calculated by subtracting the population of model 02 from the population of model 01 per 100m grid cell it can be seen that negative deviations (where model 02 distributes less population than model 01) occur primarily near the city centre where also the mean building heights have the highest values (Figure 4.9). Positive deviations on the other hand are common throughout the city. The highest negative deviation (-2,830) of the digital building model can be found at the Uno City of the United Nations (Figure 4.10, left - total population model 01: 5,000) which mainly houses working population but also pupils of trade schools and young children since a kindergarten is located in the building. The highest positive deviation (2,001) can be found at Altes AKH which mainly houses students but also small children (kindergarten) and university employees (total population model 01: 4,350).

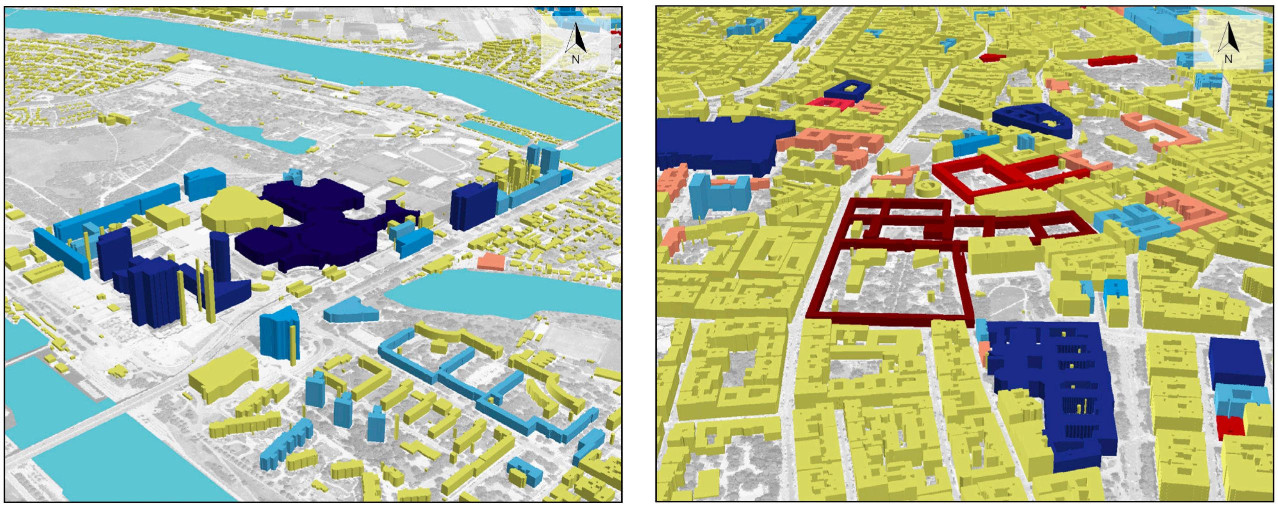

Figure 4.10: *Model 02 - Examples.* ***Left****: highest negative deviation;* ***right****: highest positive deviation. Legend see Figure 4.9. Source: own illustration.*

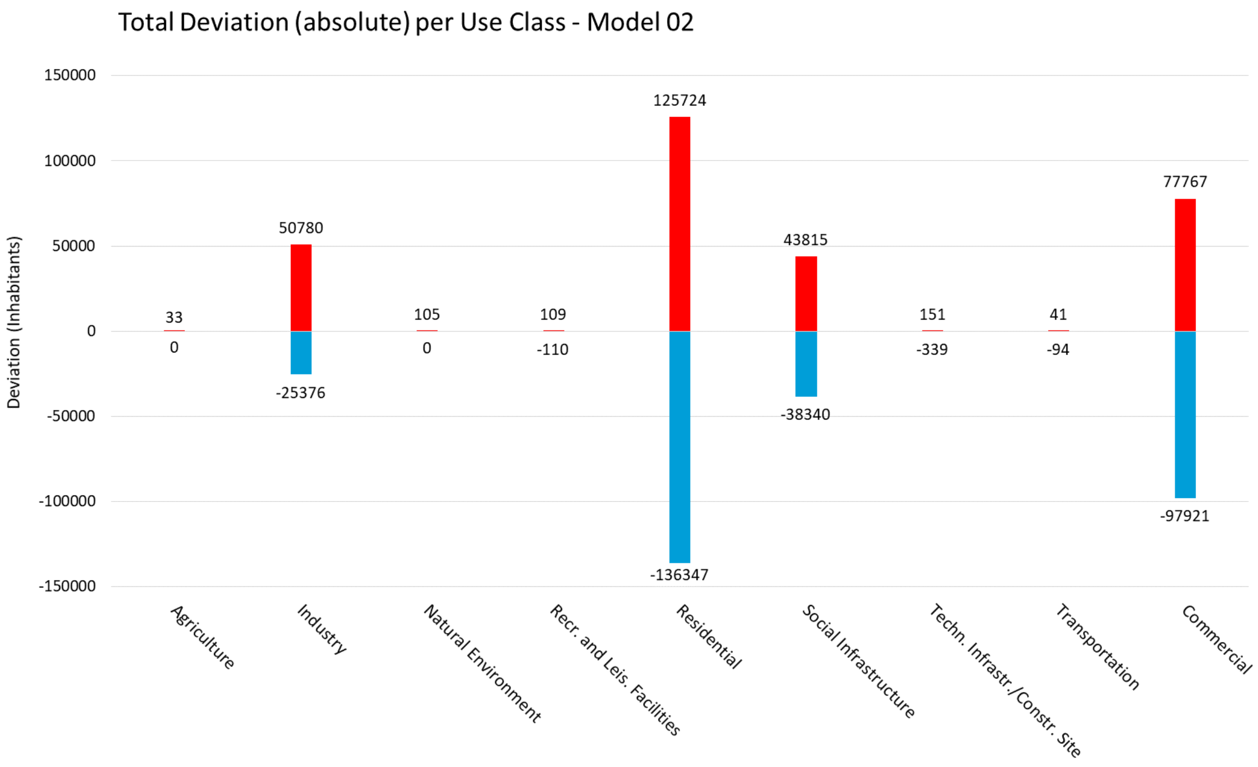

Figure 4.11: *Total Deviation (absolute) per Use Class - Model 02. All cases selected. Source: own illustration.*

Since model 02 differs from model 01 only through the weighting of the distribution (area vs. volume) it can be assumed that there is a correlation between the deviation and the volume of the buildings. Accordingly, if the distribution is weighted by the building footprints the negative deviation should be higher in buildings with a larger volume. A notable influence of the building use is not expected which can be confirmed by looking at Figure 4.11 which shows the total absolute deviation (positive and negative) per use class. The deviation was calculated by subtracting the population of model 01 from the one of model 02 per building and summarizing both the positive (red) and negative (blue) deviations. Here the positive and associated negative deviations are roughly the same and are especially pronounced for the use classes “Residential“, “Commercial“, “Social Infrastructure“ and “Industry“ which are essentially the ones where most of the population is distributed. In order to quantify the correlation between the deviation and the volume, a curve estimation as well as a Pearson Correlation coefficient was calculated (see Figure 4.12 and Table 4.8). The result shows only a weak correlation of -0.313 (1.0 would be a perfect correlation). This can also be seen graphically in the curve estimation which depicts the x/y-positions of all cases (=observed) with the abscissa being the building volume and the ordinate axis being the deviation (residents) of the buildings. The cases are dispersed in both direction showing only a slight trend to negative deviations with increasing volume.

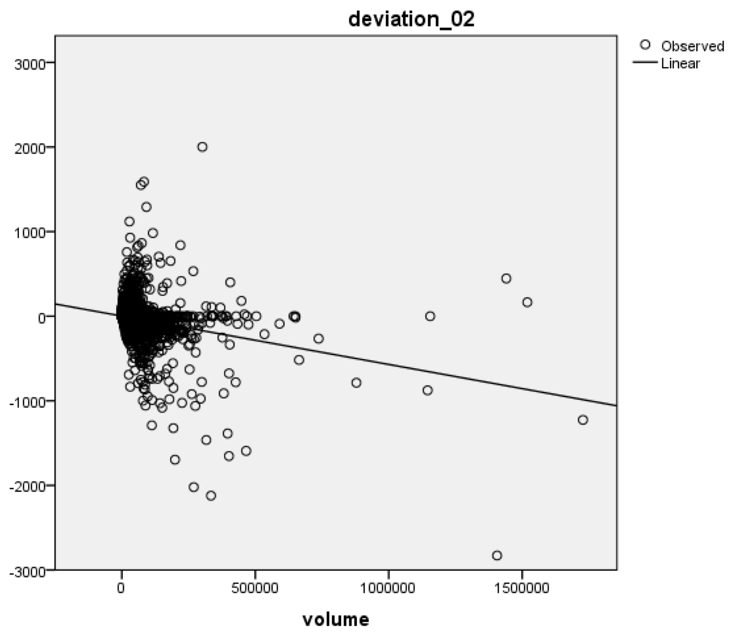

Figure 4.12: *Curve Estimation - Total Deviation & Volume. All cases selected. Source: own illustration.*

Correlations

		deviation_02	*volume*
	Pearson Correlation	1	-.313**
deviation_02	Sig. (2-tailed)		.000
	N	247560	247560
	Pearson Correlation	-.313**	1
volume	Sig. (2-tailed)	.000	
	N	247560	247560

**. Correlation is significant at the 0.01 level (2-tailed).

Table 4.8: *Pearson Correlation - Total Deviation & Volume. All cases selected. Source: own illustration.*

At first glance this seems illogical but the weak correlation can be explained by considering that different population groups with varying sizes are distributed among different building groups. For example, university students which are a relatively large group (189,877) are distributed among the relatively small group of university buildings (139) which means that even small buildings will get a higher number of students leading to higher possible deviations. Furthermore, a part of the employees (114,795) is distributed from the company register directly to the buildings without weighting. In order to avoid these problems the correlation for only one population group was calculated – the unemployed. This group is distributed among all residential buildings for the whole of Vienna using a weighting factor (area or volume). Since both numbers for the unemployed (120,815) and residential buildings (214,267) are high there are enough cases to calculate the correlation which was done by only selecting the residential buildings. Now, Figure 4.13 and Table 4.9 show that there is a strong negative Pearson Correlation of -0.834 meaning that with increasing building volume the deviation between model 02 and 01 becomes more negative. This confirms the assumption that the population of large buildings tends to be underestimated when it is distributed weighted only by the building footprints instead of the volumes.

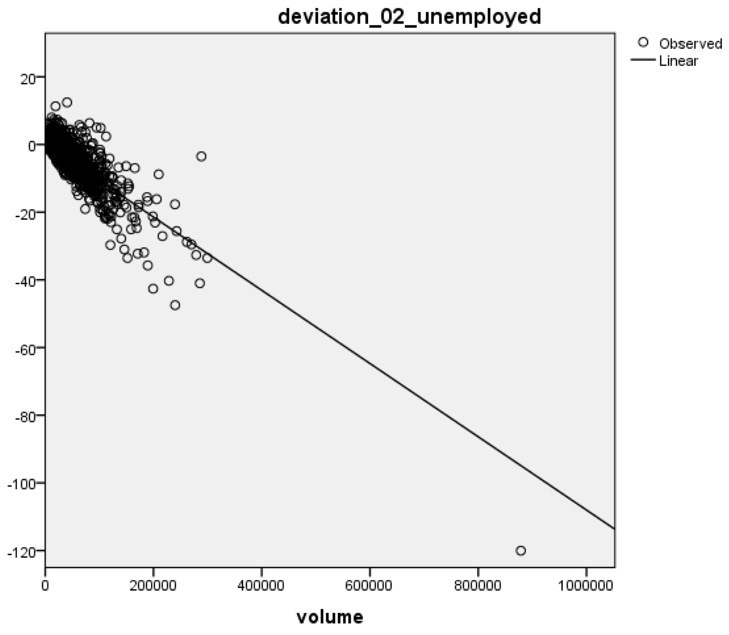

Figure 4.13: *Curve Estimation - Deviation Unemployed & Volume. Only residential buildings selected. Source: own illustration.*

Correlations

		deviation_02_unemployed	*volume*
deviation_02_unemployed	Pearson Correlation	1	-.834**
	Sig. (2-tailed)		.000
	N	214267	214267
volume	Pearson Correlation	-.834**	1
	Sig. (2-tailed)	.000	
	N	214267	214267

**. Correlation is significant at the 0.01 level (2-tailed).

Table 4.9: *Pearson Correlation - Deviation Unemployed & Volume. Only residential buildings selected. Source: own illustration.*

4.3 Comparison with Model 03

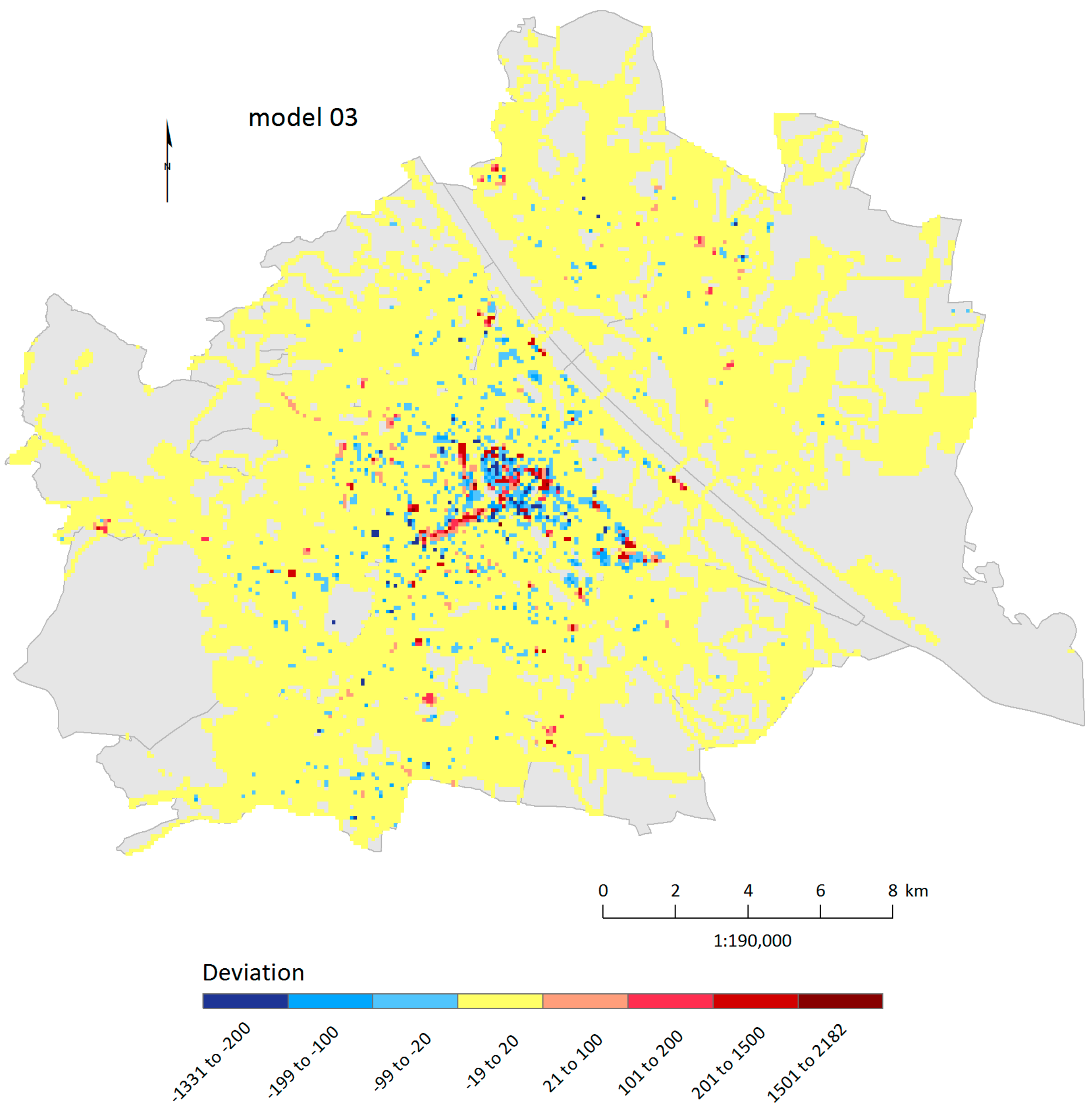

Figure 4.14: *Model 03 - Deviation (grid). Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

Model 03 distributes the population like model 01 weighted by volume with the only difference being that model 03 excludes the company register. A look at the descriptive statistics of the deviations (Table 4.10) shows that in contrast to the deviations of model 02 the skewness is positive, indicating that in model 03 positive deviations (= overestimation of inhabitants) are more frequent that negative ones. Also, the kurtosis is for both the digital building model and grid much higher than the one of model 02 meaning that in model 03 more cases lie close to the mean which is 0. This can be seen as a sign that the deviations of model 03 are lower which means the population distribution is more similar to model 01. The frequency tables of model 03 (Table 4.11 and Table 4.12) confirm this assumption as 98.2% of the buildings and 96.6% of the grid cells have a very small deviation of only +/- 20 inhabitants (in contrast to model 02 with 96.4% and 87.8%).

Descriptive Statistics

	N	Range	Min	Max	Sum	Mean	Std. Dev	Var	Skewness		Kurtosis	
	Statistic	Statistic	Statistic	Statistic	Statistic	Statistic	Statistic	Statistic	Statistic	Std.Error	Statistic	Std. Error
deviation_03	247560	6690	-1756	4934	0	.00	26.199	686.402	71.789	.005	9584.899	.010
dev_03_grid	45375	3514	-1331	2182	0	.00	43.179	1864.452	15.177	.011	699.471	.023

Table 4.10: Descriptive Statistics (deviation model 03 - digital building model & grid). Source: own illustration.

deviations digital building model

Inhabitants		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-1756 to -1500	1	.0	.0	.0
	-1499 to -200	110	.1	.1	.1
	-199 to -100	126	.2	.2	.3
	-99 to -20	894	1.1	1.1	1.3
	-19 to 20	82453	98.2	98.2	99.6
	21 to 100	183	.2	.2	99.8
	101 to 200	57	.1	.1	99.9
	201 to 1500	95	.1	.1	100.0
	1501 to 4934	14	.0	.0	100.0
	Total	83933	100.0	100.0	

Table 4.11: Frequency Table (deviation model 03 - digital building model). Select Cases where building height > 5m and building area > 30m². Source: own illustration.

deviations grid

Inhabitants		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-1331 to -200	79	.2	.2	.2
	-199 to -100	160	.4	.4	.5
	-99 to -20	838	1.8	1.8	2.4
	-19 to 20	43848	96.6	96.6	99.0
	21 to 100	233	.5	.5	99.5
	101 to 200	86	.2	.2	99.7
	201 to 1500	126	.3	.3	100.0
	1501 to 2182	5	.0	.0	100.0
	Total	45375	100.0	100.0	

Table 4.12: Frequency Table (deviation model 03 - grid). All cases selected. Source: own illustration.

Figure 4.14 shows the deviation grid of model 03 where the population of model 01 is subtracted from the population of model 03 per grid cell. Here it can be seen that the inhabitants of commercial buildings (use class "Industry" and "Commercial") tend to be overestimated. This is because in model 03 there is a higher remaining working population which is distributed because the employees of the company register are missing. Prominent in Figure 4.14 are for example the Mariahilfer Straße which can be seen as long red line near the centre and the 1st district directly in the centre. Both contain many commercial buildings resulting in positive deviations. On the other hand, residential buildings tend to be underestimated. A reason for this could be that through the company register employees are also distributed to residential buildings (since they can include also small businesses). Those employees are of course missing in model 03 because it distributes the working population exclusively to commercial buildings and hospitals. But in total the deviations between model 01 and 03 are small. The buildings with the highest negative deviation and highest positive deviation can be seen in Figure 4.15. Interestingly, the building with the highest negative deviation of -1,756 is a shopping mall in the 15th district (use class "Commercial"). It does not match with the company register so in model 01 and 03 the remaining working population and also pupils of trade schools are distributed to this building weighted by its volume resulting in 5,403 inhabitants in model 01 and 3,647 inhabitants in model 03. The building with the highest positive deviation (4,934) is also a shopping mall, located at the subway station Wien Mitte in the 3rd district. Here in model 01 there are 824 employees distributed through the company register (total population: 1,411), excluding it from receiving inhabitants from the remaining working population since it has already got its employees. In model 03 the building receives its inhabitants mainly from the remaining working population which is weighted by its large volume resulting in a large population count.

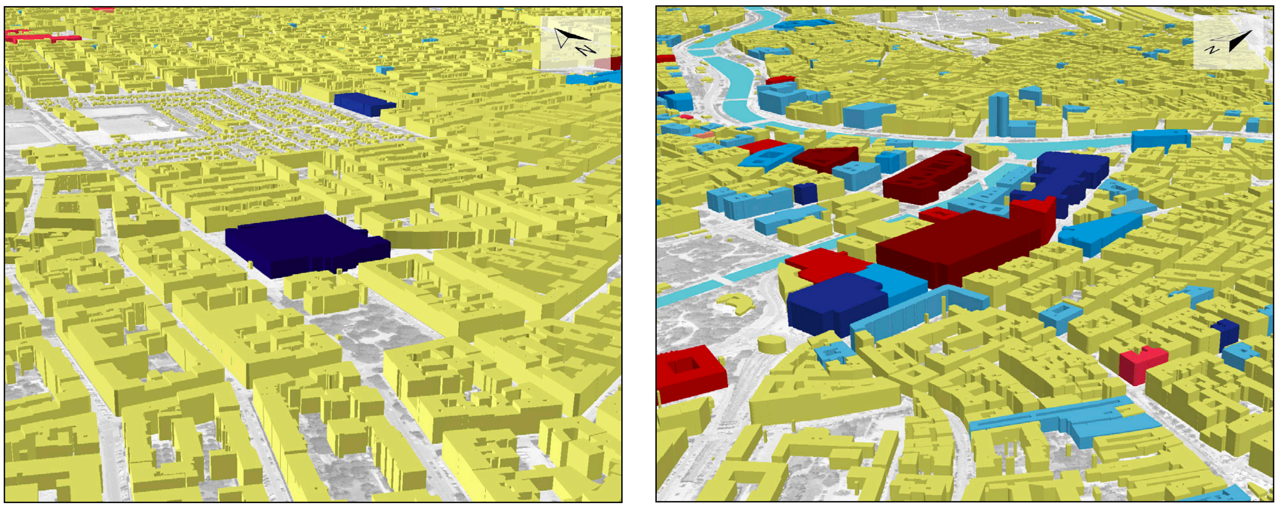

Figure 4.15: *Model 03 - Examples.* ***Left****: highest negative deviation;* ***right****: highest positive deviation. Legend see Figure 4.9. Source: own illustration.*

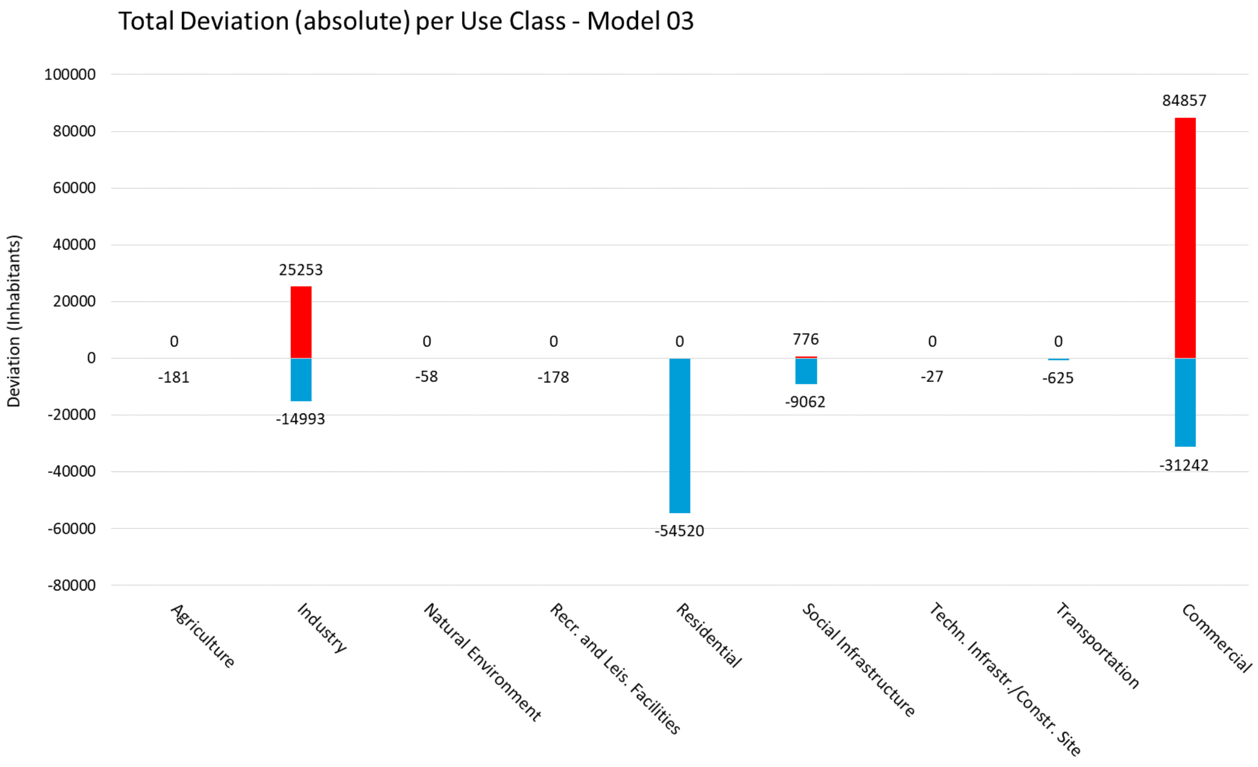

Figure 4.16: *Total Deviation (absolute) per Use Class - Model 03. All cases selected. Source: own illustration.*

Finally, Figure 4.16 shows the total absolute negative and positive deviation per use class. It confirms that positive deviations which are driven by the distribution of the remaining working population are dominant in commercial buildings while especially residential buildings get negative deviations. Another use class with primarily negative deviations is Social Infrastructure. A reason for this could be that like with the residential buildings there occur matchings with the company register resulting in distributed employees in model 01 who are missing in model 03. Additionally, the remaining working population is not distributed among all social infrastructure buildings but only to the hospitals meaning that the effects of overestimation apply less to this use class.

4.4 Comparison with Model 04

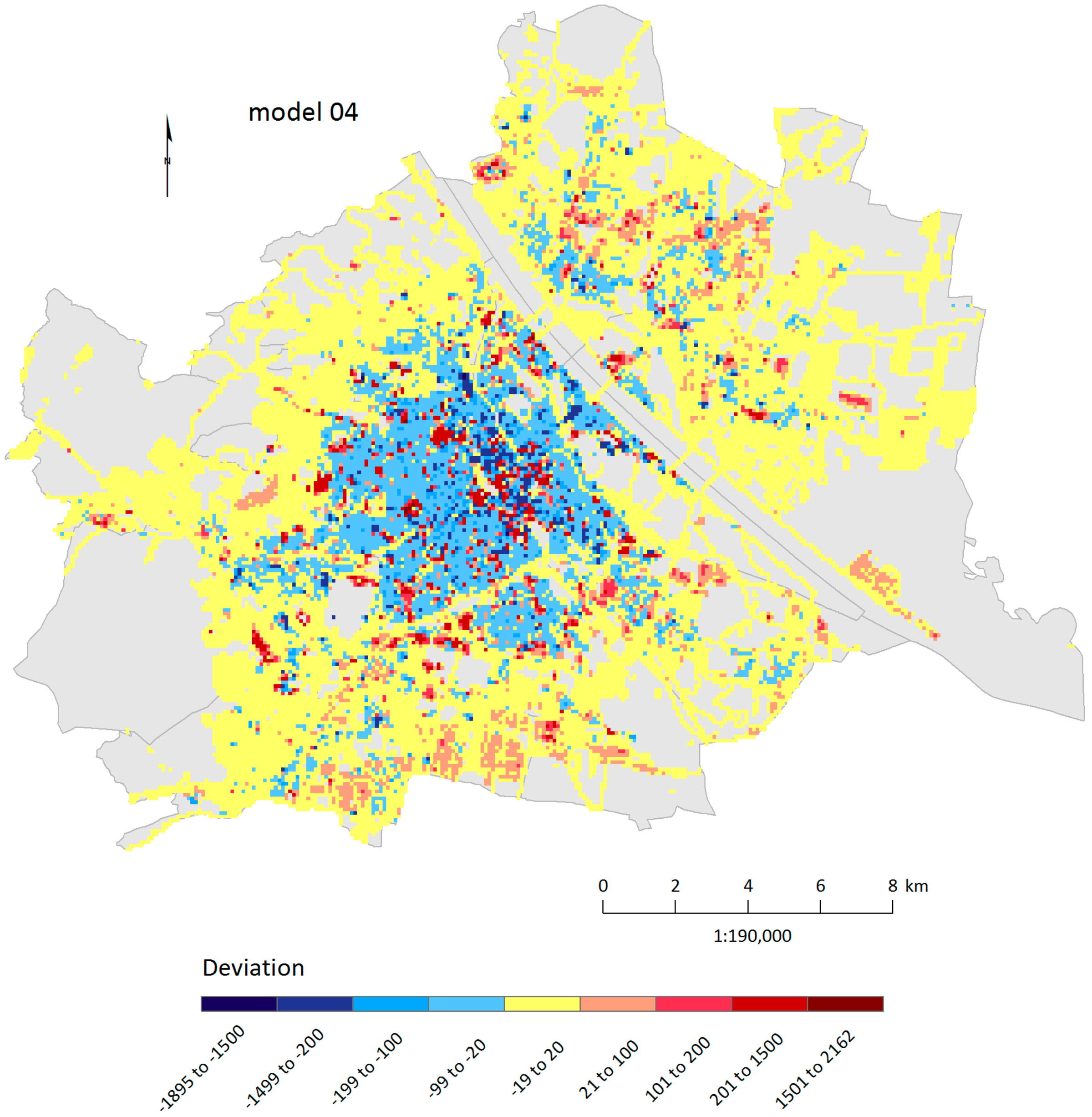

Figure 4.17: *Model 04 - Deviation (grid). Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

Unlike the other models discussed so far model 04 excludes all auxiliary data except the balance of commuters in order to distribute the same total population in all models for a better comparability. No auxiliary data means that the locations of schools, universities, hospitals etc. are unknown which means that for example school children and students have to be distributed over all social infrastructure buildings. As mentioned in chapter 2.1, those buildings can also include churches or theatres resulting in a more inaccurate distribution of population. Noticeable in Table 4.13 is the fact that the sum of the deviations (digital building model) is in contrast to the other models not 0 but 4,284. This is because in model 04 the public transportation network is missing so the employees of Wiener Linien could not be distributed to the network. Instead they are allocated like all employees to the commercial buildings. In the end, the total population of model 04 (digital building model) equals 1,966,546 which is the total daytime population while the total population of model 01 is missing the (working) employees of Wiener Linien (=4,284) as they are allocated not to the buildings but to the public transportation network. After aggregating the populations of model 01 and 04 to the grid it can be seen that the sum of the deviations between both grids is again close to 0 because now in model 01 the employees from the public transportation network are also included meaning that both grids have the same total population. Interestingly, the kurtosis of model 04 (digital building model) is higher than the one of model 03 although the latter has a smaller absolute deviation. This may be due to the different total populations of both models resulting in different mean values. A look at the kurtosis of the grid shows again that the one of model 03 (699) is much higher than the one of model 04 (123).
Further, the frequency tables (Table 4.14 & Table 4.15) show that in model 04 only 93.4% (digital building model) and 81.8% (grid) of the cases lie within +/- 20 inhabitants which is less than in the previously discussed models.

Descriptive Statistics

	N	*Range*	*Min*	*Max*	*Sum*	*Mean*	*Std. Dev*	*Var*	*Skewness*		*Kurtosis*	
	Statistic	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Std.Error*	*Statistic*	*Std. Error*
deviation_04	247560	21518	-7417	14100	4284	.02	65.663	4311.647	53.225	.005	11166.593	.010
dev_04_grid	45375	4057	-1895	2162	-4	.00	92.817	8615.077	4.601	.011	122.596	.023

Table 4.13: *Descriptive Statistics (deviation model 04 - digital building model & grid). Source: own illustration.*

deviations digital building model

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	-7417 to -1500	17	.0	.0	.0
	-1499 to -200	427	.5	.5	.5
	-199 to -100	329	.4	.4	.9
	-99 to -20	2139	2.5	2.5	3.5
	-19 to 20	78424	93.4	93.4	96.9
	21 to 100	1570	1.9	1.9	98.8
	101 to 200	392	.5	.5	99.2
	201 to 1500	586	.7	.7	99.9
	1501 to 14100	49	.1	.1	100.0
	Total	83933	100.0	100.0	

Table 4.14: *Frequency Table (deviation model 04 - digital building model). Select Cases where building height > 5m and building area > 30m². Source: own illustration.*

deviations grid

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	-1895 to -1500	4	.0	.0	.0
	-1499 to -200	451	1.0	1.0	1.0
	-199 to -100	551	1.2	1.2	2.2
	-99 to -20	4026	8.9	8.9	11.1
	-19 to 20	37118	81.8	81.8	92.9
	21 to 100	2032	4.5	4.5	97.4
	101 to 200	515	1.1	1.1	98.5
	201 to 1500	664	1.5	1.5	100.0
	1501 to 2162	14	.0	.0	100.0
	Total	45375	100.0	100.0	

Table 4.15: *Frequency Table (deviation model 04 - grid). All cases selected. Source: own illustration.*

A look at Figure 4.17 also illustrates the fact that the deviations are more distinctive in model 04 than the ones of model 02 or 03. Overestimations primarily occur at social infrastructure (=schools, universities, hospitals, theatres etc.), industrial and commercial buildings while residential buildings are almost exclusively underestimated. The overestimation of industrial and commercial buildings and simultaneous underestimation of residential buildings could be affiliated to the same effects found in model 03 because model 04 does not use a company register too. However, in model 03 the share of commercial buildings which are underestimated compared to those which are overestimated is quite high. The reason for this could be that in model 04 the age group of the working population (15-64 years) had to be extended to the pupils from the age of 10 because these groups share the same building classes (especially social infrastructure) and there are no auxiliary data to distinct between e.g. schools and hospitals. Accordingly, this extended age group had to be distributed not only among commercial buildings but also to social infrastructure buildings. Because of this a share of the working population is also distributed to these buildings which then is missing in the commercial buildings. The reason why this concerns especially commercial buildings and not industrial buildings is that the distribution is done per district and the industrial buildings are mainly located at the edge of Vienna where less buildings of social infrastructure exist. On the other hand, the commercial buildings are often concentrated in the inner districts with a higher density of social infrastructure buildings. Additionally, the latter have frequently larger volumes than commercial buildings and therefore get a higher share of the population. This is also the reason why in model 04 buildings of social infrastructure are highly overestimated – hence they receive not only additional inhabitants from the employees but this population group is also allocated to all buildings of social infrastructure (including e.g. theatres, operas and churches) because of the missing location data. In contrast, in model 01 only those social infrastructure buildings get population allocated which could be identified as school or university by a point-shapefile. The rest (theatres, churches, museums etc.) do not get population because these buildings are often very large while at the same time they house relatively few inhabitants during the day.

Figure 4.18 shows the buildings with the highest negative and positive deviation of model 04 which have interestingly both the use class "Social Infrastructure". The former is a building complex which houses the Vienna University of Economics and Business as well as institutes of the University of Vienna. It has a negative deviation of -7,417 (total population model 01: 21,189). This is because in model 01 it houses a large amount of mainly students who were allocated to the building because of the university point-shapefile while in model 04 the age group 10-64 (including students) are distributed among all social infrastructure buildings

resulting in a smaller number of inhabitants. The building with the highest deviation of 14,100 is the Messe Wien which houses fairs and congresses (Figure 4.18, right). In model 01 no population was allocated there because there are no address points intersecting while in model 04 it gets as a social infrastructure building due to its large volume a high share of the age group 10-64 (14,100 inhabitants).
Lastly, Figure 4.19 shows the total absolute negative and positive deviation per use class. It confirms the observation that without the use of auxiliary data residential buildings tend to be underestimated while industry, commercial and social infrastructure buildings tend to be overestimated.

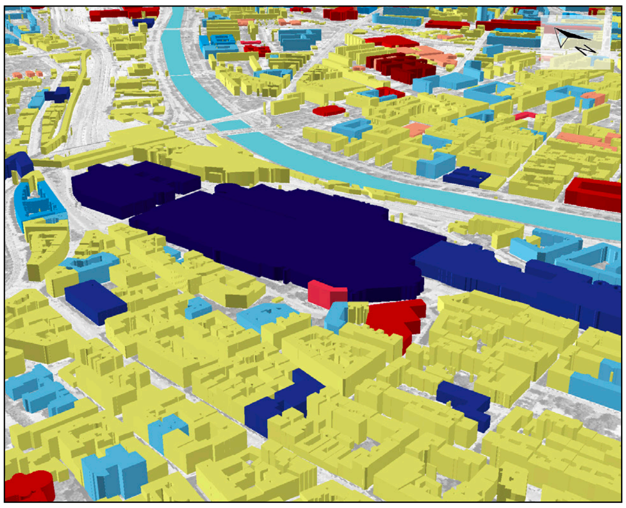
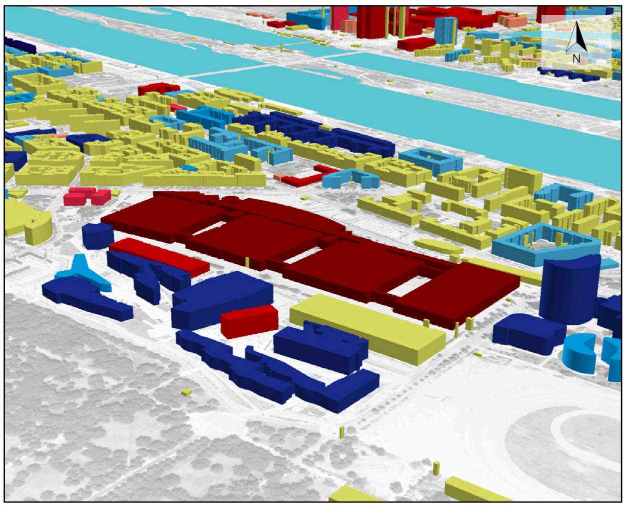

Figure 4.18: *Model 04 - Examples.* ***Left****: highest negative deviation;* ***right****: highest positive deviation. Legend see Figure 4.9. Source: own illustration.*

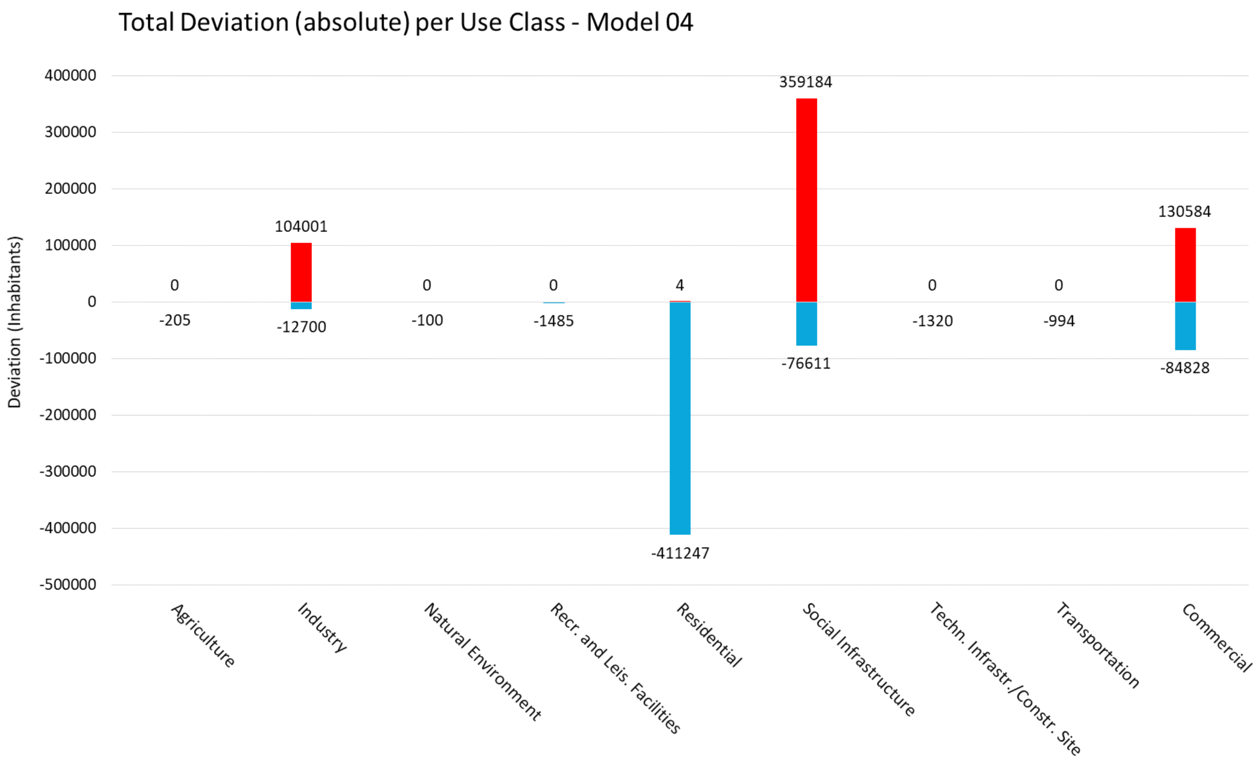

Figure 4.19: *Total Deviation (absolute) per Use Class - Model 04. All cases selected. Source: own illustration.*

4.5 Comparison with Model 05

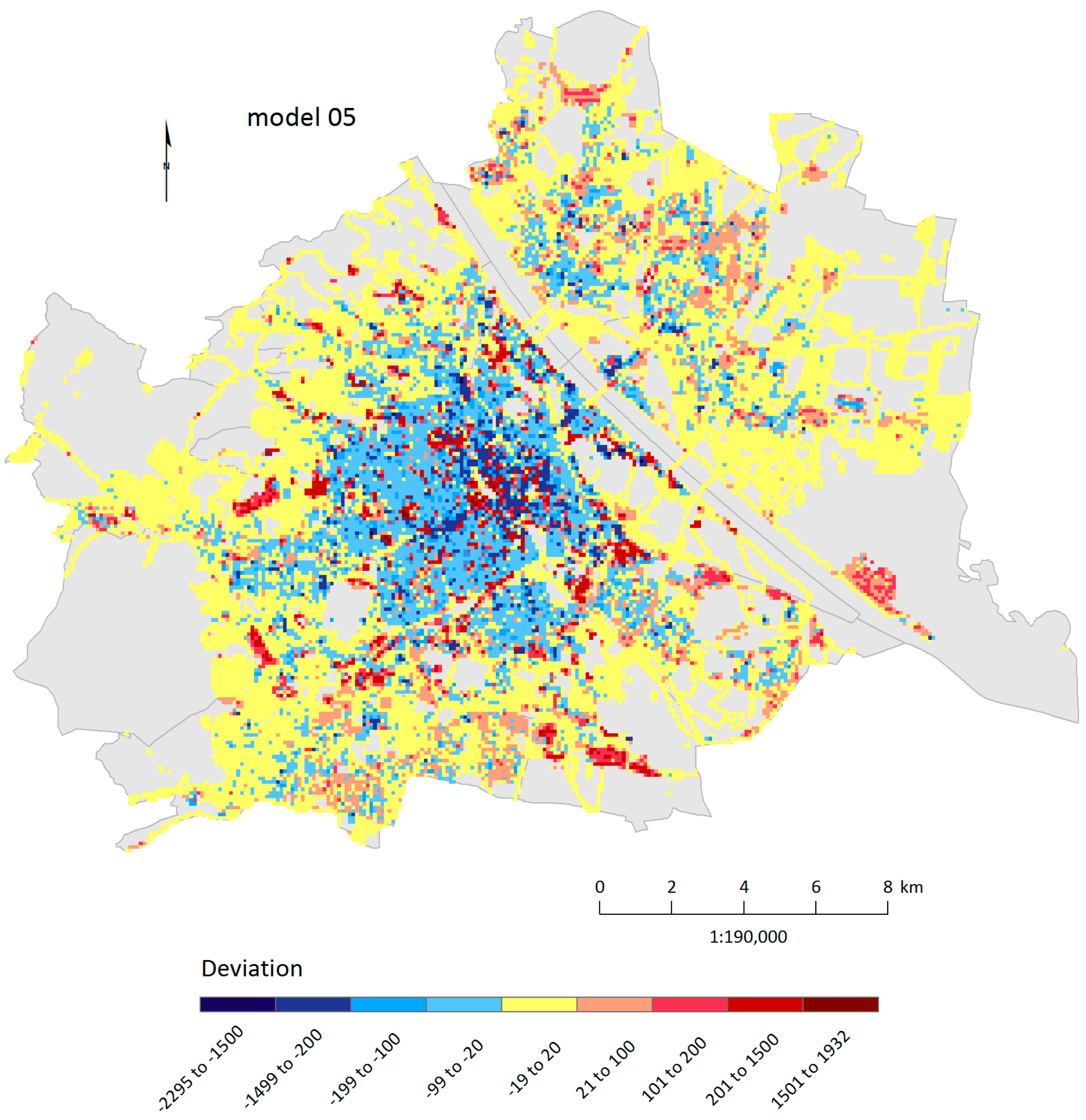

Figure 4.20: *Model 05 - Deviation (grid). Source: own illustration. Coordinate System: ETRS 1989 LAEA.*

The last model is model 05 where no auxiliary data except the balance of commuters was used like in model 04 but with the difference that the population was distributed to the Land Use 2012 dataset (see Figure 2.1 on page 8) which is the basis for the use classes of the digital building model. In contrast to the latter the land use layer consists of areas representing whole building blocks, water bodies or parks. Therefore, it lacks information on individual buildings including building heights. Because of this the deviations from model 01 are expected to be higher. A look at Figure 4.20 confirms this – it can be seen that the colour distribution compared to model 04 is similar (because the same effects described in chapter 4.4 occur also here) but more distinctive with higher deviations. Accordingly, the kurtosis of the deviation grid (Table 4.16) is even smaller than in model 04 meaning that the distribution curve is flatter and the values are more scattered around the mean. Table 4.17 also shows that only 75.9% of the deviations lie within +/- 20 inhabitants, being the smallest percentage compared to the other models.

Descriptive Statistics

	N	***Range***	***Min***	***Max***	***Sum***	***Mean***	***Std. Dev***	***Var***	***Skewness***		***Kurtosis***	
	Statistic	***Statistic***	***Statistic***	***Statistic***	***Statistic***	***Statistic***	***Statistic***	***Statistic***	***Statistic***	***Std.Error***	***Statistic***	***Std. Error***
dev_05_grid	45375	4227	-2295	1932	-2	.00	114.048	13006.937	-.212	.011	75.865	.023

Table 4.16: *Descriptive Statistics (deviation model 05 - grid). Source: own illustration.*

deviations grid

Inhabitants		***Frequency***	***Percent***	***Valid Percent***	***Cumulative Percent***
Valid	-2295 to -1500	13	.0	.0	.0
	-1499 to -200	750	1.7	1.7	1.7
	-199 to -100	887	2.0	2.0	3.6
	-99 to -20	4456	9.8	9.8	13.5
	-19 to 20	34435	75.9	75.9	89.3
	21 to 100	2871	6.3	6.3	95.7
	101 to 200	1023	2.3	2.3	97.9
	201 to 1500	927	2.0	2.0	100.0
	1501 to 1932	13	.0	.0	100.0
	Total	45375	100.0	100.0	

Table 4.17: *Frequency Table (deviation model 05 - grid). All cases selected. Source: own illustration.*

5 Comparison with Reference Data

Independently from the modelling approach or the number of auxiliary data used, the calculation of the daytime population of a city remains an estimation. The real population of a building is unknown, making accuracy evaluations of the modelling results difficult. However, it is possible to compare the results with the daytime population of another study or agency. For this thesis this was done by analysing the daytime population 2013 of Statistik Austria, the Austrian statistical office and comparing it to the population grid of model 01. Since Statistik Austria disaggregated the daytime population to 100x100m raster cells a comparison on the level of buildings was not possible. Instead of the digital building model Statistik Austria used address points and non-public register data to calculate the daytime population per raster cell. Nonetheless, by aggregating the inhabitants per building of model 01 - 05 to the same raster a comparison between the modelling results of this thesis and Statistik Austria is possible. The deviations which are discussed later in this chapter (see Figure 5.3) are calculated by subtracting the population of model 01 from the one of Statistik Austria per raster cell. Therefore, positive deviations (red) indicate Statistik Austria has more population allocated whereas negative deviations (blue) mean the opposite. Model 01 was chosen for the comparison because it is presumably the most accurate.
Table 5.1 shows the descriptive statistics of the population grids of Statistik Austia and model 01 (inhabitants per raster cell). The sum of the daytime populations of the grids differs from the one of Statistik Austria by 73,454 inhabitants (3.6%). This could be explained by the fact that Statistik Austria uses other registers and also includes areas just outside of Vienna. Also noticeable in Table 5.1 is the different range with Statistik Austria having a much higher maximum of 36.399 whereas the maximum of model 01 is only 3,347. However, this is only an outlier since the mean values are very similar. Table 5.2 and Table 5.3 also show that the frequency of inhabitants per cell (summarised in 5 groups) of both grids are quite similar. Apparently, most of the values (88.1% Statistik Austria, 88.8% model 01) lie within 1 and 100 inhabitants.

Descriptive Statistics

	N	*Range*	*Min*	*Max*	*Sum*	*Mean*	*Std. Dev*	*Var*	*Skewness*		*Kurtosis*	
	Statistic	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Std.Error*	*Statistic*	*Std. Error*
pop_daytime_ stat_austria	45375	36399	0	36399	2040004	44.96	253.792	64410.133	75.960	.011	9731.997	.023
pop_daytime_ model01	45375	3347	0	3347	1966550	43.34	146.078	21338.808	8.675	.011	104.962	.023

Table 5.1: *Descriptive Statistics (daytime population grid Statistik Austria & model 01). Source: own illustration, Statistik Austria.*

population grid (daytime) - Statistik Austria

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	1 - 100	39989	88.1	88.1	88.1
	101 - 300	4065	9.0	9.0	97.1
	301 - 700	948	2.1	2.1	99.2
	701 - 1500	295	.7	.7	99.8
	1501 - 36399	78	.2	.2	100.0
	Total	45375	100.0	100.0	

Table 5.2: *Frequency Table (daytime population grid Statistik Austria). Source: Statistik Austria, own illustration.*

population grid (daytime) - model 01

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	1 - 100	40301	88.8	88.8	88.8
	101 - 300	3758	8.3	8.3	97.1
	301 - 700	860	1.9	1.9	99.0
	701 - 1500	329	.7	.7	99.7
	1501 - 3347	127	.3	.3	100.0
	Total	45375	100.0	100.0	

Table 5.3: *Frequency Table (daytime population grid model 01). Source: own illustration.*

A look at Figure 5.1 shows the population grids of Statistik Austria and model 01. It can be seen that the distribution of the former is much more even with high amounts of population spread punctually across Vienna. On the other hand, the population distribution of model 01 shows local concentrations which make structures visible like shopping streets, the high density of commercial buildings in the 1st district, Vienna´s General Hospital or the university complex in Spittelau. However, in general the distributions are similar with higher concentrations of population near the city centre and lower values on its periphery. The peaks of high population of the Statistik Austria grid can be seen more clearly in Figure 5.2 which is a 3-dimensional representation of the grids. The two highest values (36,399 and 15,492) of the Statistik Austria grid could not be completely depicted in Figure 5.2 *(**top**)* due to the length of their bars but are shown in the smaller inset image.

Descriptive Statistics

	N	*Range*	*Min*	*Max*	*Sum*	*Mean*	*Std. Dev*	*Var*	*Skewness*		*Kurtosis*	
	Statistic	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Statistic*	*Std.Error*	*Statistic*	*Std. Error*
deviation	45375	38025	-3336	34688	73454	1.62	247.303	61158.642	68.462	.011	8877.484	.023

Table 5.4: *Descriptive Statistics (deviations Statistik Austria & model 01). Source: own illustration, Statistik Austria.*

deviations grid

Inhabitants		*Frequency*	*Percent*	*Valid Percent*	*Cumulative Percent*
Valid	-3336 to -1500	55	.1	.1	.1
	-1499 to -200	982	2.2	2.2	2.3
	-199 to -100	866	1.9	1.9	4.2
	-99 to -20	2258	5.0	5.0	9.2
	-19 to 20	35350	77.9	77.9	87.1
	21 to 100	3913	8.6	8.6	95.7
	101 to 200	1140	2.5	2.5	98.2
	201 to 1500	767	1.7	1.7	99.9
	1501 to 34688	44	.1	.1	100.0
	Total	45375	100.0	100.0	

Table 5.5: *Frequency Table (deviations Statistik Austria & model 01). Source: own illustration.*

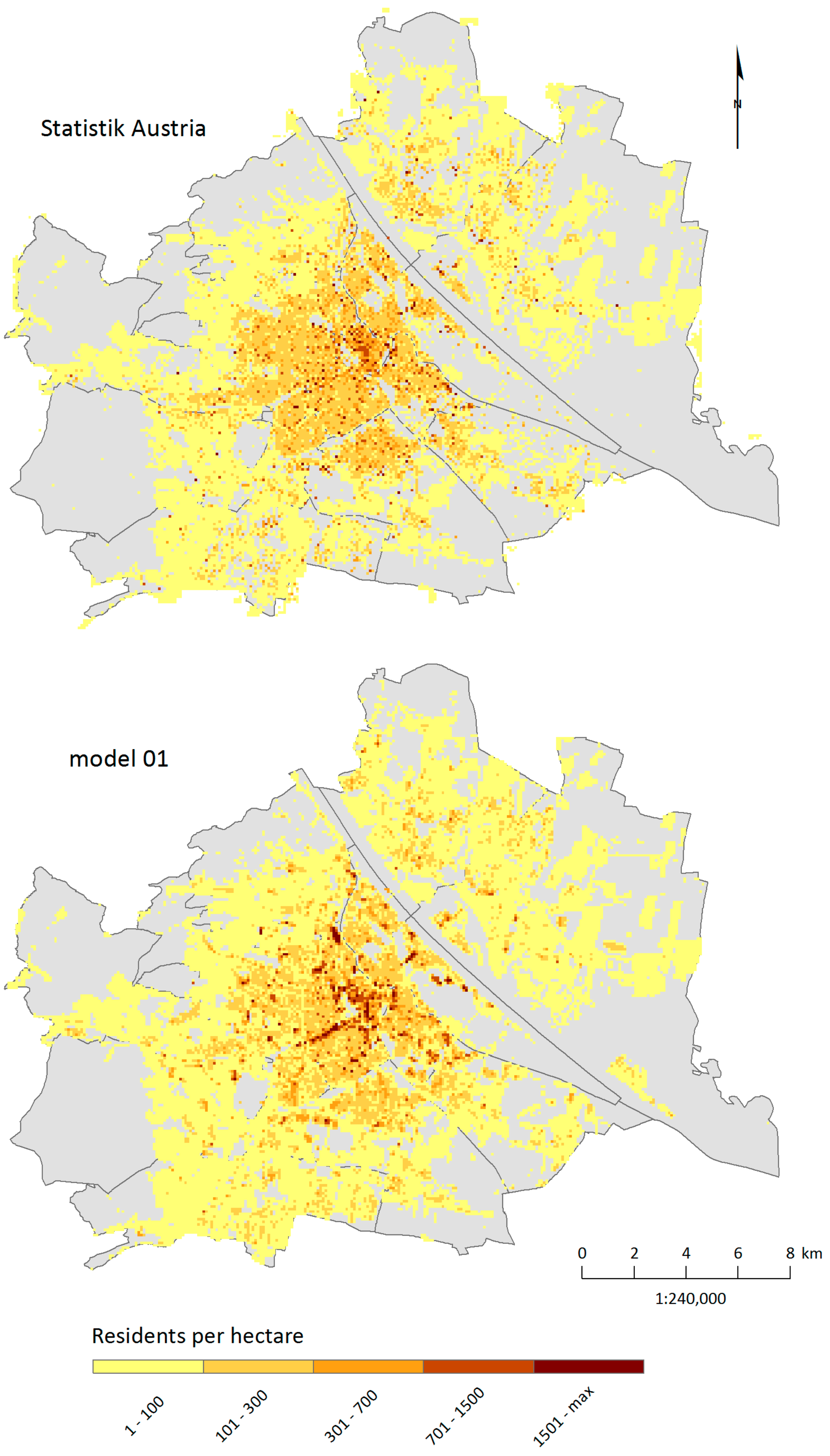

Figure 5.1: *Daytime Population Grid 2013 - Statistik Austria & Model 01. Source: own illustration, Statistika Austria. Coordinate System: ETRS 1989 LAEA.*

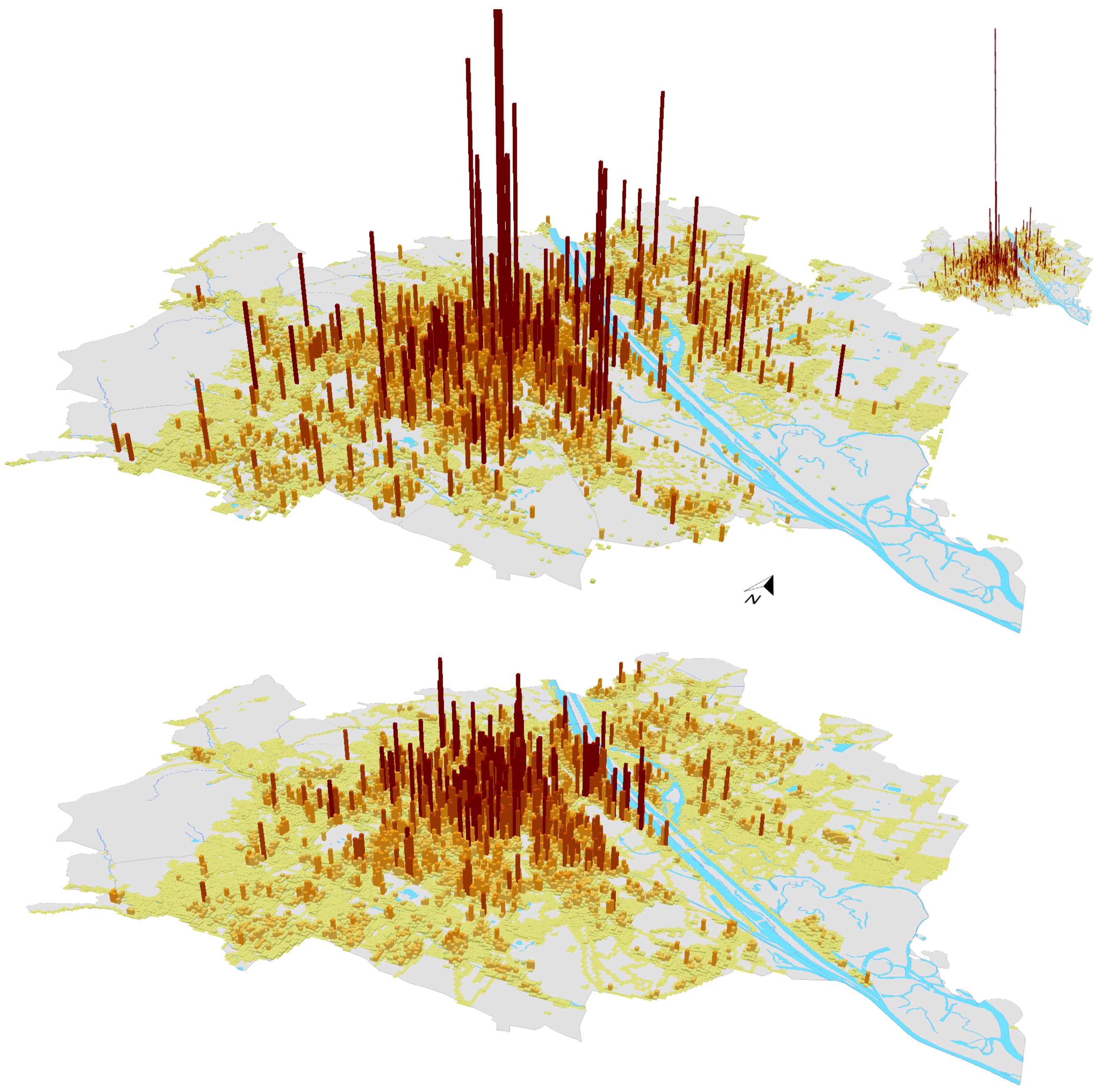

Figure 5.2: *Daytime Population Grid 2013 (3D) - Statistik Austria (**top**) & Model 01 (**bottom**). Source: own illustration, Statistik Austria.*

In order to make assertions of the similarity of the grids the deviations were calculated. Table 5.4 shows that the mean deviation is only 1.62 with the minimal deviation being -3,336 and the maximum one being 34,688 resulting in a positive skewness of 68.5. The sum of the deviations is not null but 73,454 meaning that Statistik Austria distributed more inhabitants as mentioned before. The frequency table also shows that 77.9% of the raster cells have a deviation of +/- 20 inhabitants (Table 5.5) which is even more than model 05 with 75.9% (Table 4.17).

Most of the negative deviations occur in commercial buildings (especially in the inner city and the shopping street Mariahilfer Straße), the industrial areas in the south (23rd district) and northeast (21st, 22nd district) as well as social infrastructure buildings like hospitals or universities (see Figure 5.3).

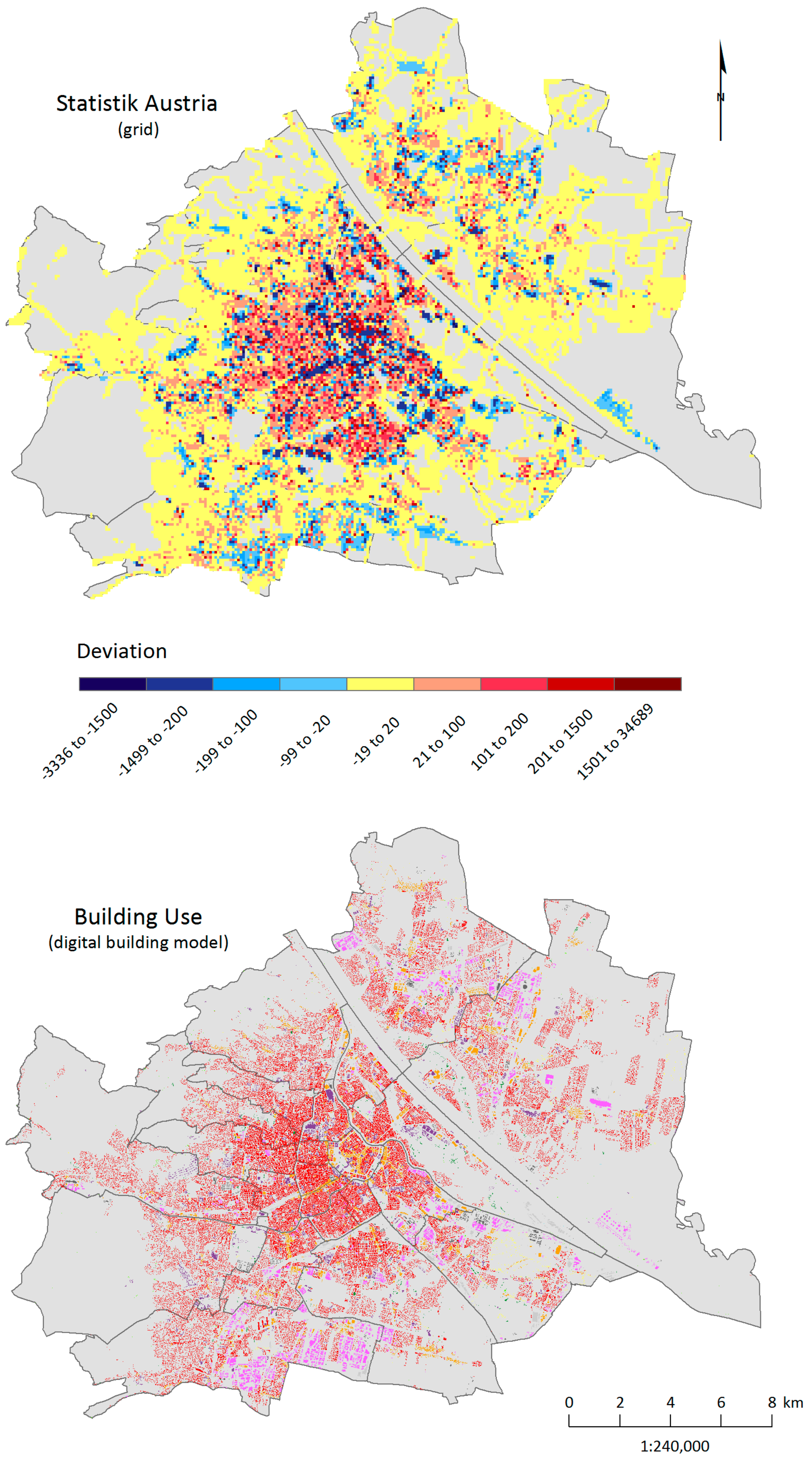

Figure 5.3: *Statistik Austria deviation grid & building use. Source: own illustration, Statistik Austria. Legend see Figure 2.5 on page 13. Coordinate System: ETRS 1989 LAEA.*

This means that model 01 tends to distribute more population to commercial and social infrastructure buildings. On the other hand, positive deviations where Statistik Austria allocates more population than model 01 occur mostly in residential buildings near the city centre where the average building heights are bigger. Small deviations of +/- 20 inhabitants occur most commonly at residential buildings in the periphery where the daytime population of both grids is generally low.

Figure 5.4 depicts an example of the General Hospital of Vienna (AKH) which shows why sometimes high deviations occur between the two grids. The upper image shows the population grid of Statistik Austria and as overlay the digital building model. Since Statistik Austria did not use a digital building model for the distribution it can be seen that most of the cells which intersect with the hospital in the centre do not contain any inhabitants with the exception of two cells with a very large population count of 9,205 and 5,141. They probably intersect with an address point and therefore got the full daytime population of the hospital. The lower image shows the population grid of model 01. Here it can be seen that the population of the hospital is distributed evenly over the main as well as the smaller buildings. However, when adding up the population count of the 33 cells which intersect with the hospital it shows that Statistik Austria distributed 16,114 inhabitants while model 01 distributed in the same area a similar amount of 18,562 inhabitants. The highest positive deviations occur in the main buildings of the University of Vienna (34,688) as well as the Technical University of Vienna (13,826) with the reason being similar to the situation described above where Statistik Austria probably allocates all of the students (which is a high number for both universities) in the raster cell where the address point of the university intersects. In model 01 the same population is distributed among all university institutes resulting in a high deviation compared to Statistik Austria.

Figure 5.5 shows the opposite where model 01 distributes more population than Statistik Austria. This occurs at the hospital Semmelweis Frauenklinik which occupies four social infrastructure buildings along the central vertical axis of the image. Statistik Austria distributes to the southernmost building 406 inhabitants while model 01 distributes among all four buildings a total of 7,623 persons. This number consists of 215 patients, 14 trade school pupils and 7,431 employees making up most of the population. The reason for this high number is that this hospital is not included in the company register and therefore the remaining working population (which is the difference between the total working population and through registers already distributed working population) is distributed among its buildings weighted by their volumes. As already seen in Figure 4.19 on page 63 the working population tends to be overestimated without a company register. Another example of a high negative deviation can be found at one of the buildings of the Technical University of Vienna (Wiedner Hauptstraße 8-10) where one raster cell has a deviation of -2743. The reason for this lies in the above described case where Statistik Austria locates all of the students of a university at the main building where the address point is located leaving the neighbouring cells empty or with very little inhabitants from e.g. intersecting residential buildings. The described negative deviation occurs in such a cell.

Despite the fact that the disaggregation approaches of Statistik Austria and model 01 differ, it can be said that both population grids show a similar distribution of inhabitants making the results of model 01 plausible. Some differences occur at large buildings where Statistik Austria distributes its whole population or the whole population of a company/university to one single raster cell whereas the distribution of model 01 is more even. However, similar errors from the register of companies are possible. On other occasions the population numbers of Statistik Austria might be more accurate because of the non-public registers which were used as input data.

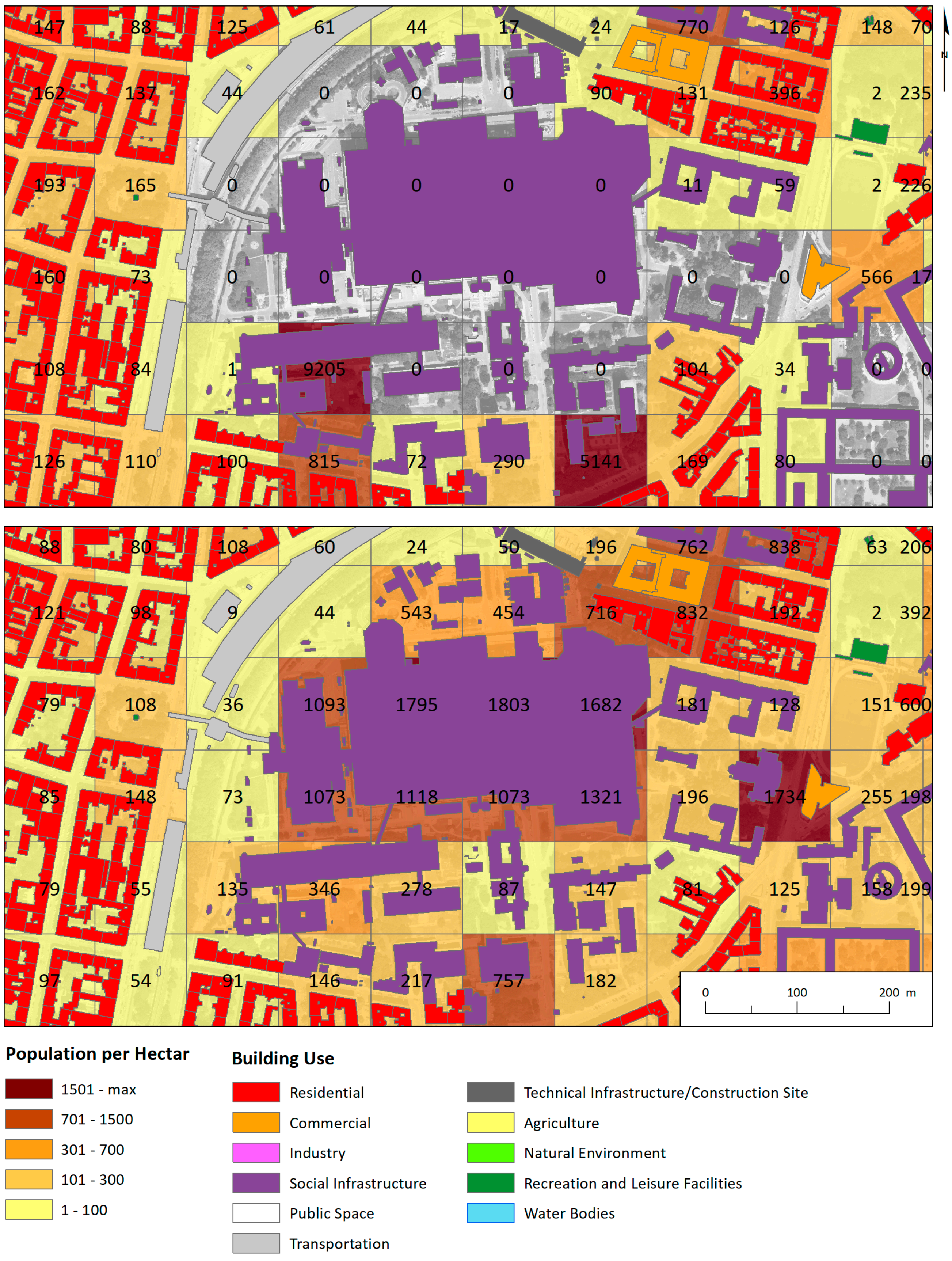

Figure 5.4: *Example 1: General Hospital of Vienna (AKH).* ***Top****: Statistik Austria population grid,* ***bottom****: model 01 population grid. Source: own illustration, Statistik Austria. Coordinate System: ETRS 1989 LAEA.*

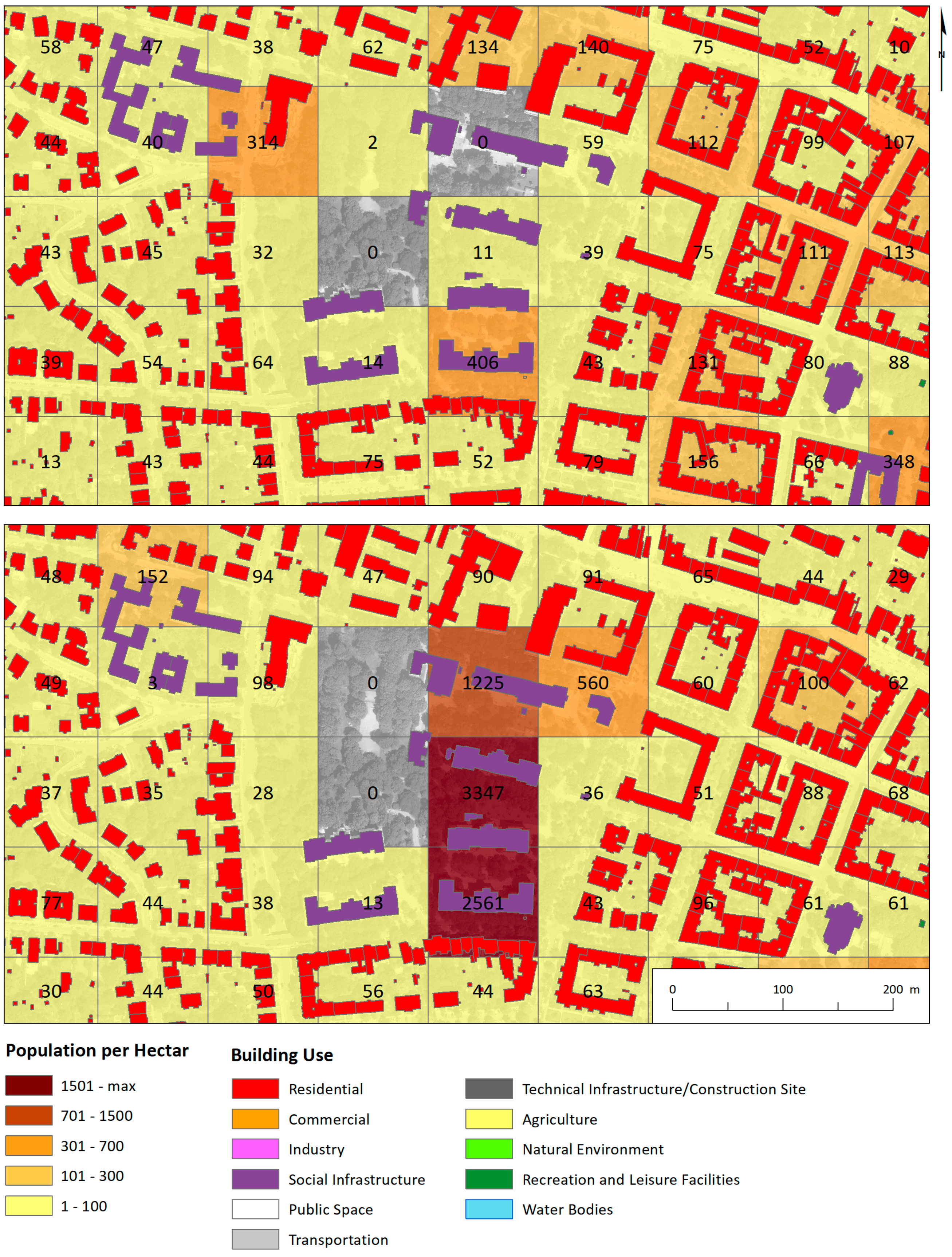

Figure 5.5: *Example 2: Semmelweis Frauenklinik.* ***Top***: *Statistik Austria population grid,* ***bottom***: *model 01 population grid. Source: own illustration, Statistik Austria. Coordinate System: ETRS 1989 LAEA.*

Conclusions

This thesis developed an approach how the daytime population of a city can be modelled based on areal weighting. Through the implementation in a Python script the analysis could be automated, facilitating the population calculation for a different time or city or the adjustment of individual parameters. Since the used method is data driven, five different models were developed in order to examine where and how high the population distribution deviate when some datasets are not available such as the building heights, company register or no ancillary data at all. Model 01 uses all available datasets and distributes the population weighted by volume. Model 02 also uses all datasets but uses only the building footprints as weighting factor. Model 03 is the same as model 01 but excludes the company register. Modell 04 does not use any auxiliary data at all and distributes the population to the buildings weighted by volume. Finally, model 05 is the same as model 04 but does not use the digital building model, distributing its population to the land use layer weighted by area.
The deviations were calculated by subtracting the population per building (or raster cell) of model 01 (which uses all datasets) from those of model 02-05 or the population grid of Statistik Austria. It can be seen that in case of Vienna the lowest deviations occur when the company register is excluded. The reason for this is that the population number which is distributed through the register makes up only 5.9% of the total daytime population. The exclusion of the company register leads particularly to overestimations in commercial and industrial buildings and to underestimations in residential buildings. The second lowest deviations can be found in model 02 where the disaggregation was weighted only by the building footprints rather than their volume. Here the over- and underestimations are less dependent on the building use and instead correlated to the building volume where the inhabitants of buildings with higher volumes tend to be rather underestimated which is expected. Next comes model 04 which does not use ancillary data but distributes the population to the buildings weighted by volume resulting in overestimations in commercial, industrial and social infrastructure buildings and underestimations in residential buildings. Similar but higher deviations can be found in model 05 which also does not use ancillary data but distributes the population to the land use 2012 layer.
It is rather difficult to make assumptions about the accuracy of the modelling results because the exact number of inhabitants per building during the day is unknown. However, what can be done is a comparison of the analysis results with other studies or institutions. In case of this thesis the results were compared to the population grid of Vienna (2013) provided by Statistik Austria. Since Statistik Austria calculated the daytime population on the level of 100m raster cells, the results of this study had to be aggregated to the same raster making also the comparison between the five models easier. Figure 1 shows the distribution of the deviations per grid cell for all models and the Statistik Austria grid. When looking at the total absolute deviations which are calculated by summarizing both the positive and negative deviations over all raster cells it can be seen that the highest deviations can be found in the grid of Statistik Austria. At the same time, its absolute deviations are closest to those of model 05 which also did not use building data. However, Statistik Austria used a different method for its population distribution based on address points and non-public data which manifests in the different distribution of the deviations. While especially models 04 and 05 tend to underestimate the inhabitants of residential buildings in comparison with model 01, Statistik Austria tends to overestimates them while underestimating commercial, industrial and social infrastructure buildings. Inaccuracies in the Statistik Austria grid especially occur where large buildings or institutions like universities with several buildings get all their inhabitants allocated to only one grid cell based on the corresponding address point.

In case of model 01 inaccuracies are expected through the company register because it includes companies which list all of their domestic and/or international employees in their headquarters. To acknowledge this, only entries of smaller companies with less than 1,000 employees were included for the modelling. Nevertheless, this threshold was estimated and probably does not exclude all cases. Other inaccuracies can lie in the fact that not all datasets are available from the same year with many of the geospatial datasets being from the year 2017. Furthermore, the building use is derived from the land use 2012 layer of Vienna which can lead to buildings getting the wrong use class. This can result in warehouses or other normally empty buildings getting inhabitants allocated to. Especially large warehouses can get large amounts of inhabitants. Therefore, the accuracy could be improved by only using datasets of the same year, improving the use classification of the digital building model and using a company register which lists the actual employees per company and address. Problematic though could be the availability or the amount of work needed to get the desired information.

Although the developed approach of modelling the daytime population does include many datasets and population groups there is still way for improvement. For example, data on tourists including hotel bookings and visitor numbers of attractions could be considered which would be in particular helpful for disaster management and risk assessment. Also, busy public spaces such as shopping streets could be modelled including the consideration of police officers and other population groups working in public space. Ultimately, much of the challenge lies in the acquisition and preparation of the datasets which are necessary for accurate modelling results.

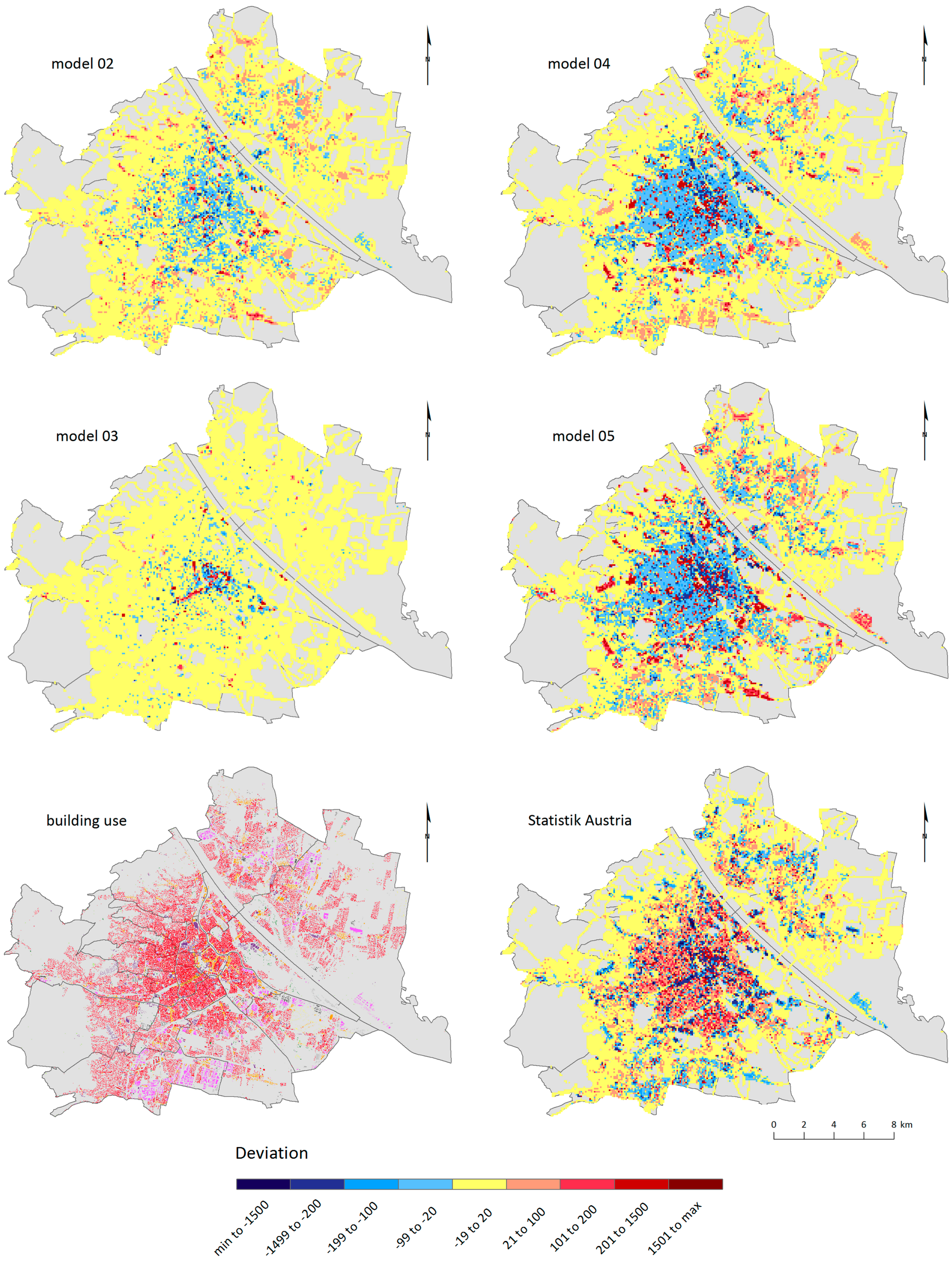

***Figure 1:** Deviations (grid) of all models & Statistik Austria. Source: own illustration, Statistik Austria. Legend of building use see Figure 2.5 on page 13. Coordinate System: ETRS 1989 LAEA.*

Literature

Amt der Oö. Landesregierung (ed.) (2014): Landeskorrespondenz Medieninfo. Unterstützungsangebote für pflegende Angehörige in Oberösterreich – Amt der Oö. Landesregierung, Linz

Aubrecht C. et al. (2009): Integrating earth observation and GIScience for high resolution spatial and functional modeling of urban land use. – In: Computers, Environment and Urban Systems 33, 15-25

Deville P. et al. (2014): Dynamic population mapping using mobile phone data. – Proceedings of the National Academy of Sciences; doi: 10.1073/pnas.1408439111

European Commission (1993): CORINE land cover. Technical guide – Brussels and Luxembourg

European Environment Agency (2002): Towards an urban atlas. Assessment of spatial data on 25 European cities and urban areas. – Copenhagen (= Environmental issue report 30)

Freire S., Florczyk A. and Ferri S. (2015): Modeling Day- and Nighttime Population Exposure at High Resolution: Application to Volcanic Risk Assessment in Campi Flegrei. – Proceedings of the ISCRAM 2015 Conference, Kristiansand

MA18 Stadtentwicklung Wien (ed.) (2005): STEP 05. Stadtentwicklung Wien 2005 – MA 18, Vienna

MA18 Stadtentwicklung Wien (ed.) (2014): STEP 2025. Stadtentwicklungsplan Wien – MA 18, Vienna

Martin D. et al. (2014): Developing a flexible framework for spatiotemporal population modeling. – Annals of the Association of American Geographers; doi: 10.1080/00045608.2015.1022089

Medizinische Universität Wien (ed.) (2015): Zukunft gestalten. Jahresbericht 2015 – Medizinische Universität Wien, Vienna

Mennis J. and Hultgren T. (2006): Intelligent Dasymetric Mapping and Its Application to Areal Interpolation. – In: Cartography and Geographic Information Science 33 (3), 179-194

Statistisches Bundesamt (ed.) (2013): Pflegestatistik 2013. Pflege im Rahmen der Pflegeversicherung Deutschlandergebnisse – Statistisches Bundesamt, Wiesbaden

Steinnocher K. et al. (2014): Modellierung raum-zeitlicher Bevölkerungsverteilungsmuster im Katastrophenmanagementkontext. – In: Proceedings REAL CORP 2014 Tagungsband, 909-913

Steinnocher K., Köstl M. and Weichselbaum J. (2015): Grid-based population and land take trend indicators. New approaches introduced by the geoland2 Core Information Service for Spatial Planning. – Vienna; https://www.researchgate.net/publication/266285682_Grid-based_population_and_land_take_trend_indicators_-_New_approaches_introduced_by_the_geoland2_Core_Information_Service_for_Spatial_Planning (30.03.2017)

Taubenböck H. et al. (2008): Multi-scale Assessment of Population Distribution utilizing Remotely Sensed Data. The Case Study Padang, West Sumatra, Indonesia – Wessling (= International Conference on Tsunami Warning); https://www.researchgate.net/publication/259902076_Multi-scale_assessment_of_population_distribution_utilizing_remotely_sensed_data_-_The_case_study_Padang_West_Sumatra_Indonesia (30.03.2017)

Taubenböck H. and Dech S. (ed.) (2010): Fernerkundung im urbanen Raum. Erdbeobachtung auf dem Weg zur Planungspraxis – WBG, Darmstadt

Wiener Linien (ed.) (2013): 2013. Zahlen Daten Fakten – Wiener Linien, Vienna

WKO (ed.) (2006): Ausbilden zahlt sich aus!. Moderne Lehrberufe im Dienstleistungsbereich – Wirtschaftskammer Österreich, Vienna

Internet

City of Vienna (n.d. a): Realnutzungskartierung - Flächennutzung im Stadtgebiet; https://www.wien.gv.at/stadtentwicklung/grundlagen/stadtforschung/siedlungsentwicklung/realnutzungskartierung/index.html (10.05.2017)

City of Vienna (n.d. b): Baukörpermodell - Produktinformation; https://www.wien.gv.at/stadtentwicklung/stadtvermessung/geodaten/bkm/produkt.html (11.05.2017)

City of Vienna (n.d. c): Flächen-Mehrzweckkarte; https://www.wien.gv.at/stadtentwicklung/stadtvermessung/geodaten/fmzk/index.html (11.05.2017)

Copernicus (n.d.): Imperviousness; http://land.copernicus.eu/pan-european/high-resolution-layers/imperviousness (25.06.2017)

Esri (n.d.): What is ArcPy?; http://pro.arcgis.com/en/pro-app/arcpy/get-started/what-is-arcpy-.htm (22.05.2017)

United Nations (2014): World's population increasingly urban with more than half living in urban areas; http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html (02.09.2016)

Table of Figures

Table of Tables

The author assures:

- that he wrote the master´s thesis independently, did not use other sources than quoted and refrained from using other unapproved aids.
- that all passages, both literal and analogous, that come from published and non-published sources are marked.
- that the topic of this Master´s thesis was never passed on as an auditing work to a supervisor, neither domestically nor abroad.
- that this thesis is identical to the one assessed by the supervisor.

19.09.2017
Date

[illegible]
Signature

Appendix

```
#-----------------------------------------------------------------------------------------------#
#---------------------------00-12 Disaggregate Data & Aggregate to Raster-----------------------#
#-----------------------------------------------------------------------------------------------#

import datetime
start1 = datetime.datetime.now()
print 'start run: %0.19s\n' % (start1)

import arcpy, os, sys
arcpy.env.overwriteOutput = True

# Set analysis_no
# 01...etrs89_output_alldata
# 02...etrs89_output_alldata_noheight
# 03...etrs89_output_alldata_nofirmenreg
# 04...etrs89_output_no_auxiliary_data
# 05...etrs89_output_no_aux_no_bkm
analysis_no = "_02" ###

# Define Workspace
ws = r"C:\Users\Aquarius\Desktop\project"
gdb = ws + os.sep + "project01.gdb"
wsInput = gdb + os.sep + "etrs89"
wsOutput2 = gdb + os.sep + "etrs89_analysis_output"
wsTemp = gdb + os.sep + "etrs89_temp"
if analysis_no == "_01":
    wsOutput = gdb + os.sep + "etrs89_output_alldata"
elif analysis_no == "_02":
    wsOutput = gdb + os.sep + "etrs89_output_alldata_noheight"
elif analysis_no == "_03":
    wsOutput = gdb + os.sep + "etrs89_output_alldata_nofirmenreg"
elif analysis_no == "_04":
    wsOutput = gdb + os.sep + "etrs89_output_no_auxiliary_data"
elif analysis_no == "_05":
    wsOutput = gdb + os.sep + "etrs89_output_no_aux_no_bkm"

# Define w_factor
if analysis_no == "_01" or analysis_no == "_03" or analysis_no == "_04":
    w_factor = "[Shape_Area]*[height]"
elif analysis_no == "_02" or analysis_no == "_05":
    w_factor = "[Shape_Area]"

# Define Output Names
output_bkm = "bkm_output"
output_bkm2 = "bkm_output2"
output_bkm3 = "bkm_output3"
output_zbez = "zbez_output"
output_bez = "bez_etrs"
output_realnut = "realnut_output"
```

```
output_realnut2 = "realnut_output2"
output_buffer_wrlinien = "public_transportation_buffer"
if analysis_no == "_01":
    output_grid = "grid_aggregated_alldata"
elif analysis_no == "_02":
    output_grid = "grid_aggregated_alldata_noheight"
elif analysis_no == "_03":
    output_grid = "grid_aggregated_alldata_nofirmenreg"
elif analysis_no == "_04":
    output_grid = "grid_aggregated_no_auxiliary_data"
elif analysis_no == "_05":
    output_grid = "grid_aggregated_no_auxiliary_data_realnut"

# Define Inputs
input_zbez = "zbez_etrs"
input_bkm = "bkm"
input_realnut = "realnut2012"
input_grid = "grid_100"
point_univ = "univOGD"
point_vhs = "vhsOGD"
point_adr = "address_point"
point_kiga = "ogd_kiga"
point_schools = "ogd_schule_wien"
point_ret_home = "ogd_wohnpflege"
network_transp = "public_transportation"
table_pendlersaldo = "pendlersaldo"
table_hosp_empl = "besch_kh"
table_univ_vhs_empl = "besch_univ_vhs"
table_firmenreg = "branche_besch_korr"
table_wrlinien_empl = "besch_wr_linien"
table_total_empl = "besch_gesamt"
table_unempl = "unemployed"
table_schueler = "schueler"
table_students_univ = "studenten"
table_patients = "patients"

#------------------------------------------------------------#

# Input Fields
# zbez_etrs
OID = "OBJECTID"
ID_zbez = "ZBEZ_NR"
ID_bez = "BEZ"
POP_total = "pop_total"
POP_0_4 = "age0_4"
POP_5_9 = "age5_9"
POP_10_14 = "age10_14"
POP_15_19 = "age15_19"
POP_20_29 = "age20_29"
POP_30_64 = "age30_64"
POP_60_64 = "age60_64"
```

```
POP_65_69 = “age65_69”
POP_70_74 = “age70_74”
POP_75_79 = “age75_79”
POP_80_84 = “age80_84”
POP_85_89 = “age85_89”
POP_90plus = “age90_plus”

# bkm
w_factor_bkm = “w_factor”
ID_bkm = “code_bkm”
height_bkm = “height”
vol_bkm = “volume”
length_bkm = “Shape_Length”
area_bkm = “Shape_Area”
ID_hospital = “kh_id”
NAME_hospital = “hospital”
USE_bkm = “NUTZUNG__1”

living = “Wohn- u. Mischnutzung (Schwerpunkt Wohnen)”
trade = “Geschäfts,- Kern- und Mischnutzung (Schwerpunkt betriebl. Tätigkeit)”
industry = “Industrie- und Gewerbenutzung”
social = “soziale Infrastruktur”

# realnut2012
OID_realnut = “OBJECTID”
length_realnut = “Shape_Length”
area_realnut = “Shape_Area”
w_factor_realnut = “w_factor”
USE_realnut = “NUTZUNG__1”

living2 = “Wohn- u. Mischnutzung (Schwerpunkt Wohnen)”
trade2 = “Geschäfts,- Kern- und Mischnutzung (Schwerpunkt betriebl. Tätigkeit)”
industry2 = “Industrie- und Gewerbenutzung”
social2 = “soziale Infrastruktur”

# grid_100
ID_grid = “grid_id”

# univOGD
NAME_univ1 = “univ1”
NAME_univ2 = “univ2”

# vhsOGD
NAME_vhs = “vhs”

# address_point
join_adr = “joi_adr”

# ogd_kiga
TYPE_kiga = “type_kiga”
BEZ_kiga = “BEZ_kiga”
```

```
# ogd_schule_wien
BEZ_schule = "BEZ_schule"
TYP_GEN = "TYP_GEN"

NAME_ogd_pflichtschule = "Pflichtschule"
NAME_ogd_berufsschule = "Berufsschule"
NAME_ogd_andere_hoehere = "andere hoehere"
NAME_ogd_ahs = "AHS"

# ogd_wohnpflege
BEZ_pflege = "BEZ_pflege"
TYPE_pflege = "care_home"

# pendlersaldo
pendlersaldo_stud = "pendlersaldo_schueler_studenten"
pendlersaldo_working = "pendlersaldo_arbeitendeBev"
pendlersaldo_total = "pendlersaldo"

# besch_kh
POP_15_64_hospital = "p15_64_w_hosp"
ID_hospital2 = "kh_id"
ID_count = "id"

# besch_univ_vhs
POP_15_64_univ_vhs = "p15_64_w_univ_vhs"
ID_count2 = "id"
NAME_univ = "NAME"
TYPE_univ = "TYP"

type_univ = "University"

# branche_besch_korr
POP_15_64_firmenreg = "p15_64_firmenreg"
join_adr_firmenreg = "joi_adr"
NAME_firmenreg = "Firmenname"

# besch_wr_linien
POP_15_64_wrlinien = "p15_64_wr_linien"

# besch_gesamt
POP_15_64_besch_ges = "age15_64_besch_gesamt"

# unemployed
POP_15_64_unemployed = "p15_64_unemployed"

# schueler
NAME_pflichtschule = "Pflichtschule"
NAME_berufsschule = "Berufsschule"
NAME_andere_hoehere = "andere_hoehere"
NAME_ahs = "AHS"
```

```
# studenten
NAME_univ_stud = "NAME"
POP_20_29_studenten = "p20_29_studenten"

# patients
POP_patients = "patients"
ID_count3 = "id"
ID_hospital_pat = "kh_id"

#-----------------------------------------------------------#

# Outputs
output_name_pendlersaldo = output_bez + analysis_no
output_name_firmenreg = output_bkm2 + analysis_no
output_name_wrlinien = output_buffer_wrlinien + analysis_no
output_name_kiga = output_zbez + analysis_no
output_name_total_pop = output_bkm3 + analysis_no
output_name_aggregate = output_grid

# Define table_temp
table_temp = gdb + os.sep + "table_temp"


###################################################################################################
##################################      MODEL 04 & 05      ######################################
###################################################################################################


if analysis_no == "_04" or analysis_no == "_05":
    if analysis_no == "_04":
        bkm_output = output_bkm + analysis_no
        bkm_output2 = output_bkm2 + analysis_no
        zbez_output = output_zbez + analysis_no
        USE_bkm_realnut = USE_bkm
        w_factor_bkm_realnut = w_factor_bkm
    elif analysis_no == "_05":
        bkm_output = output_realnut + analysis_no
        bkm_output2 = output_realnut2 + analysis_no
        zbez_output = output_zbez + analysis_no
        USE_bkm_realnut = USE_realnut
        w_factor_bkm_realnut = w_factor_realnut

    #-------------------------------------------------------------------------------------------#
    #--Dissolve ZBEZ to get aggregated pop (per BEZ) for age10_14, age15_19, age20_29, age30_64---#
    #-----------------------Add pendlersaldo to population (per BEZ)------------------------------#
    #-------------------------------------------------------------------------------------------#


    print "---------------------------------------------------------------------------"
    print "---------------Add pendlersaldo to population per BEZ----------------------"
    print "--------------------------------------------------------------------------\n"

    input_fc_pendlersaldo = wsInput + os.sep + input_zbez # zbez_etrs
    input_table_pendlersaldo = gdb + os.sep + table_pendlersaldo # pendlersaldo
    output_fc_pendlersaldo = wsOutput + os.sep + output_name_pendlersaldo # bez_etrs

    print "Input data:"
    print input_zbez + " (polygon_fc)"
    print table_pendlersaldo + " (table)"

    #------------------------------------------ANALYSIS------------------------------------------#
    print "\n----------------------------------ANALYSIS--------------------------------\n"

    # Dissolve ZBEZ to get age10_14, age15_19, age20_29, age30_64 per BEZ
    print "Process: Dissolve ZBEZ to get BEZ"
    arcpy.Dissolve_management(input_fc_pendlersaldo, output_fc_pendlersaldo, ID_bez,
        """{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;
            {} SUM;{} SUM; {} SUM; {} SUM; {} SUM; {} SUM; {} SUM""".format(POP_total, POP_0_4,
POP_5_9, POP_10_14, POP_15_19,
                       POP_20_29, POP_30_64, POP_60_64, POP_65_69, POP_70_74, POP_75_79, POP_80_84,
POP_85_89, POP_90plus),
            "MULTI_PART", "DISSOLVE_LINES")

    # Join pendlersaldo_arbeitendeBev & pendlersaldo_schueler_studenten from table pendlersaldo
    print "Process: Join pendlersaldo to BEZ"
    arcpy.JoinField_management(output_fc_pendlersaldo, ID_bez, input_table_pendlersaldo, ID_bez,
                              "{};{}".format(pendlersaldo_stud, pendlersaldo_working))

    # Calculate pendlersaldo
    print "Process: Calculate pendlersaldo"
    arcpy.AddField_management(output_fc_pendlersaldo, "pendlersaldo", "DOUBLE")
    arcpy.CalculateField_management(output_fc_pendlersaldo,"pendlersaldo","[{}]+[{}]".format(pend-
lersaldo_stud, pendlersaldo_working))

    # Calculate field age10_64_plus_pendler
    print "Process: Calculate field age10_64_plus_pendler"
    arcpy.AddField_management(output_fc_pendlersaldo, "age10_64_plus_pendler", "DOUBLE")
    arcpy.CalculateField_management(output_fc_pendlersaldo,"age10_64_plus_pendler",
            "[pendlersaldo]+[SUM_{}]+[SUM_{}]+[SUM_{}]+[SUM_{}]".format(POP_10_14, POP_15_19,
POP_20_29, POP_30_64))

    #--------------------------------------------------------------------------------------------#
    print "\n--------------------------------------------------------------------------"

    print "\nOutput Data:"
    print output_name_pendlersaldo + " (polygon_fc)"

    print "\nOutput Fields in " + output_name_pendlersaldo + ":"
    print "age10_64_plus_pendler"

    end = datetime.datetime.now()
```

```
print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start1)

print “\n###########################################################################\n\n”

#----------------------------------------------------------------------------------------------#
#------------------------------------Distribute age0_4-----------------------------------------#
#----------------------------------------------------------------------------------------------#
print “---------------------------------------------------------------------------”
print “-------------------------Distribute age0_4---------------------------------”
print “---------------------------------------------------------------------------\n”

import datetime
start = datetime.datetime.now()
print ‘start run: %0.19s\n’ % (start)

if analysis_no == “_04”:
    source_fc = wsInput + os.sep + input_bkm
elif analysis_no == “_05”:
    source_fc = wsInput + os.sep + input_realnut

arcpy.CalculateField_management(source_fc, “w_factor”, w_factor)
arcpy.CopyFeatures_management(source_fc, wsOutput + os.sep + bkm_output)

source_fc_kiga = wsInput + os.sep + input_zbez
arcpy.CopyFeatures_management(source_fc_kiga, wsOutput + os.sep + zbez_output)

input_fc_kiga = wsOutput + os.sep + bkm_output # bkm_output
input_fc2_kiga = wsOutput + os.sep + zbez_output # zbez_output

print “Input data:”
print bkm_output + “ (polygon_fc)”
print zbez_output + “ (polygon_fc)”

#------------------------------------------ANALYSIS--------------------------------------------#
#---------------------Distribute Population (age0_4_athome) to bkm----------------------------#
print “\n---------------------------------ANALYSIS---------------------------------”
print “------------------Distribute Population (age0_4) to bkm--------------------\n”

# Process: Identity (get buildings per ZBEZ)
print “Identity (bkm, zbez)”
output_identity = wsTemp + os.sep + “output_identity_zbez”
arcpy.Identity_analysis(input_fc_kiga, input_fc2_kiga, output_identity)
arcpy.AddField_management(output_identity, “w_factor2”, “DOUBLE”)
arcpy.CalculateField_management(output_identity, “w_factor2”, w_factor)

# Add Fields
arcpy.AddField_management(output_identity, “w_factor_total2”, “DOUBLE”)
arcpy.AddField_management(output_identity, “w_factor_percent2”, “DOUBLE”)
arcpy.AddField_management(output_identity, “p0_4_w_athome”, “DOUBLE”)
```

```
    print "\nDistribute p0_4_w_athome to Wohngebäude per ZBEZ:"
    SC = arcpy.SearchCursor(input_fc2_kiga)
    field_nr = OID
    field_id = ID_zbez
    field_pop = POP_0_4
    for row in SC:
        f_nr = row.getValue(field_nr)
        f_id = row.getValue(field_id)
        pop_athome = row.getValue(field_pop)

        # Selecting Features (Wohngebäude) per ZBEZ
        arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

        if analysis_no == "_04":
            use = living
        elif analysis_no == "_05":
            use = living2
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                                "{} = {} and {} = '{}'".format(ID_zbez, f_id, USE_bkm_
realnut, use))

        # Process: Summary Statistics (getting total w_factor of all buildings per ZBEZ)
        arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = "SUM_w_factor2"
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print "Calculating Fields for ZBEZ " + str(f_id) + "(" + str(f_nr) + ")"
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_
factor_total2]")
        arcpy.CalculateField_management("input_fc_f_layer", "p0_4_w_athome", "{} * [w_factor_per-
cent2]".format(pop_athome))

    print "\nDissolve output_identity"
    output_diss_identity = wsTemp + os.sep + "output_diss_identity2"
    arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p0_4_w_athome SUM",
"MULTI_PART", "DISSOLVE_LINES")

    print "Join p0_4_w_athome to bkm_output2"
    arcpy.DeleteField_management(input_fc_kiga, ["SUM_p0_4_w_athome"])
    arcpy.JoinField_management(input_fc_kiga, ID_bkm, output_diss_identity, ID_bkm, "SUM_p0_4_w_
athome")

    # Check distributed population
    arcpy.Statistics_analysis(input_fc2_kiga, table_temp, "{} SUM".format(POP_0_4))
```

```
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_{}”.format(POP_0_4)
for row in SC:
    pop_zbez = row.getValue(field_sum)

arcpy.Statistics_analysis(input_fc_kiga, table_temp, “SUM_p0_4_w_athome SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_SUM_p0_4_w_athome”
for row in SC:
    pop = row.getValue(field_sum)

print “\nTotal Population (age0_4) = “ + str(int(round(pop_zbez,0)))
print “Distributed Population (p0_4_w_athome) = “ + str(int(round(pop,0)))

#---------------------------------------------------------------------------------------------#
print “\n--------------------------------------------------------------------------”

print “\nOutput Data:”
print bkm_output + “ (polygon_fc)”

print “\nOutput Fields in “ + bkm_output + “:”
print “SUM_p0_4_w_athome”


end = datetime.datetime.now()

print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start)

print “\n##########################################################################\n\n”

#---------------------------------------------------------------------------------------------#
#---------------------------------Distribute age5_9 to bkm------------------------------------#
#---------------------------------------------------------------------------------------------#
print “--------------------------------------------------------------------------”
print “-------------------------Distribute age5_9 to bkm-------------------------”
print “--------------------------------------------------------------------------\n”

import datetime
start = datetime.datetime.now()
print ‘start run: %0.19s\n’ % (start)

input_fc_schueler = wsOutput + os.sep + bkm_output # bkm_output
input_fc2_schueler = wsOutput + os.sep + output_name_pendlersaldo # bez_etrs

print “Input data:”
print bkm_output + “ (polygon_fc)”
print output_name_pendlersaldo + “ (polygon_fc)”

#---------------------------------------ANALYSIS---------------------------------------------#
#-------------------------------Distribute age5_9 to bkm-------------------------------------#
```

```
    print "\n----------------------------------ANALYSIS---------------------------------"
    print "--------------------------Distribute age5_9 to bkm------------------------\n"

    # Process: Identity (get buildings per BEZ)
    print "Identity (bkm, bez)"
    output_identity = wsTemp + os.sep + "output_identity2"
    arcpy.Identity_analysis(input_fc_schueler, input_fc2_schueler, output_identity)
    arcpy.AddField_management(output_identity, "w_factor2", "DOUBLE")
    arcpy.CalculateField_management(output_identity, "w_factor2", w_factor)

    # Add Fields
    arcpy.AddField_management(output_identity, "w_factor_total2", "DOUBLE")
    arcpy.AddField_management(output_identity, "w_factor_percent2", "DOUBLE")
    arcpy.AddField_management(output_identity, "p5_9", "DOUBLE")

    print "\nDistribute age5_9 to bkm:"
    SC = arcpy.SearchCursor(input_fc2_schueler)
    field_id = ID_bez
    field_pop = "SUM_{}".format(POP_5_9)
    for row in SC:
        f_id = row.getValue(field_id)
        pop_5_9 = row.getValue(field_pop)

        # Selecting Features (soziale Infrastruktur) per BEZ
        arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

        if analysis_no == "_04":
            use = social
        elif analysis_no == "_05":
            use = social2
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                                "{} = {} and {} = '{}'".format(ID_bez, f_id, USE_
bkm_realnut, use))

        # Process: Summary Statistics (getting total w_factor of all buildings per BEZ)
        arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = "SUM_w_factor2"
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print "Calculating Fields for BEZ " + str(f_id)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_
factor_total2]")
        arcpy.CalculateField_management("input_fc_f_layer", "p5_9", "{} * [w_factor_percent2]".
format(pop_5_9))
```

```
    print "\nDissolve output_identity"
    output_diss_identity = wsTemp + os.sep + "output_diss_identity"
    arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p5_9 SUM", "MULTI_
PART", "DISSOLVE_LINES")

    print "Join p5_9 to bkm_output2"
    arcpy.DeleteField_management(input_fc_schueler, ["SUM_p5_9"])
    arcpy.JoinField_management(input_fc_schueler, ID_bkm, output_diss_identity, ID_bkm, "SUM_p5_9")

    # Check distributed population
    arcpy.Statistics_analysis(input_fc2_schueler, table_temp, "SUM_{} SUM".format(POP_5_9))
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_{}".format(POP_5_9)
    for row in SC:
        p5_9 = row.getValue(field_sum)

    arcpy.Statistics_analysis(input_fc_schueler, table_temp, "SUM_p5_9 SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_p5_9"
    for row in SC:
        p5_9_distr = row.getValue(field_sum)

    print "\nTotal Population (5_9) = " + str(int(round(p5_9,0)))
    print "Distributed Population (5_9) = " + str(int(round(p5_9_distr,0)))

    #----------------------------------------------------------------------------------------------#
    print "\n--------------------------------------------------------------------------"

    print "\nOutput Data:"
    print bkm_output + " (polygon_fc)"

    print "\nOutput Fields in " + bkm_output + ":"
    print "SUM_p5_9"


    end = datetime.datetime.now()

    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s\n' % (end-start)

    print "\n##########################################################################\n\n"

    #----------------------------------------------------------------------------------------------#
    #----------------------------------Distribute age10_64 to bkm----------------------------------#
    #----------------------------------------------------------------------------------------------#
    print "-----------------------------------------------------------------------"
    print "------------------------Distribute age10_64 to bkm------------------------"
    print "-----------------------------------------------------------------------\n"

    import datetime
    start = datetime.datetime.now()
    print 'start run: %0.19s\n' % (start)

    input_fc_10_64 = wsOutput + os.sep + bkm_output # bkm_output
    input_fc2_10_64 = wsOutput + os.sep + output_name_pendlersaldo # bez_etrs

    print "Input data:"
    print bkm_output + " (polygon_fc)"
    print output_name_pendlersaldo + " (polygon_fc)"

    #-------------------------------------------ANALYSIS----------------------------------------#
    #-----------------------------------Distribute age10_64 to bkm-------------------------------#
    print "\n---------------------------------ANALYSIS---------------------------------"
    print "-------------------------Distribute age10_64 to bkm-----------------------\n"

    # Add Field
    arcpy.AddField_management(output_identity, "p10_64_plus_pendler", "DOUBLE")

    print "Distribute age10_64 to bkm per BEZ:"
    SC = arcpy.SearchCursor(input_fc2_10_64)
    field_id = ID_bez
    field_pop = "age10_64_plus_pendler"
    for row in SC:
        f_id = row.getValue(field_id)
        pop_10_64 = row.getValue(field_pop)

        # Selecting Features (soziale Infrastruktur) per BEZ
        arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

        if analysis_no == "_04":
            use1 = trade
            use2 = industry
            use3 = social
        elif analysis_no == "_05":
            use1 = trade2
            use2 = industry2
            use3 = social2
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                                "{} = {} and {} in ('{}', '{}', '{}')".format(ID_
bez, f_id, USE_bkm_realnut, use1, use2, use3))

        # Process: Summary Statistics (getting total w_factor of all buildings per BEZ)
        arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = "SUM_w_factor2"
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print "Calculating Fields for BEZ " + str(f_id)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_
factor_total2]")
        arcpy.CalculateField_management("input_fc_f_layer", "p10_64_plus_pendler", "{} * [w_fac-
tor_percent2]".format(pop_10_64))

    print "\nDissolve output_identity"
    output_diss_identity = wsTemp + os.sep + "output_diss_identity2"
    arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p10_64_plus_pendler
SUM", "MULTI_PART", "DISSOLVE_LINES")

    print "Join p10_64_plus_pendler to bkm_output"
    arcpy.DeleteField_management(input_fc_10_64, ["SUM_p10_64_plus_pendler"])
    arcpy.JoinField_management(input_fc_10_64, ID_bkm, output_diss_identity, ID_bkm, "SUM_p10_64_
plus_pendler")

    # Check distributed population
    arcpy.Statistics_analysis(input_fc2_10_64, table_temp, "age10_64_plus_pendler SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_age10_64_plus_pendler"
    for row in SC:
        p10_64 = row.getValue(field_sum)

    arcpy.Statistics_analysis(input_fc_10_64, table_temp, "SUM_p10_64_plus_pendler SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_p10_64_plus_pendler"
    for row in SC:
        p10_64_distr = row.getValue(field_sum)

    print "\nTotal Population (10_64) = " + str(int(round(p10_64,0)))
    print "Distributed Population (10_64) = " + str(int(round(p10_64_distr,0)))

    #-----------------------------------------------------------------------------------------------#
    print "\n--------------------------------------------------------------------------"

    print "\nOutput Data:"
    print bkm_output + " (polygon_fc)"

    print "\nOutput Fields in " + bkm_output + ":"
    print "SUM_p10_64_plus_pendler"


    end = datetime.datetime.now()

    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s\n' % (end-start)

    print "\n################################################################################\n\n"

    #-----------------------------------------------------------------------------------------------#
    #-----------------------------------Distribute age65_plus to bkm------------------------------#
    #-----------------------------------------------------------------------------------------------#
    print "---------------------------------------------------------------------------"
    print "-----------------------Distribute age65_plus to bkm-----------------------"
    print "---------------------------------------------------------------------------\n"

    import datetime
    start = datetime.datetime.now()
    print 'start run: %0.19s\n' % (start)

    input_fc_65_plus = wsOutput + os.sep + bkm_output # bkm_output
    input_fc2_65_plus = wsOutput + os.sep + zbez_output # zbez_output

    print "Input data:"
    print bkm_output + " (polygon_fc)"
    print zbez_output + " (polygon_fc)"

    #-----------------------------------------ANALYSIS-----------------------------------------#
    #-------------------------Distribute age65_plus to bkm per ZBEZ----------------------------#
    print "\n---------------------------------ANALYSIS---------------------------------"
    print "------------------Distribute age65_plus to bkm per ZBEZ-------------------\n"

    # Calculate age_65_plus in zbez_output
    arcpy.DeleteField_management(input_fc2_65_plus, ["age_65_plus"])
    arcpy.AddField_management(input_fc2_65_plus, "age_65_plus", "DOUBLE")
    arcpy.CalculateField_management(input_fc2_65_plus, "age_65_plus",
"[{}]+[{}]+[{}]+[{}]+[{}]+[{}]".format(POP_65_69, POP_70_74, POP_75_79, POP_80_84, POP_85_89,
POP_90plus))

    output_identity = wsTemp + os.sep + "output_identity_zbez"

    # Add Fields
    arcpy.AddField_management(output_identity, "p65_plus_ret_athome", "DOUBLE")

    print "\nDistribute p65_plus_ret_athome to Wohngebäude per ZBEZ:"
    SC = arcpy.SearchCursor(input_fc2_65_plus)
    field_nr = OID
    field_id = ID_zbez
    field_pop = "age_65_plus"
    for row in SC:
        f_nr = row.getValue(field_nr)
        f_id = row.getValue(field_id)
        pop_athome = row.getValue(field_pop)

        # Selecting Features (Wohngebäude) per ZBEZ
        arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

        if analysis_no == "_04":
            use = living
        elif analysis_no == "_05":
            use = living2
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                                    "{} = {} and {} = '{}'".format(ID_zbez, f_id, USE_
bkm_realnut, use))

        # Process: Summary Statistics (getting total w_factor of all buildings per ZBEZ)
        arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = "SUM_w_factor2"
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print "Calculating Fields for ZBEZ " + str(f_id) + "(" + str(f_nr) + ")"
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_
factor_total2]")
        arcpy.CalculateField_management("input_fc_f_layer", "p65_plus_ret_athome", "{} * [w_factor_
percent2]".format(pop_athome))

    print "\nDissolve output_identity"
    output_diss_identity = wsTemp + os.sep + "output_diss_identity_new"
    arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p65_plus_ret_athome
SUM", "MULTI_PART", "DISSOLVE_LINES")

    print "Join p65_plus_ret_athome to bkm_output"
    arcpy.DeleteField_management(input_fc_65_plus, ["SUM_p65_plus_ret_athome"])
    arcpy.JoinField_management(input_fc_65_plus, ID_bkm, output_diss_identity, ID_bkm, "SUM_p65_
plus_ret_athome")

    # Check distributed population
    arcpy.Statistics_analysis(input_fc2_65_plus, table_temp, "age_65_plus SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_age_65_plus"
    for row in SC:
        pop = row.getValue(field_sum)

    arcpy.Statistics_analysis(input_fc_65_plus, table_temp, "SUM_p65_plus_ret_athome SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_p65_plus_ret_athome"
    for row in SC:
        pop_distr = row.getValue(field_sum)

    print "\nTotal Population (age_65_plus) = " + str(int(round(pop,0)))
    print "Distributed Population (age_65_plus) = " + str(int(round(pop_distr,0)))

    #-----------------------------------------------------------------------------------------------#
    print "\n---------------------------------------------------------------------------"

    print "\nOutput Data:"
    print bkm_output + " (polygon_fc)"
    print zbez_output + “ (polygon_fc)”

    print “\nOutput Fields in “ + bkm_output + “:”
    print “SUM_p65_plus_ret_athome”

    print “\nOutput Fields in “ + zbez_output + “:”
    print “age_65_plus”


    end = datetime.datetime.now()

    print ‘\nfinished run: %0.19s\n’ % (end),
    print ‘runtime: %0.7s\n’ % (end-start)

    print “\n##########################################################################\n\n”

    #---------------------------------------------------------------------------------------------#
    #----------------------Create bkm_output2 and calculate total population----------------------#
    #---------------------------------------------------------------------------------------------#
    print “--------------------------------------------------------------------------”
    print “------------Create bkm_output2 and calculate total population-------------”
    print “--------------------------------------------------------------------------\n”

    import datetime
    start = datetime.datetime.now()
    print ‘start run: %0.19s\n’ % (start)

    if analysis_no == “_04”:
        input_fc_total_pop = wsInput + os.sep + input_bkm # bkm
    elif analysis_no == “_05”:
        input_fc_total_pop = wsInput + os.sep + input_realnut
    input_fc2_total_pop = wsOutput + os.sep + bkm_output # bkm_output
    input_fc3_total_pop = wsOutput + os.sep + zbez_output # zbez_output
    output_fc_total_pop = wsOutput2 + os.sep + bkm_output2 # bkm_output2
    input_table_pendlersaldo = gdb + os.sep + table_pendlersaldo # pendlersaldo

    print “Input data:”
    if analysis_no == “_04”:
        print input_bkm + “ (polygon_fc)”
    elif analysis_no == “_05”:
        print input_realnut + “ (polygon_fc)”
    print bkm_output + “ (polygon_fc)”
    print zbez_output + “ (polygon_fc)”
    print table_pendlersaldo + “ (table)”


#----------------------------------------------ANALYSIS-------------------------------------------#
    #-------------------Create bkm_output2 and calculate total population-------------------------#
    print “\n---------------------------------ANALYSIS---------------------------------”
    print “------------Create bkm_output2 and calculate total population-------------\n”
    # Create bkm_output2
    print "Create bkm_output2"
    arcpy.CopyFeatures_management(input_fc_total_pop, output_fc_total_pop)

    # Join Fields
    f1 = "SUM_p0_4_w_athome"
    f2 = "SUM_p5_9"
    f3 = "SUM_p10_64_plus_pendler"
    f4 = "SUM_p65_plus_ret_athome"

    print "Join Fields"
    arcpy.JoinField_management(output_fc_total_pop, "code_bkm", input_fc2_total_pop, ID_bkm,
        ["SUM_p0_4_w_athome", "SUM_p5_9", "SUM_p10_64_plus_pendler", "SUM_p65_plus_ret_athome"])

    # Replace <Null> with 0
    print "Replace <Null> with 0"
    expression = "myCalc(!SUM_p0_4_w_athome!)"
    codeblock = """def myCalc (population):
        if population == None:
            return 0
        else:
            return (population)"""
    arcpy.CalculateField_management(output_fc_total_pop, "SUM_p0_4_w_athome", expression, "PY-
THON_9.3", codeblock)

    expression = "myCalc(!SUM_p5_9!)"
    codeblock = """def myCalc (population):
        if population == None:
            return 0
        else:
            return (population)"""
    arcpy.CalculateField_management(output_fc_total_pop, "SUM_p5_9", expression, "PYTHON_9.3",
codeblock)

    expression = "myCalc(!SUM_p10_64_plus_pendler!)"
    codeblock = """def myCalc (population):
        if population == None:
            return 0
        else:
            return (population)"""
    arcpy.CalculateField_management(output_fc_total_pop, "SUM_p10_64_plus_pendler", expression,
"PYTHON_9.3", codeblock)

    expression = "myCalc(!SUM_p65_plus_ret_athome!)"
    codeblock = """def myCalc (population):
        if population == None:
            return 0
        else:
            return (population)"""
    arcpy.CalculateField_management(output_fc_total_pop, "SUM_p65_plus_ret_athome", expression,
"PYTHON_9.3", codeblock)
```

```
    # Calculate total population
    print "Calculate total population"
    arcpy.AddField_management(output_fc_total_pop, "population_vienna", "DOUBLE")
    arcpy.CalculateField_management(output_fc_total_pop, "population_vienna",
        "[{}]+[{}]+[{}]+[{}]".format(f1, f2, f3, f4))

    # Check calculated population
    arcpy.Statistics_analysis(input_fc3_total_pop, table_temp, "{} SUM".format(POP_total))
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_{}".format(POP_total)
    for row in SC:
        pop = row.getValue(field_sum)

    arcpy.Statistics_analysis(input_table_pendlersaldo, table_temp, "{} SUM".format(pendlersal-
do_total))
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_{}".format(pendlersaldo_total)
    for row in SC:
        pendlersaldo = row.getValue(field_sum)

    population = pop + pendlersaldo

    arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "population_vienna SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_population_vienna"
    for row in SC:
        pop_distr = row.getValue(field_sum)

    print "\nTotal Population = " + str(int(round(population,0)))
    print "Distributed Population = " + str(int(round(pop_distr,0)))

    #-----------------------------------------------------------------------------------------------#
    print "\n--------------------------------------------------------------------------"

    print "\nOutput Data:"
    print bkm_output2 + " (polygon_fc)"

    print "\nOutput Fields in " + bkm_output2 + ":"
    print "population_vienna"


    end = datetime.datetime.now()

    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s\n' % (end-start)

    print "\n##########################################################################\n\n"

        #-------------------------------------------------------------------------------------------#
    #------------------------------Aggregate population to raster---------------------------------#
```

```
#------------------------------------------------------------------------------------------#
    print "--------------------------------------------------------------------------"
    print "-----------------------Aggregate population to raster----------------------"
    print "--------------------------------------------------------------------------\n"

    import datetime
    start = datetime.datetime.now()
    print 'start run: %0.19s\n' % (start)

    input_fc = wsOutput2 + os.sep + bkm_output2 # bkm_output2
    input_fc2 = wsInput + os.sep + input_grid # grid
    output_fc = wsOutput2 + os.sep + output_name_aggregate # grid_aggregated

    print "Input data:"
    print bkm_output2 + " (polygon_fc)"
    print input_grid + " (polygon_fc)"


    #----------------------------------------ANALYSIS 1-------------------------------------------#
    #---------------------------Aggregate population to raster-------------------------------------#
    print "\n--------------------------------ANALYSIS 1--------------------------------"
    print "------------------------Aggregate population to raster--------------------\n"

    # Make Feature Layer
    arcpy.MakeFeatureLayer_management(input_fc, "f_layer", "", "",
        """SUM_p0_4_w_athome SUM_p0_4_w_athome VISIBLE RATIO;
            SUM_p5_9 SUM_p5_9 VISIBLE RATIO;
            SUM_p10_64_plus_pendler SUM_p10_64_plus_pendler VISIBLE RATIO;
           SUM_p65_plus_ret_athome SUM_p65_plus_ret_athome VISIBLE RATIO;
           population_vienna population_vienna VISIBLE RATIO""")

    # Process: Identity
    print "Identity (buildings per grid cell)"
    output_identity = wsTemp + os.sep + "identity_grid_aggr"
    arcpy.Identity_analysis("f_layer", input_fc2, output_identity, "ALL", "", "NO_RELATIONSHIPS")

    # Process: Spatial Join
    print "Spatial Join (bkm from identity to grid)"
    FID_bkm_output2 = "FID_" + output_bkm2 + analysis_no
    arcpy.SpatialJoin_analysis(input_fc2, output_identity, output_fc, "JOIN_ONE_TO_ONE", "KEEP_
ALL",
        """Shape_Length "Shape_Length" false true true 8 Double 0 0 ,First,#,{0},Shape_
Length,-1,-1;
            Shape_Area "Shape_Area" false true true 8 Double 0 0 ,First,#,{0},Shape_Area,-1,-1;
            {3} "{3}" false true true 8 Double 0 0 ,First,#,{0},{3},-1,-1;
            {1} "{1}" true true false 4 Long 0 0 ,First,#,{1},{2},-1,-1;
            {4} "{4}" true true false 8 Double 0 0 ,First,#,{2},{4},-1,-1;
            {5} "{5}" true true false 8 Double 0 0 ,First,#,{2},{5},-1,-1;
            {6} "{6}" true true false 8 Double 0 0 ,Mean,#,{2},{6},-1,-1;
            {7} "{7}" true true false 8 Double 0 0 ,First,#,{2},{7},-1,-1;
            {8} "{8}" true true false 150 Text 0 0 ,First,#,{2},{8},-1,-1;
            {9} "{9}" true true false 50 Text 0 0 ,First,#,{2},{9},-1,-1;
            {10} "{10}" true true false 70 Text 0 0 ,First,#,{2},{10},-1,-1;
            SUM_p0_4_w_athome "SUM_p0_4_w_athome" true true false 8 Double 0 0 ,Sum,#,{2},SUM_
p0_4_w_athome,-1,-1;
            SUM_p5_9 "SUM_p5_9" true true false 8 Double 0 0 ,Sum,#,{2},SUM_p5_9,-1,-1;
            SUM_p10_64_plus_pendler "SUM_p10_64_plus_pendler" true true false 4 Long 0 0
,Sum,#,{2},SUM_p10_64_plus_pendler,-1,-1;
            SUM_p65_plus_ret_athome "SUM_p65_plus_ret_athome" false true true 8 Double 0 0
,Sum,#,{2},SUM_p65_plus_ret_athome,-1,-1;
            population_vienna "population_vienna" false true true 8 Double 0 0 ,Sum,#,{2},popula-
tion_vienna,-1,-1"""
            .format(input_fc2, FID_bkm_output2, output_identity, ID_grid, w_factor_bkm_realnut,
ID_bkm, height_bkm, vol_bkm, ID_hospital, NAME_hospital, USE_bkm),
                       "CONTAINS", "#", "#")

    # Check if Spatial Join worked - if not (code_bkm = 0) print error message and exit
    var = arcpy.da.SearchCursor(output_fc, ("{}".format(ID_bkm),)).next()[0]
    if var == 0:
        print "ERROR IN FIELD MAPS"
        sys.exit()

    # Check calculated population
    arcpy.Statistics_analysis(output_fc, table_temp, "population_vienna SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_population_vienna"
    for row in SC:
        pop_distr = row.getValue(field_sum)

    print "\nTotal Population = " + str(int(round(population,0)))
    print "Distributed Population = " + str(int(round(pop_distr,0)))

    #-----------------------------------------------------------------------------------------------#
    print "\n-------------------------------------------------------------------------------"

    print "\nOutput Data:"
    print output_name_aggregate + " (polygon_fc)"

    print "\nOutput Fields in " + output_name_aggregate + ":"
    print "SUM_p0_4_w_athome"
    print "SUM_p5_9"
    print "SUM_p10_64_plus_pendler"
    print "SUM_p65_plus_ret_athome"
    print "population_vienna"

    end = datetime.datetime.now()

    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s' % (end-start)
    print '\ntotal runtime: %0.7s\n' % (end-start1)

    sys.exit()
```

```
####################################################################################################
###################################      MODEL 01 - 03      #######################################
####################################################################################################

print “\n###########################################################################\n\n”


#--------------------------------------------------------------------------------------------------#
#---Dissolve ZBEZ to get aggregated pop (per BEZ) for age10_14, age15_19, age20_29, age30_64------#
#--------------------------------Add pendlersaldo to population (per BEZ)-------------------------#
#--------------------------------------------------------------------------------------------------#

print “--------------------------------------------------------------------------”
print “---------------Add pendlersaldo to population per BEZ---------------------”
print “--------------------------------------------------------------------------\n”

input_fc_pendlersaldo = wsInput + os.sep + input_zbez # zbez_etrs
input_table_pendlersaldo = gdb + os.sep + table_pendlersaldo # pendlersaldo
output_fc_pendlersaldo = wsOutput + os.sep + output_name_pendlersaldo # bez_etrs

print “Input data:”
print input_zbez + “ (polygon_fc)”
print table_pendlersaldo + “ (table)”

#----------------------------------------ANALYSIS-------------------------------------------------#
print “\n----------------------------------ANALYSIS--------------------------------\n”

# Dissolve ZBEZ to get age10_14, age15_19, age20_29, age30_64 per BEZ
print “Process: Dissolve ZBEZ to get BEZ”
arcpy.Dissolve_management(input_fc_pendlersaldo, output_fc_pendlersaldo, ID_bez,
    “””{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;{} SUM;
        {} SUM;{} SUM; {} SUM; {} SUM; {} SUM; {} SUM; {} SUM”””.format(POP_total, POP_0_4,
POP_5_9, POP_10_14, POP_15_19,
               POP_20_29, POP_30_64, POP_60_64, POP_65_69, POP_70_74, POP_75_79, POP_80_84,
POP_85_89, POP_90plus),
        “MULTI_PART”, “DISSOLVE_LINES”)

# Calculate age10_29 & age15_64 for BEZ
print “Process: Calculate fields age10_29 & age15_64”
arcpy.AddField_management(output_fc_pendlersaldo, “age10_29”, “DOUBLE”)
arcpy.AddField_management(output_fc_pendlersaldo, “age15_64”, “DOUBLE”)
arcpy.CalculateField_management(output_fc_pendlersaldo, “age10_29”, “[SUM_{}]+[SUM_{}]+[SUM_{}]”.
format(POP_10_14, POP_15_19, POP_20_29))
arcpy.CalculateField_management(output_fc_pendlersaldo, “age15_64”, “[SUM_{}]+[SUM_{}]+[SUM_{}]”.
format(POP_15_19, POP_20_29, POP_30_64))

# Join pendlersaldo_arbeitendeBev & pendlersaldo_schueler_studenten from table pendlersaldo
print “Process: Join pendlersaldo to BEZ”
arcpy.JoinField_management(output_fc_pendlersaldo, ID_bez, input_table_pendlersaldo, ID_bez,
    “{};{}”.format(pendlersaldo_stud, pendlersaldo_working))
```

```
# Calculate age10_29_plus_pendler & age15_64_plus_pendler for BEZ
print "Process: Calculate fields age10_29 & age15_64 (plus pendler)"
arcpy.AddField_management(output_fc_pendlersaldo, "age10_29_plus_pendler", "DOUBLE")
arcpy.AddField_management(output_fc_pendlersaldo, "age15_64_plus_pendler", "DOUBLE")
arcpy.CalculateField_management(output_fc_pendlersaldo, "age10_29_plus_pendler", "[age10_29]+[{}]".
format(pendlersaldo_stud))
arcpy.CalculateField_management(output_fc_pendlersaldo, "age15_64_plus_pendler", "[age15_64]+[{}]".
format(pendlersaldo_working))

#-----------------------------------------------------------------------------------------------#
print "\n-------------------------------------------------------------------------"

print "\nOutput Data:"
print output_name_pendlersaldo + " (polygon_fc)"

print "\nOutput Fields in " + output_name_pendlersaldo + ":"
print "age10_29_plus_pendler"
print "age15_64_plus_pendler"

end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start1)


#################################################################################################
print "\n#########################################################################\n\n"


#-----------------------------------------------------------------------------------------------#
#Distribute Working Population (Hospitals [1/2], University, VHS) to bkm and subtract from bez
(age15_64_plus_pendler)#
#-----------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "-----------------------------------------------------------------------"
print "----Distribute Working Pop (hosp, univ, vhs) to bkm & subtract from BEZ---"
print "-----------------------------------------------------------------------\n"

source_fc_h_univ_vhs = wsInput + os.sep + input_bkm
arcpy.CalculateField_management(source_fc_h_univ_vhs, w_factor_bkm, w_factor)
arcpy.CopyFeatures_management(source_fc_h_univ_vhs, wsOutput + os.sep + output_bkm + analysis_no)

input_fc_h_univ_vhs = wsOutput + os.sep + output_bkm + analysis_no # bkm_output
input_fc2_h_univ_vhs = wsOutput + os.sep + output_bez + analysis_no # bez_etrs
input_fc3_h_univ_vhs = wsInput + os.sep + point_univ # univOGD
input_fc4_h_univ_vhs = wsInput + os.sep + point_vhs # vhsOGD
input_table1_h_univ_vhs = gdb + os.sep + table_hosp_empl # besch_kh
```

```
input_table2_h_univ_vhs = gdb + os.sep + table_univ_vhs_empl # besch_univ_vhs

print “Input data:”
print input_bkm + “ (polygon_fc)”
print output_bez + analysis_no + “ (polygon_fc)”
print point_univ + “ (point_fc)”
print point_vhs + “ (point_fc)”
print table_hosp_empl + “ (table)”
print table_univ_vhs_empl + “ (table)”

#--------------------------------------------ANALYSIS 1--------------------------------------------#
#---Distribute Working Population (age15_64) to Group of Buildings (hospitals [1/2], univ, vhs)---#
print “\n--------------------------------ANALYSIS 1--------------------------------”
print “--------------Distribute working pop (age15_64) to buildings--------------\n”

# Add Fields
arcpy.AddField_management(input_fc_h_univ_vhs, “w_factor_total”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “w_factor_percent”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “p15_64_w_hosp”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “p15_64_w_univ”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “p15_64_w_vhs”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “p15_64_w_univ_vhs”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “p15_64_w_hosp_univ_vhs”, “DOUBLE”)
arcpy.AddField_management(input_fc_h_univ_vhs, “percent”, “DOUBLE”)

# Calculate p15_64_w_hosp (1/2)
SC = arcpy.SearchCursor(input_table1_h_univ_vhs)
field_id = ID_count
field_name =  ID_hospital2
field_pop = POP_15_64_hospital
for row in SC:
    f_id = row.getValue(field_id)
    name = row.getValue(field_name)
    pop_besch = row.getValue(field_pop)

    arcpy.MakeFeatureLayer_management (input_fc_h_univ_vhs, “input_fc_f_layer”)
    arcpy.SelectLayerByAttribute_management(“input_fc_f_layer”, “NEW_SELECTION”, “{} = ‘{}’”.for-
mat(ID_hospital, name))

    # Process: Summary Statistics
    arcpy.Statistics_analysis(“input_fc_f_layer”, table_temp, “{} SUM”.format(w_factor_bkm))

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = “SUM_{}”.format(w_factor_bkm)
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields
    print “Calculating Fields for “ + str(name) + “ (“ + str(int(round(f_id,0))) + “)”
    arcpy.CalculateField_management(“input_fc_f_layer”, “w_factor_total”, value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent", "[w_factor]/[w_factor_
total]")
    arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_hosp", "{} * [w_factor_percent] *
1/2".format(pop_besch))
    arcpy.CalculateField_management("input_fc_f_layer", "percent", "1")

#---------------------------------------#

# Calculate p15_64_w_hosp_athome (1/2)
arcpy.AddField_management(input_fc_h_univ_vhs, "w_factor_total_hosp_athome", "DOUBLE")
arcpy.AddField_management(input_fc_h_univ_vhs, "w_factor_percent_hosp_athome", "DOUBLE")
arcpy.AddField_management(input_fc_h_univ_vhs, "p15_64_w_hosp_athome", "DOUBLE")

# Selecting Features (Wohngebäude)
arcpy.MakeFeatureLayer_management (input_fc_h_univ_vhs, "input_fc_f_layer")

use = living
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(USE_bkm, use))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get Number of Employees at home
arcpy.Statistics_analysis(input_table1_h_univ_vhs, table_temp, "p15_64_w_hosp SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_hosp"
for row in SC:
    pop = row.getValue(field_sum)
    pop_1_2 = pop * 1/2

# Process: Calculate Fields
print "Calculating Fields for p15_64_w_hosp_athome (1/2)"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_hosp_athome", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_hosp_athome", "[w_factor]/
[w_factor_total_hosp_athome]")
arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_hosp_athome", "{} * [w_factor_per-
cent_hosp_athome]".format(pop_1_2))

#---------------------------------------#

# Process: Spatial Join 1
print "\nSpatial Join: ogd_univ to bkm"
output_spatialjoin1 = wsTemp + os.sep + "output_spatialjoin_univ"
arcpy.SpatialJoin_analysis(input_fc_h_univ_vhs, input_fc3_h_univ_vhs, output_spatialjoin1, "JOIN_
ONE_TO_ONE", "KEEP_ALL", "",
                           "INTERSECT", "0.5 Meters")

del_list = [NAME_univ1, NAME_univ2]
arcpy.DeleteField_management(input_fc_h_univ_vhs, del_list)
arcpy.JoinField_management(input_fc_h_univ_vhs, ID_bkm, output_spatialjoin1, ID_bkm, "{}; {}".for-
mat(NAME_univ1, NAME_univ2))

# Process: Spatial Join 2
print "Spatial Join: ogd_vhs to bkm\n"
output_spatialjoin2 = wsTemp + os.sep + "output_spatialjoin_univ"
arcpy.SpatialJoin_analysis(input_fc_h_univ_vhs, input_fc4_h_univ_vhs, output_spatialjoin2, "JOIN_
ONE_TO_ONE", "KEEP_ALL", "",
                           "INTERSECT", "0.5 Meters")

arcpy.DeleteField_management(input_fc_h_univ_vhs, ["{}".format(NAME_vhs)])
arcpy.JoinField_management(input_fc_h_univ_vhs, ID_bkm, output_spatialjoin2, ID_bkm, "{}".for-
mat(NAME_vhs))

# Calculate p15_64_w_univ & p15_64_w_vhs
SC = arcpy.SearchCursor(input_table2_h_univ_vhs)
field_id = ID_count2
field_name = NAME_univ
field_type = TYPE_univ
field_pop = POP_15_64_univ_vhs
for row in SC:
    f_id = row.getValue(field_id)
    name = row.getValue(field_name)
    type = row.getValue(field_type)
    pop_besch = row.getValue(field_pop)

    arcpy.MakeFeatureLayer_management (input_fc_h_univ_vhs, "input_fc_f_layer")
    if type == type_univ:
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".
format(NAME_univ1, name))

        # Process: Summary Statistics
        arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = "SUM_{}".format(w_factor_bkm)
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print "Calculating Fields for " + str(name) + " (" + str(int(round(f_id,0))) + ")"
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total", value)
        arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent", "[w_factor]/[w_fac-
tor_total]")
        arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_univ", "{} * [w_factor_per-
cent]”.format(pop_besch))
        arcpy.CalculateField_management(“input_fc_f_layer”, “percent”, “1”)

    else:
        arcpy.SelectLayerByAttribute_management(“input_fc_f_layer”, “NEW_SELECTION”, “{} IS NOT 
NULL”.format(NAME_vhs))

        # Process: Summary Statistics
        arcpy.Statistics_analysis(“input_fc_f_layer”, table_temp, “{} SUM”.format(w_factor_bkm))

        # Get Field Value
        SC2 = arcpy.SearchCursor(table_temp)
        field_sum = “SUM_{}”.format(w_factor_bkm)
        for row2 in SC2:
            value = row2.getValue(field_sum)

        # Process: Calculate Fields
        print “Calculating Fields for “ + str(name) + “ (“ + str(int(round(f_id,0))) + “)”
        arcpy.CalculateField_management(“input_fc_f_layer”, “w_factor_total”, value)
        arcpy.CalculateField_management(“input_fc_f_layer”, “w_factor_percent”, “[w_factor]/[w_fac-
tor_total]”)
        arcpy.CalculateField_management(“input_fc_f_layer”, “p15_64_w_vhs”, “{} * [w_factor_per-
cent]”.format(pop_besch))
        arcpy.CalculateField_management(“input_fc_f_layer”, “percent”, “1”)

# Calculate p15_64_w_hosp_univ_vhs and p15_64_w_univ_vhs, replace <NULL> with 0
expression = “myCalc(!p15_64_w_hosp!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(input_fc_h_univ_vhs, “p15_64_w_hosp”, expression, “PYTHON_9.3”, 
codeblock)

expression = “myCalc(!p15_64_w_hosp_athome!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(input_fc_h_univ_vhs, “p15_64_w_hosp_athome”, expression, “PY-
THON_9.3”, codeblock)

expression = “myCalc(!p15_64_w_univ!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(input_fc_h_univ_vhs, “p15_64_w_univ”, expression, “PYTHON_9.3”,
codeblock)

expression = "myCalc(!p15_64_w_vhs!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(input_fc_h_univ_vhs, "p15_64_w_vhs", expression, "PYTHON_9.3",
codeblock)

arcpy.CalculateField_management(input_fc_h_univ_vhs, "p15_64_w_univ_vhs", "[p15_64_w_univ] +
[p15_64_w_vhs]")
arcpy.CalculateField_management(input_fc_h_univ_vhs, "p15_64_w_hosp_univ_vhs", "[p15_64_w_hosp] +
[p15_64_w_univ] + [p15_64_w_vhs]")

# Statistics
print "\nStatistics:"
arcpy.Statistics_analysis(input_table1_h_univ_vhs, table_temp, "{} SUM".format(POP_15_64_hospital))
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_15_64_hospital)
for row in SC:
    pop = row.getValue(field_sum)
    pop_1_2 = pop * 1/2
    print "Number of employees (hospital) = " + str(int(round(pop,0)))
    print "Number of employees (hospital) * 1/2 = " + str(int(round(pop_1_2,0)))

arcpy.Statistics_analysis(input_fc_h_univ_vhs, table_temp, "p15_64_w_hosp SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_hosp"
for row in SC:
    distr_empl_hosp = row.getValue(field_sum)
    print "Number of distributed employees (hospital) = " + str(int(round(distr_empl_hosp,0)))

arcpy.Statistics_analysis(input_fc_h_univ_vhs, table_temp, "p15_64_w_hosp_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_hosp_athome"
for row in SC:
    empl_hosp_athome = row.getValue(field_sum)
    print "Number of distributed employees (hospital) at home = " + str(int(round(empl_hosp_ath-
ome,0)))

arcpy.Statistics_analysis(input_table2_h_univ_vhs, table_temp, "{} SUM".format(POP_15_64_univ_vhs))
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_15_64_univ_vhs)
for row in SC:
    pop = row.getValue(field_sum)
    print "\nNumber of employees (university + vhs) = " + str(int(round(pop,0)))

arcpy.Statistics_analysis(input_fc_h_univ_vhs, table_temp, "p15_64_w_univ SUM")
SC = arcpy.SearchCursor(table_temp)
```

```
field_sum = "SUM_p15_64_w_univ"
for row in SC:
    distr_empl1 = row.getValue(field_sum)

arcpy.Statistics_analysis(input_fc_h_univ_vhs, table_temp, "p15_64_w_vhs SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_vhs"
for row in SC:
    distr_empl2 = row.getValue(field_sum)

distr_empl = distr_empl1 + distr_empl2
print "Number of distributed employees (university + vhs) = " + str(int(round(distr_empl,0)))

arcpy.Statistics_analysis(input_fc_h_univ_vhs, table_temp, "p15_64_w_hosp_univ_vhs SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_hosp_univ_vhs"
for row in SC:
    distr_empl_total = row.getValue(field_sum)

contr_sum = distr_empl_hosp + distr_empl
print "\nNumber of total distr empl (hosp + univ + vhs) = " + str(int(round(distr_empl_total,0))) +
" (" + str(int(round(contr_sum,0))) + ")"

#-------------------------------------------ANALYSIS 2-------------------------------------------#
#----------------------------Get distributed Population per BEZ-----------------------------------#
print "\n---------------------------------ANALYSIS 2---------------------------------"
print "---------------------Get distributed Population per BEZ---------------------\n"

# Select buildings of bkm_output where population has been distributed
print "Select buildings of bkm_output (univ+vhs)"
arcpy.MakeFeatureLayer_management (input_fc_h_univ_vhs, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "p15_64_w_univ_vhs IS
NOT NULL")

# Make Feature Layer for Identity
arcpy.MakeFeatureLayer_management("input_fc_f_layer", "input_fc_f_layer2", "", "", "percent percent
VISIBLE RATIO")

# Identity, Dissolve (get distributed population per BEZ)
print "Identity (bkm, bez_etrs)"
output_identity = wsTemp + os.sep + "output_identity_univ_vhs"
arcpy.Identity_analysis("input_fc_f_layer2", input_fc2_h_univ_vhs, output_identity, "ALL", "", "NO_
RELATIONSHIPS")

arcpy.AddField_management(output_identity, "distr_pop_univ_vhs", "DOUBLE")
arcpy.CalculateField_management(output_identity, "distr_pop_univ_vhs", "[p15_64_w_univ_vhs]*[per-
cent]")

print "Dissolve"
output_dissolve = wsTemp + os.sep + "output_dissolve"
arcpy.Dissolve_management(output_identity, output_dissolve, ID_bez, "distr_pop_univ_vhs SUM", "MUL-
TI_PART", "DISSOLVE_LINES")

# Get distributed Population per BEZ
print "Join Field SUM_distr_pop_univ_vhs to bkm"
arcpy.DeleteField_management(input_fc2_h_univ_vhs, ["SUM_distr_pop_univ_vhs"])
arcpy.JoinField_management(input_fc2_h_univ_vhs, ID_bez, output_dissolve, ID_bez, "SUM_distr_pop_
univ_vhs")

print "\nGet distributed population per BEZ:"
SC = arcpy.SearchCursor(input_fc2_h_univ_vhs)
f_id = ID_bez
field_val = "SUM_distr_pop_univ_vhs"
for row in SC:
    bez_id = row.getValue(f_id)
    val = row.getValue(field_val)
    if val is not None:
        print "Distributed Population (univ+vhs) for BEZ " + str(bez_id) + " = " +
str(int(round(val,0)))
    else:
        print "Distributed Population (univ+vhs) for BEZ " + str(bez_id) + " = " + str(val)

arcpy.Statistics_analysis(input_fc2_h_univ_vhs, table_temp, "SUM_distr_pop_univ_vhs SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_univ_vhs"
for row in SC:
    sum1 = row.getValue(field_sum)
    print "\nTotal Distributed Population = " + str(int(round(sum1,0)))

#--------------------------------------------------------------------------#

# Select buildings of bkm_output where population has been distributed
print "\nSelect buildings of bkm_output (hosp)"
arcpy.MakeFeatureLayer_management (input_fc_h_univ_vhs, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "p15_64_w_hosp IS NOT
NULL")

# Make Feature Layer for Identity
arcpy.MakeFeatureLayer_management("input_fc_f_layer", "input_fc_f_layer2", "", "", "percent percent
VISIBLE RATIO")

# Identity, Dissolve (get distributed population per BEZ)
print "Get distributed population per BEZ:"
output_identity = wsTemp + os.sep + "output_identity_hosp"
arcpy.Identity_analysis("input_fc_f_layer2", input_fc2_h_univ_vhs, output_identity, "ALL", "", "NO_
RELATIONSHIPS")

arcpy.AddField_management(output_identity, "distr_pop_hosp", "DOUBLE")
arcpy.CalculateField_management(output_identity, "distr_pop_hosp", "[p15_64_w_hosp]*[percent]")

output_dissolve = wsTemp + os.sep + "output_dissolve"
arcpy.Dissolve_management(output_identity, output_dissolve, ID_bez, "distr_pop_hosp SUM", "MULTI_
PART”, “DISSOLVE_LINES”)

# Get distributed Population per BEZ
arcpy.DeleteField_management(input_fc2_h_univ_vhs, [“SUM_distr_pop_hosp”])
arcpy.JoinField_management(input_fc2_h_univ_vhs, ID_bez, output_dissolve, ID_bez, “SUM_distr_pop_
hosp”)

SC = arcpy.SearchCursor(input_fc2_h_univ_vhs)
f_id = ID_bez
field_val = “SUM_distr_pop_hosp”
for row in SC:
    bez_id = row.getValue(f_id)
    val = row.getValue(field_val)
    if val is not None:
        print “Distributed Population (hosp) for BEZ “ + str(bez_id) + “ = “ +
str(int(round(val,0)))
    else:
        print “Distributed Population (hosp) for BEZ “ + str(bez_id) + “ = “ + str(val)

arcpy.Statistics_analysis(input_fc2_h_univ_vhs, table_temp, “SUM_distr_pop_hosp SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_SUM_distr_pop_hosp”
for row in SC:
    sum1 = row.getValue(field_sum)
    print “\nTotal Distributed Population = “ + str(int(round(sum1,0)))

#-------------------------------------------------------------------------------------------------#
print “\n---------------------------------------------------------------------------”

print “\nOutput Data:”
print output_bkm + analysis_no + “ (polygon_fc)”
print output_bez + analysis_no + “ (polygon_fc)”

print “\nOutput Fields in “ + output_bkm + analysis_no + “:”
print “p15_64_w_hosp”
print “p15_64_w_hosp_athome”
print “p15_64_w_univ”
print “p15_64_w_vhs”
print “p15_64_w_univ_vhs”
print “p15_64_w_hosp_univ_vhs”

print “\nOutput Fields in “ + output_bez + analysis_no + “:”
print “SUM_distr_pop_hosp”
print “SUM_distr_pop_univ_vhs”
print “SUM_distr_pop_hosp_univ_vhs”


end = datetime.datetime.now()

print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start)
```

```
###################################################################################################
print "\n##########################################################################\n\n"


#-------------------------------------------------------------------------------------------------#
#Distribute Working Population (Firmenreg) to bkm and subtract from bez (age15_64_plus_pendler----#
#-------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

if analysis_no == "_03":
    print "--------------------------------------------------------------------------"
    print "-----------Create bkm_output2 and set SUM_p15_64_firmenreg to 0------------"
    print "--------------------------------------------------------------------------\n"

    input_fc_firmenreg = wsOutput + os.sep + output_bkm + analysis_no # bkm_output
    input_fc2_firmenreg = wsOutput + os.sep + output_bez + analysis_no # bez_etrs
    output_fc = wsOutput + os.sep + output_name_firmenreg # bkm_output2

    print "Input data:"
    print output_bkm + analysis_no + " (polygon_fc)"
    print output_bez + analysis_no + " (polygon_fc)"

    arcpy.Copy_management (input_fc_firmenreg, output_fc)
    arcpy.AddField_management (output_fc, "SUM_p15_64_firmenreg")
    arcpy.CalculateField_management (output_fc, "SUM_p15_64_firmenreg", "0")

    arcpy.DeleteField_management (input_fc2_firmenreg, ["SUM_distr_pop_firmenreg"])
    arcpy.AddField_management (input_fc2_firmenreg, "SUM_distr_pop_firmenreg")
    arcpy.CalculateField_management (input_fc2_firmenreg, "SUM_distr_pop_firmenreg", "0")

    #---------------------------------------------------------------------------------------------#
    print "\n--------------------------------------------------------------------------"

    print "\nOutput Data:"
    print output_name_firmenreg + " (polygon_fc)"
    print output_bez + analysis_no + " (polygon_fc)"

    print "\nOutput Fields in " + output_name_firmenreg + ":"
    print "SUM_p15_64_firmenreg"

    print "\nOutput Fields in " + output_bez + analysis_no + ":"
    print "SUM_distr_pop_firmenreg"


    end = datetime.datetime.now()
    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s\n' % (end-start)

else:
    print "---------------------------------------------------------------------------"
    print "----Distribute Working Pop (firmenregister) to bkm & subtract from BEZ----"
    print "---------------------------------------------------------------------------\n"

    input_fc_firmenreg = wsOutput + os.sep + output_bkm + analysis_no # bkm_output
    input_fc2_firmenreg = wsOutput + os.sep + output_bez + analysis_no # bez_etrs
    input_fc3_firmenreg = wsInput + os.sep + point_adr # address_point
    input_table_firmenreg = gdb + os.sep + table_firmenreg # bkm_firmenreg

    output_fc = wsOutput + os.sep + output_name_firmenreg # bkm_output2

    print "Input data:"
    print output_bkm + analysis_no + " (polygon_fc)"
    print output_bez + analysis_no + " (polygon_fc)"
    print point_adr + " (point_fc)"
    print table_firmenreg + " (table)"

#--------------------------------------------ANALYSIS 1--------------------------------------------#
    #----------Distribute Population from Firmenregister to Buildings-----------------------#
    print "\n---------------------------------ANALYSIS 1-------------------------------"
    print "----------Distribute Population from Firmenregister to Buildings----------\n"

    print "Dissolve adr_points"
    adr_points_diss = wsTemp + os.sep + "adr_points_diss"
    arcpy.Dissolve_management(input_fc3_firmenreg, adr_points_diss, join_adr, "", "SINGLE_PART",
"DISSOLVE_LINES")

    print "Join Field"
    arcpy.JoinField_management(adr_points_diss, join_adr, input_table_firmenreg, join_adr_firmenreg,
"{};{}".format(POP_15_64_firmenreg, NAME_firmenreg))

    # Spatial Join: ONE TO MANY because occasionally there are multiple points for one building
    print "Spatial Join (1)"
    temp_fc = wsTemp + os.sep + "output_spatial_join"
    arcpy.SpatialJoin_analysis(input_fc_firmenreg, adr_points_diss, temp_fc, "JOIN_ONE_TO_MANY",
"KEEP_ALL",
    """{2} "{2}" true true false 8 Double 0 0 ,First,#,{0},{2},-1,-1;
            {3} "{3}" true true false 8 Double 0 0 ,First,#,{0},{3},-1,-1;
            {4} "{4}" true true false 8 Double 0 0 ,First,#,{0},{4},-1,-1;
            {5} "{5}" true true false 8 Double 0 0 ,First,#,{0},{5},-1,-1;
            {6} "{6}" true true false 150 Text 0 0 ,First,#,{0},{6},-1,-1;
            {7} "{7}" true true false 50 Text 0 0 ,First,#,{0},{7},-1,-1;
            {8} "{8}" true true false 70 Text 0 0 ,First,#,{0},{8},-1,-1;
            {9} "{9}" false true true 8 Double 0 0 ,First,#,{0},{9},-1,-1;
            {10} "{10}" false true true 8 Double 0 0 ,First,#,{0},{10},-1,-1;
            w_factor_total "w_factor_total" true true false 8 Double 0 0 ,First,#,{0},w_factor_to-
tal,-1,-1;
```

```
            w_factor_percent "w_factor_percent" true true false 8 Double 0 0 ,First,#,{0},w_fac-
tor_percent,-1,-1;
            p15_64_w_hosp "p15_64_w_hosp" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_hosp,-
1,-1;
            p15_64_w_univ "p15_64_w_univ" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_univ,-
1,-1;
            p15_64_w_vhs "p15_64_w_vhs" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_vhs,-1,-
1;
            p15_64_w_hosp_univ_vhs "p15_64_w_hosp_univ_vhs" true true false 8 Double 0 0
,First,#,{0},p15_64_w_hosp_univ_vhs,-1,-1;
            percent "percent" true true false 8 Double 0 0 ,First,#,{0},percent,-1,-1;
            {11} "{11}" true true false 250 Text 0 0 ,First,#,{0},{11},-1,-1;
            {12} "{12}" true true false 150 Text 0 0 ,First,#,{0},{12},-1,-1;
            {13} "{13}" true true false 250 Text 0 0 ,First,#,{0},{13},-1,-1;
            {14} "{14}" true true false 100 Text 0 0 ,First,#,{1},{14},-1,-1;
            {15} "{15}" true true false 8 Double 0 0 ,Sum,#,{1},{15},-1,-1;
            {16} "{16}" true true false 150 Text 0 0 ,First,#,{1},{16},-1,-1"""
            .format(input_fc_firmenreg, adr_points_diss, w_factor_bkm, ID_bkm, height_bkm, vol_bkm,
ID_hospital, NAME_hospital, USE_bkm,
               length_bkm, area_bkm, NAME_univ1, NAME_univ2, NAME_vhs, join_adr_firmenreg,
POP_15_64_firmenreg, NAME_firmenreg),
               "INTERSECT", "0.5 Meters", "#")

    # Check if Spatial Join (1) worked - if not (code_bkm = 0) execute spatialjoin again
    var = arcpy.da.SearchCursor(temp_fc, ("code_bkm",)).next()[0]
    if var == 0:
        print "Spatial Join (2)"
        temp_fc = wsTemp + os.sep + "output_spatial_join"
        arcpy.SpatialJoin_analysis(input_fc_firmenreg, adr_points_diss, temp_fc, "JOIN_ONE_TO_MANY",
"KEEP_ALL",
    """{2} "{2}" true true false 8 Double 0 0 ,First,#,{0},{2},-1,-1;
            {3} "{3}" true true false 8 Double 0 0 ,First,#,{0},{3},-1,-1;
            {4} "{4}" true true false 8 Double 0 0 ,First,#,{0},{4},-1,-1;
            {5} "{5}" true true false 8 Double 0 0 ,First,#,{0},{5},-1,-1;
            {6} "{6}" true true false 150 Text 0 0 ,First,#,{0},{6},-1,-1;
            {7} "{7}" true true false 50 Text 0 0 ,First,#,{0},{7},-1,-1;
            {8} "{8}" true true false 70 Text 0 0 ,First,#,{0},{8},-1,-1;
            {9} "{9}" false true true 8 Double 0 0 ,First,#,{0},{9},-1,-1;
            {10} "{10}" false true true 8 Double 0 0 ,First,#,{0},{10},-1,-1;
            w_factor_total "w_factor_total" true true false 8 Double 0 0 ,First,#,{0},w_factor_to-
tal,-1,-1;
            w_factor_percent "w_factor_percent" true true false 8 Double 0 0 ,First,#,{0},w_fac-
tor_percent,-1,-1;
            p15_64_w_hosp "p15_64_w_hosp" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_hosp,-
1,-1;
            p15_64_w_univ "p15_64_w_univ" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_univ,-
1,-1;
            p15_64_w_vhs "p15_64_w_vhs" true true false 8 Double 0 0 ,First,#,{0},p15_64_w_vhs,-1,-
1;
            p15_64_w_hosp_univ_vhs "p15_64_w_hosp_univ_vhs" true true false 8 Double 0 0
,First,#,{0},p15_64_w_hosp_univ_vhs,-1,-1;
```

```
            percent "percent" true true false 8 Double 0 0 ,First,#,{0},percent,-1,-1;
            {11} "{11}" true true false 250 Text 0 0 ,First,#,{0},{11},-1,-1;
            {12} "{12}" true true false 150 Text 0 0 ,First,#,{0},{12},-1,-1;
            {13} "{13}" true true false 250 Text 0 0 ,First,#,{0},{13},-1,-1;
            {14} "{14}" true true false 100 Text 0 0 ,First,#,{1},{14},-1,-1;
            {15} "{15}" true true false 8 Double 0 0 ,Sum,#,{1},{15},-1,-1;
            {16} "{16}" true true false 150 Text 0 0 ,First,#,{1},{16},-1,-1"""
            .format(input_fc_firmenreg, adr_points_diss, w_factor_bkm, ID_bkm, height_bkm, vol_bkm,
ID_hospital, NAME_hospital, USE_bkm,
              length_bkm, area_bkm, NAME_univ1, NAME_univ2, NAME_vhs, join_adr_firmenreg,
POP_15_64_firmenreg, NAME_firmenreg),
              "INTERSECT", "0.5 Meters", "#")

    # Check if Spatial Join (2) worked - if not (code_bkm = 0) print error message and exit
    var = arcpy.da.SearchCursor(temp_fc, ("{}".format(ID_bkm),)).next()[0]
    if var == 0:
        print "ERROR IN FIELD MAPS"
        sys.exit()

    # Dissolve bkm and build sum of p15_64_firmenreg
    print "Dissolve"
    temp_fc2 = wsTemp + os.sep + "diss_firmenreg"
    arcpy.Dissolve_management(temp_fc, temp_fc2, ID_bkm, "{} SUM".format(POP_15_64_firmenreg), "SIN-
GLE_PART", "DISSOLVE_LINES")

    # Join field SUM_p15_64_firmenreg from input_table to bkm
    print "Join Field"
    arcpy.Copy_management (input_fc_firmenreg, output_fc)
    arcpy.JoinField_management (output_fc, ID_bkm, temp_fc2, ID_bkm, ["SUM_{}".format(POP_15_64_fir-
menreg)])

    # Get number of distributed employees
    arcpy.Statistics_analysis(output_fc, table_temp, "SUM_{} SUM".format(POP_15_64_firmenreg))

    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_{}".format(POP_15_64_firmenreg)
    for row in SC:
        distr_empl = row.getValue(field_sum)
        print "\nNumber of distributed employees = " + str(int(round(distr_empl,0)))

#--------------------------------------------ANALYSIS 1----------------------------------------------#
    #-------------------------Get distributed Population per BEZ-----------------------------------#
    print "\n--------------------------------ANALYSIS 2--------------------------------"
    print "--------------------Get distributed Population per BEZ--------------------\n"

    # Select buildings of bkm_output where population has been distributed
    print "Select buildings of bkm_output"
    arcpy.MakeFeatureLayer_management (output_fc, "input_fc_f_layer")
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "SUM_{} IS NOT
NULL".format(POP_15_64_firmenreg))
    arcpy.CalculateField_management("input_fc_f_layer", "percent", "1")
    # Make Feature Layer for Identity
    arcpy.MakeFeatureLayer_management("input_fc_f_layer", "input_fc_f_layer2", "", "", "percent
percent VISIBLE RATIO")

    # Identity, Dissolve (get distributed population per BEZ)
    print "Get distributed population per BEZ:"
    output_identity = wsTemp + os.sep + "output_identity"
    arcpy.Identity_analysis("input_fc_f_layer2", input_fc2_firmenreg, output_identity, "ALL", "",
"NO_RELATIONSHIPS")

    arcpy.AddField_management(output_identity, "distr_pop_firmenreg", "DOUBLE")
    arcpy.CalculateField_management(output_identity, "distr_pop_firmenreg", "[SUM_{}]*[percent]".
format(POP_15_64_firmenreg))

    output_dissolve = wsTemp + os.sep + "output_dissolve"
    arcpy.Dissolve_management(output_identity, output_dissolve, ID_bez, "distr_pop_firmenreg SUM",
"MULTI_PART", "DISSOLVE_LINES")

    # Get distributed Population per BEZ
    arcpy.DeleteField_management(input_fc2_firmenreg, ["SUM_distr_pop_firmenreg"])
    arcpy.JoinField_management(input_fc2_firmenreg, ID_bez, output_dissolve, ID_bez, "SUM_distr_pop_
firmenreg")

    SC = arcpy.SearchCursor(input_fc2_firmenreg)
    f_id = ID_bez
    field_val = "SUM_distr_pop_firmenreg"
    for row in SC:
        bez_id = row.getValue(f_id)
        val = row.getValue(field_val)
        if val is not None:
            print "Distributed Population for BEZ " + str(bez_id) + " = " + str(int(round(val,0)))
        else:
            print "Distributed Population for BEZ " + str(bez_id) + " = " + str(val)

    arcpy.Statistics_analysis(input_fc2_firmenreg, table_temp, "SUM_distr_pop_firmenreg SUM")
    SC = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_SUM_distr_pop_firmenreg"
    for row in SC:
        sum1 = row.getValue(field_sum)
        print "\nTotal Distributed Population = " + str(int(round(sum1,0)))

    #---------------------------------------------------------------------------------------------#
    print "\n---------------------------------------------------------------------------"

    print "\nOutput Data:"
    print output_name_firmenreg + " (polygon_fc)"
    print output_bez + analysis_no + " (polygon_fc)"

    print "\nOutput Fields in " + output_name_firmenreg + ":"
    print "SUM_{}".format(POP_15_64_firmenreg)
    print "\nOutput Fields in " + output_bez + analysis_no + ":"
    print "SUM_distr_pop_firmenreg"


    end = datetime.datetime.now()

    print '\nfinished run: %0.19s\n' % (end),
    print 'runtime: %0.7s\n' % (end-start)


###################################################################################################
print "\n##########################################################################\n\n"


#-------------------------------------------------------------------------------------------------#
#----------Distribute Working Population (Wr Linien) to public_transportation and subtract from bez
(age15_64_plus_pendler)---------#
#-------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "-------------------------------------------------------------------------"
print "---Distr Work Pop (Wr Linien) to public_transportation & subtr from BEZ---"
print "-------------------------------------------------------------------------\n"

input_fc_wrlinien = wsInput + os.sep + network_transp # public_transportation
input_fc2_wrlinien = wsOutput + os.sep + output_bez + analysis_no # bez_etrs
input_fc3_wrlinien = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_table_wrlinien = gdb + os.sep + table_wrlinien_empl # besch_wr_linien
output_fc_wrlinien = wsOutput + os.sep + output_name_wrlinien # public_transportation_buffer

print "Input data:"
print network_transp + " (line_fc)"
print output_bez + analysis_no + " (polygon_fc)"
print table_wrlinien_empl + " (table)"

#-----------------------------------------ANALYSIS 1---------------------------------------------#
#--------Distribute Wiener Linien Population (age15_64) to public_transportation------------------#
print "\n----------------------------------ANALYSIS 1--------------------------------"
print "----Distribute Working Population (Wr Linien) to public_transportation----\n"

print "\nProcess: Buffer (2.5m)"
out_buffer = wsTemp + os.sep + "public_transportation_buffer_temp"
arcpy.Buffer_analysis (input_fc_wrlinien, out_buffer, "2.5 Meters", "FULL", "ROUND", "ALL")

print "Process: Clip"
arcpy.Clip_analysis (out_buffer, input_fc2_wrlinien, output_fc_wrlinien)
```

```
SC = arcpy.SearchCursor(input_table_wrlinien)
field = "{}".format(POP_15_64_wrlinien)
for row in SC:
    employees_wr_linien = row.getValue(field)
    employees_wr_linien2 = employees_wr_linien/2

arcpy.AddField_management(output_fc_wrlinien, "p15_64_wr_linien", "DOUBLE")
arcpy.CalculateField_management(output_fc_wrlinien, "p15_64_wr_linien", "{}/2".format(employees_wr_
linien))

print "\nNumber of employees = " + str(int(round(employees_wr_linien,0)))
print "Number of employees / 2 = " + str(int(round(employees_wr_linien2,0)))

arcpy.Statistics_analysis(output_fc_wrlinien, table_temp, "p15_64_wr_linien SUM")
SC = arcpy.SearchCursor(table_temp)
field = "SUM_p15_64_wr_linien"
for row in SC:
    distr_empl = row.getValue(field)

print "Number of distributed employees (empl/2) = " + str(int(round(distr_empl,0)))

#-------------------------------------------ANALYSIS 2--------------------------------------------#
#----------------Distribute Wiener Linien Population (age15_64) to Wohngebäude--------------------#
print "\n--------------------------------ANALYSIS 2------------------------------"
print "------Distribute Wiener Linien Population (age15_64) to Wohngebäude-------\n"

# Calculate p15_64_w_wrlinien_athome
arcpy.DeleteField_management(input_fc3_wrlinien, ["w_factor_total_wrlinien_athome", "w_factor_per-
cent_wrlinien_athome",
                                                  "p15_64_w_wrlinien_athome"])

arcpy.AddField_management(input_fc3_wrlinien, "w_factor_total_wrlinien_athome", "DOUBLE")
arcpy.AddField_management(input_fc3_wrlinien, "w_factor_percent_wrlinien_athome", "DOUBLE")
arcpy.AddField_management(input_fc3_wrlinien, "p15_64_w_wrlinien_athome", "DOUBLE")

# Selecting Features (Wohngebäude)
arcpy.MakeFeatureLayer_management (input_fc3_wrlinien, "input_fc_f_layer")

use = living
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(USE_bkm, use))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)
```

```
# Process: Calculate Fields
print "Calculating Fields for p15_64_w_wrlinien_athome (1/2)"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_wrlinien_athome", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_wrlinien_athome", "[{}]/[w_
factor_total_wrlinien_athome]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_wrlinien_athome",
                                "{} * [w_factor_percent_wrlinien_athome]".format(employees_wr_lin-
ien2))

# Statistics
arcpy.Statistics_analysis(input_fc3_wrlinien, table_temp, "p15_64_w_wrlinien_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_wrlinien_athome"
for row in SC:
    sum = row.getValue(field_sum)
    print "\nNumber of employees / 2 = " + str(int(round(employees_wr_linien2,0)))
    print "Distributed employees at home = " + str(int(round(sum,0)))

#------------------------------------------ANALYSIS 3--------------------------------------------#
#-------------------------------Get distributed Population per BEZ------------------------------#
print "\n---------------------------------ANALYSIS 3---------------------------------"
print "---------------------Get distributed Population per BEZ---------------------\n"

arcpy.AddField_management(output_fc_wrlinien, "percent", "DOUBLE")
arcpy.CalculateField_management(output_fc_wrlinien, "percent", "1")

# Make Feature Layer for Identity
arcpy.MakeFeatureLayer_management(output_fc_wrlinien, "input_fc_f_layer", "", "", "percent percent
VISIBLE RATIO")

# Identity, Dissolve (get distributed population per BEZ)
print "Get distributed population per BEZ:"
output_identity = wsTemp + os.sep + "output_identity"
arcpy.Identity_analysis("input_fc_f_layer", input_fc2_wrlinien, output_identity, "ALL", "", "NO_RE-
LATIONSHIPS")

arcpy.AddField_management(output_identity, "distr_pop_wrlinien", "DOUBLE")
arcpy.CalculateField_management(output_identity, "distr_pop_wrlinien", "[p15_64_wr_linien]*[per-
cent]")

output_dissolve = wsTemp + os.sep + "output_dissolve"
arcpy.Dissolve_management(output_identity, output_dissolve, ID_bez, "distr_pop_wrlinien SUM", "MUL-
TI_PART", "DISSOLVE_LINES")

# Get distributed Population per BEZ
arcpy.DeleteField_management(input_fc2_wrlinien, ["SUM_distr_pop_wrlinien"])
arcpy.JoinField_management(input_fc2_wrlinien, ID_bez, output_dissolve, ID_bez, "SUM_distr_pop_
wrlinien")

SC = arcpy.SearchCursor(input_fc2_wrlinien)
f_id = ID_bez
```

```
field_val = "SUM_distr_pop_wrlinien"
for row in SC:
    bez_id = row.getValue(f_id)
    val = row.getValue(field_val)
    if val is not None:
        print "Distributed Population for BEZ " + str(bez_id) + " = " + str(int(round(val,0)))
    else:
        print "Distributed Population for BEZ " + str(bez_id) + " = " + str(val)

arcpy.Statistics_analysis(input_fc2_wrlinien, table_temp, "SUM_distr_pop_wrlinien SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_wrlinien"
for row in SC:
    sum1 = row.getValue(field_sum)
    print "\nTotal Distributed Population = " + str(int(round(sum1,0)))

#----------------------------------------------------------------------------------------------------#
print "\n---------------------------------------------------------------------------"

print "\nOutput Data:"
print output_name_wrlinien + " (polygon_fc)"
print output_bez + analysis_no + " (polygon_fc)"
print output_bkm2 + analysis_no + " (polygon_fc)"

print "\nOutput Fields in " + output_name_wrlinien + ":"
print "p15_64_wr_linien"

print "\nOutput Fields in " + output_bez + analysis_no + ":"
print "SUM_distr_pop_wrlinien"

print "\nOutput Fields in " + output_bkm2 + analysis_no + ":"
print "p15_64_w_wrlinien_athome"

end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)


####################################################################################################
print "\n###########################################################################\n\n"


#----------------------------------------------------------------------------------------------------#
#-------------------Distribute remaining working population to Gewerbegebäude---------------------#
#----------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)
print "---------------------------------------------------------------------------"
print "--------Distribute remaining working population to Gewerbegebäude--------"
print "---------------------------------------------------------------------------\n"

input_fc_rem_workpop = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_fc2_rem_workpop = wsOutput + os.sep + output_bez + analysis_no # bez_etrs
input_table_rem_workpop = gdb + os.sep + table_total_empl # besch_gesamt

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print output_bez + analysis_no + " (polygon_fc)"
print table_total_empl + " (table)"

#------------------------------------------ANALYSIS 1-------------------------------------------#
#-------Calculate distributed working pop (hosp, univ, firmenreg, wrlinien) per BEZ----------------#
#----------------------------------and subtract from besch_gesamt---------------------------------#
print "\n---------------------------------ANALYSIS 1-------------------------------"
print "----Calculate distributed working pop per BEZ and subtr from besch_ges----\n"

# Get distributed working pop
print "Get distributed working pop (hosp, univ, firmenreg, wrlinien):"
arcpy.Statistics_analysis(input_fc2_rem_workpop, table_temp, "SUM_distr_pop_hosp SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_hosp"
for row in SC:
    sum_hosp = row.getValue(field_sum)
    print "Distributed Working Pop (hosp) = " + str(int(round(sum_hosp,0)))

arcpy.Statistics_analysis(input_fc2_rem_workpop, table_temp, "SUM_distr_pop_univ_vhs SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_univ_vhs"
for row in SC:
    sum_univ_vhs = row.getValue(field_sum)
    print "Distributed Working Pop (univ, vhs) = " + str(int(round(sum_univ_vhs,0)))

arcpy.Statistics_analysis(input_fc2_rem_workpop, table_temp, "SUM_distr_pop_firmenreg SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_firmenreg"
for row in SC:
    sum_firmenreg = row.getValue(field_sum)
    print "Distributed Working Pop (firmenreg) = " + str(int(round(sum_firmenreg,0)))

arcpy.Statistics_analysis(input_fc2_rem_workpop, table_temp, "SUM_distr_pop_wrlinien SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_distr_pop_wrlinien"
for row in SC:
    sum_wrlinien = row.getValue(field_sum)
    print "Distributed Working Pop (wrlinien) = " + str(int(round(sum_wrlinien,0)))

sum_distr_working_pop = sum_hosp + sum_univ_vhs + sum_firmenreg + sum_wrlinien
sum_distr_working_pop_incl_athome = (sum_hosp * 2) + sum_univ_vhs + sum_firmenreg + (sum_wrlinien *
2)
print "\nTotal Distributed Working Population = " + str(int(round(sum_distr_working_pop,0)))
print "Total Distributed Working Population (incl. hosp_athome & wrlinien_athome) = " +
str(int(round(sum_distr_working_pop_incl_athome,0)))

# Calculate remaining working pop
print "\nCalculate remaining working pop (hosp, univ, firmenreg, wrlinien) per BEZ:"
arcpy.DeleteField_management(input_fc2_rem_workpop, ["{}".format(POP_15_64_besch_ges)])
arcpy.JoinField_management(input_fc2_rem_workpop, ID_bez, input_table_rem_workpop, ID_bez, "{}".
format(POP_15_64_besch_ges))

expression = "myCalc(!SUM_distr_pop_hosp!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(input_fc2_rem_workpop, "SUM_distr_pop_hosp", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!SUM_distr_pop_univ_vhs!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(input_fc2_rem_workpop, "SUM_distr_pop_univ_vhs", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!SUM_distr_pop_firmenreg!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(input_fc2_rem_workpop, "SUM_distr_pop_firmenreg", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!SUM_distr_pop_wrlinien!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(input_fc2_rem_workpop, "SUM_distr_pop_wrlinien", expression, "PY-
THON_9.3", codeblock)

arcpy.DeleteField_management(input_fc2_rem_workpop, ["remaining_working_pop"])
arcpy.AddField_management(input_fc2_rem_workpop, "remaining_working_pop", "DOUBLE")
arcpy.CalculateField_management(input_fc2_rem_workpop, "remaining_working_pop",
    "[{}]-(([SUM_distr_pop_hosp]*2)+[SUM_distr_pop_univ_vhs]+[SUM_distr_pop_firmenreg]+([SUM_dis-
tr_pop_wrlinien]*2))".format(POP_15_64_besch_ges))

# Statistics
SC = arcpy.SearchCursor(input_fc2_rem_workpop)
f_id = ID_bez
field_val = "remaining_working_pop"
for row in SC:
    bez_id = row.getValue(f_id)
    val = row.getValue(field_val)
    if val is not None:
        print "Remaining Working Population for BEZ " + str(bez_id) + " = " +
str(int(round(val,0)))
    else:
        print "Remaining Working Population for BEZ " + str(bez_id) + " = " + str(val)

arcpy.Statistics_analysis(input_table_rem_workpop, table_temp, "{} SUM".format(POP_15_64_besch_
ges))
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_15_64_besch_ges)
for row in SC:
    sum1 = row.getValue(field_sum)
    print "\nTotal Working Population = " + str(int(round(sum1,0)))

arcpy.Statistics_analysis(input_fc2_rem_workpop, table_temp, "remaining_working_pop SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_remaining_working_pop"
for row in SC:
    sum2 = row.getValue(field_sum)
    sum3 = sum1-sum_distr_working_pop_incl_athome
    print "Total Remaining Working Population = " + str(int(round(sum2,0))) + "(" + str(int(round(-
sum3,0))) + ")"

#------------------------------------------ANALYSIS 2---------------------------------------------#
#-----------------Distribute remaining working population to Gewerbegebäude-----------------------#
print "\n----------------------------------ANALYSIS 2-------------------------------"
print "---------Distribute remaining working population to Gewerbegebäude--------\n"

# Process: Identity (get buildings per BEZ)
print "\nIdentity (bkm, bez)"
output_identity = wsTemp + os.sep + "output_identity2"
arcpy.Identity_analysis(input_fc_rem_workpop, input_fc2_rem_workpop, output_identity)
arcpy.AddField_management(output_identity, "w_factor2", "DOUBLE")
arcpy.CalculateField_management(output_identity, "w_factor2", w_factor)

# Add Fields
arcpy.AddField_management(output_identity, "w_factor_total2", "DOUBLE")
arcpy.AddField_management(output_identity, "w_factor_percent2", "DOUBLE")
arcpy.AddField_management(output_identity, "p15_64_w_remaining_working_pop", "DOUBLE")

print "\nDistribute remaining working population to Gewerbegebäude:"
SC = arcpy.SearchCursor(input_fc2_rem_workpop)
```

```
field_id = ID_bez
field_pop = "remaining_working_pop"
for row in SC:
    f_id = row.getValue(field_id)
    pop_besch = row.getValue(field_pop)

    # Selecting Features (Gewerbegebäude) per BEZ
    arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

    use1 = trade
    use2 = industry
    use3 = social

    if analysis_no == "_01" or analysis_no == "_02":
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
        "{0} = {1} and {2} in ('{3}', '{4}') and SUM_{5} IS NULL or {0} = {1} and {2} = '{6}' and
{7} IS NOT NULL and p15_64_w_hosp_univ_vhs = 0"
               .format(ID_bez, f_id, USE_bkm, use1, use2, POP_15_64_firmenreg, use3, ID_hospital))

    elif analysis_no == "_03":
        arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
        "{0} = {1} and {2} in ('{3}', '{4}') and SUM_{5} = 0 or {0} = {1} and {2} = '{6}' and {7}
IS NOT NULL and p15_64_w_hosp_univ_vhs = 0"
               .format(ID_bez, f_id, USE_bkm, use1, use2, POP_15_64_firmenreg, use3, ID_hospital))

    # Process: Summary Statistics (getting total w_factor of all buildings per BEZ)
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_w_factor2"
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields
    print "Calculating Fields for BEZ " + str(f_id)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_fac-
tor_total2]")
    arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_remaining_working_pop", "{} *
[w_factor_percent2]".format(pop_besch))

print "\nDissolve output_identity"
output_diss_identity = wsTemp + os.sep + "output_diss_identity"
arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p15_64_w_remaining_work-
ing_pop SUM", "MULTI_PART", "DISSOLVE_LINES")

print "Join p15_64_w_remaining_working_pop to bkm_output2"
arcpy.DeleteField_management(input_fc_rem_workpop, ["SUM_p15_64_w_remaining_working_pop"])
arcpy.JoinField_management(input_fc_rem_workpop, ID_bkm, output_diss_identity, ID_bkm, "SUM_
p15_64_w_remaining_working_pop")
```

```
# Check distributed population
arcpy.Statistics_analysis(input_fc_rem_workpop, table_temp, "SUM_p15_64_w_remaining_working_pop
SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_p15_64_w_remaining_working_pop"
for row in SC:
    remaining_empl = row.getValue(field_sum)

print "\nDistributed Population = " + str(int(round(remaining_empl,0)))

#---------------------------------------------------------------------------------------------------#
print "\n--------------------------------------------------------------------------"

print "\nOutput Data:"
print output_bez + analysis_no + " (polygon_fc)"
print output_bkm2 + analysis_no + " (polygon_fc)"

print "\nOutput Fields in " + output_bez + analysis_no + ":"
print "remaining_working_pop"

print "\nOutput Fields in " + output_bkm2 + analysis_no + ":"
print "SUM_p15_64_w_remaining_working_pop"

end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)


####################################################################################################
print "\n##########################################################################\n\n"


#---------------------------------------------------------------------------------------------------#
#-----------------------------------Distribute Unemployed to bkm-----------------------------------#
#---------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "--------------------------------------------------------------------------"
print "-----------------------Distribute Unemployed to bkm-----------------------"
print "--------------------------------------------------------------------------\n"

input_fc_unempl = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_table_unempl = gdb + os.sep + table_unempl # unemployed

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print table_unempl + " (table)"

#--------------------------------------------ANALYSIS---------------------------------------------#
#------------------------------------Distribute Unemployed to bkm--------------------------------#
print "\n---------------------------------ANALYSIS---------------------------------"
print "-----------------------Distribute Unemployed to bkm-----------------------\n"

# Add Fields
arcpy.DeleteField_management(input_fc_unempl, ["w_factor_total_unempl", "w_factor_percent_unempl",
"p15_64_w_unemployed"])
arcpy.AddField_management(input_fc_unempl, "w_factor_total_unempl", "DOUBLE")
arcpy.AddField_management(input_fc_unempl, "w_factor_percent_unempl", "DOUBLE")
arcpy.AddField_management(input_fc_unempl, "p15_64_w_unemployed", "DOUBLE")

# Selecting Features (Wohngebäude)
arcpy.MakeFeatureLayer_management (input_fc_unempl, "input_fc_f_layer")

use = living
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(USE_bkm, use))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get Number of Unemployed
SC = arcpy.SearchCursor(input_table_unempl)
field_sum = POP_15_64_unemployed
for row in SC:
    pop_unempl = row.getValue(field_sum)

# Process: Calculate Fields
print "Calculating Fields"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_unempl", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_unempl", "[{}]/[w_factor_to-
tal_unempl]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p15_64_w_unemployed", "{} * [w_factor_percent_
unempl]".format(pop_unempl))

# Check distributed population
arcpy.Statistics_analysis(input_fc_unempl, table_temp, "p15_64_w_unemployed SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_64_w_unemployed"
for row in SC:
    pop = row.getValue(field_sum)
```

```
print "\nNumber of Unemployed = " + str(int(round(pop_unempl,0)))
print "Distributed Population (p15_64_w_unemployed) = " + str(int(round(pop,0)))

#--------------------------------------------------------------------------------------------------#
print "\n--------------------------------------------------------------------------"

print "\nOutput Data:"
print output_bkm2 + analysis_no + " (polygon_fc)"

print "\nOutput Fields in " + output_bkm2 + analysis_no + ":"
print "p15_64_w_unemployed"


end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)


####################################################################################################
print "\n##########################################################################\n\n"


#--------------------------------------------------------------------------------------------------#
#------------Distribute Population (age0_4_kiga - 50% and age0_4_athome - 50%) to bkm-------------#
#--------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "--------------------------------------------------------------------------"
print "-------Distribute Population (age0_4_kiga and age0_4_athome) to bkm-------"
print "--------------------------------------------------------------------------\n"

source_fc_kiga = wsInput + os.sep + input_zbez
arcpy.CopyFeatures_management(source_fc_kiga, wsOutput + os.sep + output_name_kiga)

input_fc_kiga = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_fc2_kiga = wsOutput + os.sep + output_name_kiga # zbez_output
input_fc3_kiga = wsInput + os.sep + point_kiga # ogd_kiga
input_fc4_kiga = wsOutput + os.sep + output_bez + analysis_no # bez_etrs

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print input_zbez + " (polygon_fc)"
print output_bez + analysis_no + " (polygon_fc)"
print point_kiga + " (point_fc)"
```

```
#-----------------------------------------ANALYSIS 1--------------------------------------------#
#----------------------Calculate age0_4_athome (50%) and age0_4_kiga (50%)------------------------#
print "\n--------------------------------ANALYSIS 1---------------------------------"
print "-------------Calculate age0_4_athome (50%) and age0_4_kiga (50%)----------\n"

print "Spatial Join: kiga to bkm"
del_list = [TYPE_kiga, BEZ_kiga]
arcpy.DeleteField_management(input_fc_kiga, del_list)

output_spatialjoin = wsTemp + os.sep + "output_spatialjoin_kiga"
arcpy.SpatialJoin_analysis(input_fc_kiga, input_fc3_kiga, output_spatialjoin, "JOIN_ONE_TO_ONE",
"KEEP_ALL", "", "INTERSECT", "0.5 Meters")
arcpy.JoinField_management(input_fc_kiga, ID_bkm, output_spatialjoin, ID_bkm, "{}; {}".format(TYPE_
kiga, BEZ_kiga))

print "Calculate age0_4_athome (50%) and age0_4_kiga (50%)"
arcpy.DeleteField_management(input_fc2_kiga, ["age0_4_athome", "age0_4_kiga"])
arcpy.AddField_management(input_fc2_kiga, "age0_4_athome", "DOUBLE")
arcpy.AddField_management(input_fc2_kiga, "age0_4_kiga", "DOUBLE")
arcpy.CalculateField_management(input_fc2_kiga, "age0_4_athome", "[{}]*0.5".format(POP_0_4))
arcpy.CalculateField_management(input_fc2_kiga, "age0_4_kiga", "[{}]*0.5".format(POP_0_4))

print "Process: Dissolve (get age0_4 SUM per BEZ)"
output_dissolve = wsTemp + os.sep + "output_dissolve_zbez2"
arcpy.Dissolve_management(input_fc2_kiga, output_dissolve, ID_bez, "age0_4_kiga SUM", "MULTI_PART",
"DISSOLVE_LINES")
arcpy.DeleteField_management(input_fc4_kiga, ["SUM_age0_4_kiga"])
arcpy.JoinField_management(input_fc4_kiga, ID_bez, output_dissolve, ID_bez, "SUM_age0_4_kiga")

# Statistics
print "\nStatistics:"
arcpy.Statistics_analysis(input_fc2_kiga, table_temp, "{} SUM".format(POP_0_4))
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_0_4)
for row in SC:
    sum1 = row.getValue(field_sum)
    print "age0_4 SUM (100%) = " + str(int(round(sum1,0)))

arcpy.Statistics_analysis(output_dissolve, table_temp, "SUM_age0_4_kiga SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_age0_4_kiga"
for row in SC:
    sum2 = row.getValue(field_sum)
    print "age0_4_kiga SUM (50%) = " + str(int(round(sum2,0)))

arcpy.Statistics_analysis(input_fc2_kiga, table_temp, "age0_4_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_age0_4_athome"
for row in SC:
    sum3 = row.getValue(field_sum)
    print "age0_4_athome SUM (50%) = " + str(int(round(sum3,0)))
```

```
#------------------------------------------ANALYSIS 2---------------------------------------------#
#--------------------------Distribute Population (age0_4_kiga - 50%) to bkm-----------------------#
print "\n--------------------------------ANALYSIS 2--------------------------------"
print "--------------Distribute Population (age0_4_kiga - 50%) to bkm------------\n"

# Process: Add Fields
print "Adding Fields"
arcpy.DeleteField_management(input_fc_kiga, ["w_factor_total_kiga", "w_factor_percent_kiga", 
"p0_4_w_kiga"])
arcpy.AddField_management(input_fc_kiga, "w_factor_total_kiga", "DOUBLE")
arcpy.AddField_management(input_fc_kiga, "w_factor_percent_kiga", "DOUBLE")
arcpy.AddField_management(input_fc_kiga, "p0_4_w_kiga", "DOUBLE")

print "\nDistribute age0_4_kiga to bkm per BEZ:"
SC = arcpy.SearchCursor(output_dissolve)
field_id = ID_bez
field_pop = "SUM_age0_4_kiga"
for row in SC:
    f_id = row.getValue(field_id)
    pop_kiga = row.getValue(field_pop)

    # Selecting Features (kiga) per BEZ
    arcpy.MakeFeatureLayer_management (input_fc_kiga, "input_fc_f_layer")
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = {}".for-
mat(BEZ_kiga, f_id))

    # Process: Summary Statistics (getting total w_factor of all buildings per BEZ)
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_{}".format(w_factor_bkm)
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields
    print "Calculating Fields for BEZ " + str(f_id)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_kiga", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_kiga", "[{}]/[w_factor_
total_kiga]".format(w_factor_bkm))
    arcpy.CalculateField_management("input_fc_f_layer", "p0_4_w_kiga", "{} * [w_factor_percent_
kiga]".format(pop_kiga))

# Statistics 2
arcpy.Statistics_analysis(input_fc_kiga, table_temp, "p0_4_w_kiga SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p0_4_w_kiga"
for row in SC:
    sum4 = row.getValue(field_sum)
    print "\nDistributed Population (p0_4_w_kiga) = " + str(int(round(sum4,0)))
```

```
#------------------------------------------ANALYSIS 3--------------------------------------------#
#--------------------------Distribute Population (age0_4_athome - 50%) to bkm-------------------#
print "\n-------------------------------ANALYSIS 3--------------------------------"
print "-------------Distribute Population (age0_4_athome - 50%) to bkm-----------\n"

# Process: Identity (get buildings per ZBEZ)
print "Identity (bkm, zbez)"
output_identity = wsTemp + os.sep + "output_identity3"
arcpy.Identity_analysis(input_fc_kiga, input_fc2_kiga, output_identity)
arcpy.AddField_management(output_identity, "w_factor2", "DOUBLE")
arcpy.CalculateField_management(output_identity, "w_factor2", w_factor)

# Add Fields
arcpy.AddField_management(output_identity, "w_factor_total2", "DOUBLE")
arcpy.AddField_management(output_identity, "w_factor_percent2", "DOUBLE")
arcpy.AddField_management(output_identity, "p0_4_w_athome", "DOUBLE")

print "\nDistribute p0_4_w_athome to Wohngebäude per ZBEZ:"
SC = arcpy.SearchCursor(input_fc2_kiga)
field_nr = OID
field_id = ID_zbez
field_pop = "age0_4_athome"
for row in SC:
    f_nr = row.getValue(field_nr)
    f_id = row.getValue(field_id)
    pop_athome = row.getValue(field_pop)

    # Selecting Features (Wohngebäude) per ZBEZ
    arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

    use = living
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                            "{} = {} and {} = '{}'".format(ID_zbez, f_id, USE_bkm,
use))

    # Process: Summary Statistics (getting total w_factor of all buildings per ZBEZ)
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor2 SUM")

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_w_factor2"
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields
    print "Calculating Fields for ZBEZ " + str(f_id) + "(" + str(f_nr) + ")"
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[w_factor2]/[w_fac-
tor_total2]")
    arcpy.CalculateField_management("input_fc_f_layer", "p0_4_w_athome", "{} * [w_factor_per-
cent2]".format(pop_athome))

print "\nDissolve output_identity"
output_diss_identity = wsTemp + os.sep + "output_diss_identity2"
arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p0_4_w_athome SUM", "MUL-
TI_PART", "DISSOLVE_LINES")

print "Join p0_4_w_athome to bkm_output2"
arcpy.DeleteField_management(input_fc_kiga, ["SUM_p0_4_w_athome"])
arcpy.JoinField_management(input_fc_kiga, ID_bkm, output_diss_identity, ID_bkm, "SUM_p0_4_w_ath-
ome")

# Check distributed population
arcpy.Statistics_analysis(input_fc_kiga, table_temp, "SUM_p0_4_w_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_SUM_p0_4_w_athome"
for row in SC:
    pop = row.getValue(field_sum)

print "\nDistributed Population (p0_4_w_athome) = " + str(int(round(pop,0)))

#--------------------------------------------------------------------------------------------------#
print "\n--------------------------------------------------------------------------"

print "\nOutput Data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print output_bez + analysis_no + " (polygon_fc)"
print output_name_kiga + " (polygon_fc)"

print "\nOutput Fields in " + output_bkm2 + analysis_no + ":"
print "p0_4_w_kiga"
print "SUM_p0_4_w_athome"

print "\nOutput Fields in " + output_bez + analysis_no + ":"
print "SUM_age0_4_kiga"

print "\nOutput Fields in " + output_name_kiga + ":"
print "age0_4_athome"
print "age0_4_kiga"


end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)


####################################################################################################
print "\n##########################################################################\n\n"
```

```
#-----------------------------------------------------------------------------------------------#
#-----------Distribute Population (pflichtschule, ahs, andere hoehere, berufsschule) to bkm--------#
#-----------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "--------------------------------------------------------------------------"
print "----Distr Pop (pflichtschule, ahs, andere hoehere, berufsschule) to bkm---"
print "--------------------------------------------------------------------------\n"

input_fc_schueler = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_fc2_schueler = wsInput + os.sep + point_schools # ogd_schule_wien
input_table_schueler = gdb + os.sep + table_schueler # schueler

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print point_schools + " (point_fc)"
print table_schueler + " (table)"

#------------------------------------------ANALYSIS 1--------------------------------------------#
#---------------------------------Join ogd_schule_wien to bkm-------------------------------------#
print "\n---------------------------------ANALYSIS---------------------------------"
print "----Distr Pop (pflichtschule, ahs, andere hoehere, berufsschule) to bkm---\n"

# Process: Spatial Join
print "Spatial Join: ogd_schule_wien to bkm"
output_spatialjoin = wsTemp + os.sep + "output_spatialjoin_schule"
arcpy.SpatialJoin_analysis(input_fc_schueler, input_fc2_schueler, output_spatialjoin, "JOIN_ONE_TO_
ONE", "KEEP_ALL", "", "INTERSECT", "0.5 Meters")

print "Join Fields"
del_list = [TYP_GEN, BEZ_schule]
arcpy.DeleteField_management(input_fc_schueler, del_list)
arcpy.JoinField_management(input_fc_schueler, ID_bkm, output_spatialjoin, ID_bkm, "{}; {}".format(-
TYP_GEN, BEZ_schule))

# Process: Add Fields
print "Adding Fields"
arcpy.DeleteField_management(input_fc_schueler, ["w_factor_total_pflichtschule", "w_factor_percent_
pflichtschule", "p5_14_w_pflichtschule"])
arcpy.AddField_management(input_fc_schueler, "w_factor_total_pflichtschule", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "w_factor_percent_pflichtschule", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "p5_14_w_pflichtschule", "DOUBLE")

arcpy.DeleteField_management(input_fc_schueler, ["w_factor_total_ahs", "w_factor_percent_ahs",
"p10_19_w_ahs"])
arcpy.AddField_management(input_fc_schueler, "w_factor_total_ahs", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "w_factor_percent_ahs", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "p10_19_w_ahs", "DOUBLE")
arcpy.DeleteField_management(input_fc_schueler, ["w_factor_total_andere_hoehere", "w_factor_per-
cent_andere_hoehere", "p15_19_w_andere_hoehere"])
arcpy.AddField_management(input_fc_schueler, "w_factor_total_andere_hoehere", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "w_factor_percent_andere_hoehere", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "p15_19_w_andere_hoehere", "DOUBLE")

arcpy.DeleteField_management(input_fc_schueler, ["w_factor_total_berufsschule", "w_factor_percent_
berufsschule", "p15_19_w_berufsschule_schule"])
arcpy.AddField_management(input_fc_schueler, "w_factor_total_berufsschule", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "w_factor_percent_berufsschule", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "p15_19_w_berufsschule_schule", "DOUBLE")

arcpy.DeleteField_management(input_fc_schueler, ["w_factor_total_berufsschule_betrieb", "w_factor_
percent_berufsschule_betrieb", "p15_19_w_berufsschule_betrieb"])
arcpy.AddField_management(input_fc_schueler, "w_factor_total_berufsschule_betrieb", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "w_factor_percent_berufsschule_betrieb", "DOUBLE")
arcpy.AddField_management(input_fc_schueler, "p15_19_w_berufsschule_betrieb", "DOUBLE")

#------------------------------------------ANALYSIS 2---------------------------------------------#
#------------------------------Distribute pflichtschueler to Buildings------------------------------#

print "\nDistribute pflichtschueler to bkm for the whole of Vienna"
# Selecting Features (Pflichtschule)
arcpy.MakeFeatureLayer_management (input_fc_schueler, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".format(T-
YP_GEN, NAME_ogd_pflichtschule))

# Process: Summary Statistics (getting total w_factor of all buildings (Pflichtschule) for the whole
of Vienna)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value (w_factor SUM)
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get Field Value (Total number of population Pflichtschule)
SC = arcpy.SearchCursor(input_table_schueler)
field_sum = NAME_pflichtschule
for row in SC:
    pop_pflichtschule = row.getValue(field_sum)

# Process: Calculate Fields
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_pflichtschule", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_pflichtschule", "[{}]/[w_fac-
tor_total_pflichtschule]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p5_14_w_pflichtschule", "{} * [w_factor_per-
cent_pflichtschule]".format(pop_pflichtschule))
```

```
#------------------------------------------ANALYSIS 3--------------------------------------------#
#-------------------------------Distribute population of ahs to Buildings-------------------------#

print "Distribute population of ahs to bkm for the whole of Vienna"
# Selecting Features (ahs)
arcpy.MakeFeatureLayer_management (input_fc_schueler, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".format(T-
YP_GEN, NAME_ogd_ahs))

# Process: Summary Statistics (getting total w_factor of all buildings (AHS) for the whole of Vien-
na)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value (w_factor SUM)
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get Field Value (Total number of population AHS)
SC = arcpy.SearchCursor(input_table_schueler)
field_sum = NAME_ahs
for row in SC:
    pop_ahs = row.getValue(field_sum)

# Process: Calculate Fields
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_ahs", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_ahs", "[{}]/[w_factor_total_
ahs]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p10_19_w_ahs", "{} * [w_factor_percent_ahs]".
format(pop_ahs))

#------------------------------------------ANALYSIS 4--------------------------------------------#
#--------------------------Distribute population of andere hoehere to Buildings-------------------#

print "Distribute population of andere hoehere to bkm for the whole of Vienna"
# Selecting Features (andere hoehere)
arcpy.MakeFeatureLayer_management (input_fc_schueler, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".format(T-
YP_GEN, NAME_ogd_andere_hoehere))

# Process: Summary Statistics (getting total w_factor of all buildings (andere hoehere) for the
whole of Vienna)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value (w_factor SUM)
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)
```

```
# Get Field Value (Total number of population andere hoehere)
SC = arcpy.SearchCursor(input_table_schueler)
field_sum = NAME_andere_hoehere
for row in SC:
    pop_andere_hoehere = row.getValue(field_sum)

# Process: Calculate Fields
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_andere_hoehere", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_andere_hoehere", "[{}]/[w_
factor_total_andere_hoehere]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p15_19_w_andere_hoehere", "{} * [w_factor_per-
cent_andere_hoehere]".format(pop_andere_hoehere))

#------------------------------------------ANALYSIS 5--------------------------------------------#
#---------------------------Distribute population of berufsschule to Buildings--------------------#

print "Distribute population of berufsschule to bkm (schule) for the whole of Vienna"
# Selecting Features (berufsschule)
arcpy.MakeFeatureLayer_management (input_fc_schueler, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".format(T-
YP_GEN, NAME_ogd_berufsschule))

# Process: Summary Statistics (getting total w_factor of all buildings (berufsschule) for the whole
of Vienna)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value (w_factor SUM)
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get Field Value (Total number of population berufsschule)
SC = arcpy.SearchCursor(input_table_schueler)
field_sum = NAME_berufsschule
for row in SC:
    pop_berufsschule = row.getValue(field_sum)
    pop_berufsschule_schule = pop_berufsschule * 0.2
    pop_berufsschule_betrieb = pop_berufsschule * 0.8

# Process: Calculate Fields
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_berufsschule", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_berufsschule", "[{}]/[w_fac-
tor_total_berufsschule]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p15_19_w_berufsschule_schule", "{} * [w_fac-
tor_percent_berufsschule]".format(pop_berufsschule_schule))

# Distribute population of berufsschule to bkm (betrieb) for the whole of Vienna
print "Distribute population of berufsschule to bkm (betrieb) for the whole of Vienna"

# Selecting Features (Gewerbegebäude)
```

```
arcpy.MakeFeatureLayer_management (input_fc_schueler, "input_fc_f_layer")

use1 = trade
use2 = industry
use3 = social

arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                        "{0} in ('{1}', '{2}') or {0} = '{3}' and {4} IS NOT NULL".
format(USE_bkm, use1, use2, use3, ID_hospital))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC2 = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row2 in SC2:
    value = row2.getValue(field_sum)

# Process: Calculate Fields
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_berufsschule_betrieb", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_berufsschule_betrieb", "[{}]/
[w_factor_total_berufsschule_betrieb]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p15_19_w_berufsschule_betrieb", "{} * [w_fac-
tor_percent_berufsschule_betrieb]".format(pop_berufsschule_betrieb))

#---------------------------------------------------------------------------------------------------#

# Statistics
arcpy.Statistics_analysis(input_fc_schueler, table_temp, "p5_14_w_pflichtschule SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p5_14_w_pflichtschule"
for row in SC:
    sum1 = row.getValue(field_sum)

print "\nStatistics:"
print "Total number of population Pflichtschule = " + str(int(round(pop_pflichtschule,0)))
print "Distributed Population (p5_14_w_pflichtschule) = " + str(int(round(sum1,0)))

arcpy.Statistics_analysis(input_fc_schueler, table_temp, "p10_19_w_ahs SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p10_19_w_ahs"
for row in SC:
    sum2 = row.getValue(field_sum)

print "\nTotal number of population AHS = " + str(int(round(pop_ahs,0)))
print "Distributed Population (p10_19_w_ahs) = " + str(int(round(sum2,0)))

arcpy.Statistics_analysis(input_fc_schueler, table_temp, "p15_19_w_andere_hoehere SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p15_19_w_andere_hoehere"
```

```
for row in SC:
    sum3 = row.getValue(field_sum)

print “\nTotal number of population andere hoehere = “ + str(int(round(pop_andere_hoehere,0)))
print “Distributed Population (p15_19_w_andere_hoehere) = “ + str(int(round(sum3,0)))

arcpy.Statistics_analysis(input_fc_schueler, table_temp, “p15_19_w_berufsschule_schule SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_p15_19_w_berufsschule_schule”
for row in SC:
    sum4 = row.getValue(field_sum)

arcpy.Statistics_analysis(input_fc_schueler, table_temp, “p15_19_w_berufsschule_betrieb SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_p15_19_w_berufsschule_betrieb”
for row in SC:
    sum5 = row.getValue(field_sum)

print “\nTotal number of population berufsschule = “ + str(int(round(pop_berufsschule,0)))
print “Population berufsschule in schule (20%) = “ + str(int(round(pop_berufsschule_schule,0)))
print “Population berufsschule in betrieb (80%) = “ + str(int(round(pop_berufsschule_betrieb,0)))
print “Distributed Population (p15_19_w_berufsschule_schule) = “ + str(int(round(sum4,0)))
print “Distributed Population (p15_19_w_berufsschule_betrieb) = “ + str(int(round(sum5,0)))

#-------------------------------------------------------------------------------------------------#
print “\n--------------------------------------------------------------------------”

print “\nOutput Data:”
print output_bkm2 + analysis_no + “ (polygon_fc)”

print “\nOutput Fields in “ + output_bkm2 + analysis_no + “:”
print “p5_14_w_pflichtschule”
print “p10_19_w_ahs”
print “p15_19_w_andere_hoehere”
print “p15_19_w_berufsschule_schule”


end = datetime.datetime.now()

print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start)


###################################################################################################
print “\n##########################################################################\n\n”


#-------------------------------------------------------------------------------------------------#
#-----------------------------------Distribute University Students--------------------------------#
#-------------------------------------------------------------------------------------------------#
```

```
import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "--------------------------------------------------------------------------"
print "----------------------Distribute University Students----------------------"
print "--------------------------------------------------------------------------\n"

input_fc_univ_stud = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_table_univ_stud = gdb + os.sep + table_students_univ # studenten

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print table_students_univ + " (table)"

#------------------------------------------ANALYSIS 1--------------------------------------------#
#----------------------------Distribute Students (*0.5) to University Buildings------------------#
print "\n--------------------------------ANALYSIS 1---------------------------------"
print "------------Distribute Students (*0.5) to University Buildings------------\n"

# Add Fields
arcpy.DeleteField_management(input_fc_univ_stud, ["w_factor_total_univ", "w_factor_percent_univ", 
"p20_29_studenten_univ"])
arcpy.AddField_management(input_fc_univ_stud, "w_factor_total_univ", "DOUBLE")
arcpy.AddField_management(input_fc_univ_stud, "w_factor_percent_univ", "DOUBLE")
arcpy.AddField_management(input_fc_univ_stud, "p20_29_studenten_univ", "DOUBLE")

# Calculate p20_29_studenten
SC = arcpy.SearchCursor(input_table_univ_stud)
field_name =  NAME_univ_stud
field_pop = POP_20_29_studenten
for row in SC:
    name = row.getValue(field_name)
    pop_stud = row.getValue(field_pop)
    print "Number of Students (" + name + ") = " + str(int(round(pop_stud,0)))

print "\n"

SC = arcpy.SearchCursor(input_table_univ_stud)
field_name =  NAME_univ_stud
field_pop = POP_20_29_studenten
for row in SC:
    name = row.getValue(field_name)
    pop_stud = row.getValue(field_pop)

    arcpy.MakeFeatureLayer_management (input_fc_univ_stud, "input_fc_f_layer")
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(NAME_univ2, name))

    # Process: Summary Statistics
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))
    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_{}".format(w_factor_bkm)
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields (Students * 0.5)
    print "Calculating Fields for " + name
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_univ", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_univ", "[{}]/[w_factor_
total_univ]".format(w_factor_bkm))
    arcpy.CalculateField_management("input_fc_f_layer", "p20_29_studenten_univ", "{} * [w_factor_
percent_univ] * 0.5".format(pop_stud))

# Statistics
arcpy.Statistics_analysis(input_table_univ_stud, table_temp, "{} SUM".format(POP_20_29_studenten))
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_20_29_studenten)
for row in SC:
    pop = row.getValue(field_sum)
    pop_univ = pop * 0.5
    pop_athome = pop * 0.5
    print "\nTotal number of students = " + str(int(round(pop,0)))
    print "Number of students at university (0.5) = " + str(int(round(pop_univ,0)))
    print "Number of students at home (0.5) = " + str(int(round(pop_athome,0)))

arcpy.Statistics_analysis(input_fc_univ_stud, table_temp, "p20_29_studenten_univ SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p20_29_studenten_univ"
for row in SC:
    distr_stud = row.getValue(field_sum)
    print "Number of distributed students = " + str(int(round(distr_stud,0)))

#------------------------------------------ANALYSIS 2--------------------------------------------#
#----------------------------Distribute p20_29_studenten_athome to bkm-------------------------#
print "\n--------------------------------ANALYSIS 2--------------------------------"
print "-----------------Distribute p20_29_studenten_athome to bkm----------------\n"

# Add Fields
arcpy.DeleteField_management(input_fc_univ_stud, ["w_factor_total_stud_athome", "w_factor_percent_
stud_athome", "p20_29_studenten_athome"])
arcpy.AddField_management(input_fc_univ_stud, "w_factor_total_stud_athome", "DOUBLE")
arcpy.AddField_management(input_fc_univ_stud, "w_factor_percent_stud_athome", "DOUBLE")
arcpy.AddField_management(input_fc_univ_stud, "p20_29_studenten_athome", "DOUBLE")

# Selecting Features (Wohngebäude)
arcpy.MakeFeatureLayer_management (input_fc_univ_stud, "input_fc_f_layer")

use = living
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(USE_bkm, use))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Process: Calculate Fields
print "Calculating Fields"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_stud_athome", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_stud_athome", "[{}]/[w_fac-
tor_total_stud_athome]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p20_29_studenten_athome", "{} * [w_factor_per-
cent_stud_athome]".format(pop_athome))

# Check distributed population
arcpy.Statistics_analysis(input_fc_univ_stud, table_temp, "p20_29_studenten_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p20_29_studenten_athome"
for row in SC:
    pop = row.getValue(field_sum)

print "\nNumber of p20_29_studenten_athome = " + str(int(round(pop_athome,0)))
print "Distributed Population (p20_29_studenten_athome) = " + str(int(round(pop,0)))

#---------------------------------------------------------------------------------------------------#
print "\n--------------------------------------------------------------------------"

print "\nOutput Data:"
print output_bkm2 + analysis_no + " (polygon_fc)"

print "\nOutput Fields in " + output_bkm2 + analysis_no + ":"
print "p20_29_studenten_univ"
print "p20_29_studenten_athome"


end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)


###################################################################################################
print "\n#################################################################################\n\n"
```

```
#------------------------------------------------------------------------------------------------#
#-------------------------------------Distribute Retirees to bkm---------------------------------#
#------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print 'start run: %0.19s\n' % (start)

print "--------------------------------------------------------------------------"
print "------------------------Distribute Retirees to bkm------------------------"
print "--------------------------------------------------------------------------\n"

input_fc_retirees = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_fc2_retirees = wsOutput + os.sep + output_zbez + analysis_no # zbez_output
input_fc3_retirees = gdb + os.sep + point_ret_home # ogd_wohnpflege

print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print output_zbez + analysis_no + " (polygon_fc)"
print point_ret_home + " (point_fc)"

#------------------------------------------ANALYSIS 1--------------------------------------------#
#---------------------Calculate Number of Retirees in care homes and at home---------------------#
print "\n-------------------------------ANALYSIS 1--------------------------------"
print "----------Calculate Number of Retirees in care homes and at home----------\n"

# Calculate Retirees in care homes
print "Calculate Retirees in care homes"
arcpy.DeleteField_management(input_fc2_retirees, ["age65_plus", "age65_69_pflege", "age70_74_pflege",
"age75_79_pflege", "age80_84_pflege", "age85_89_pflege", "age90_plus_pflege", "age65_plus_pflege"])

arcpy.AddField_management(input_fc2_retirees, "age65_plus", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age65_69_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age70_74_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age75_79_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age80_84_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age85_89_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age90_plus_pflege", "DOUBLE")
arcpy.AddField_management(input_fc2_retirees, "age65_plus_pflege", "DOUBLE")

arcpy.CalculateField_management(input_fc2_retirees, "age65_plus", "[{}]+[{}]+[{}]+[{}]+[{}]+[{}]".
format(POP_65_69, POP_70_74, POP_75_79, POP_80_84, POP_85_89, POP_90plus))
arcpy.CalculateField_management(input_fc2_retirees, "age65_69_pflege", "[{}] * 0.03 * 0.291".for-
mat(POP_65_69))
arcpy.CalculateField_management(input_fc2_retirees, "age70_74_pflege", "[{}] * 0.05 * 0.291".for-
mat(POP_70_74))
arcpy.CalculateField_management(input_fc2_retirees, "age75_79_pflege", "[{}] * 0.098 * 0.291".for-
mat(POP_75_79))
arcpy.CalculateField_management(input_fc2_retirees, "age80_84_pflege", "[{}] * 0.21 * 0.291".for-
mat(POP_80_84))
arcpy.CalculateField_management(input_fc2_retirees, "age85_89_pflege", "[{}] * 0.382 * 0.291".for-
mat(POP_85_89))
arcpy.CalculateField_management(input_fc2_retirees, "age90_plus_pflege", "[{}] * 0.644 * 0.291".
format(POP_90plus))
arcpy.CalculateField_management(input_fc2_retirees, "age65_plus_pflege",
                                "[age65_69_pflege]+[age70_74_pflege]+[age75_79_pflege]+[age80_84_
pflege]+[age85_89_pflege]+[age90_plus_pflege]")

# Calculate Retirees at home
print "Calculate Retirees at home"
arcpy.DeleteField_management(input_fc2_retirees, ["age65_plus_athome"])
arcpy.AddField_management(input_fc2_retirees, "age65_plus_athome", "DOUBLE")
arcpy.CalculateField_management(input_fc2_retirees, "age65_plus_athome", "[age65_plus]-[age65_plus_
pflege]")

# Statistics
print "\nStatistics:"
arcpy.Statistics_analysis(input_fc2_retirees, table_temp, "age65_plus SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_age65_plus"
for row in SC:
    sum = row.getValue(field_sum)
    print "SUM_age65_plus = " + str(int(round(sum,0)))

arcpy.Statistics_analysis(input_fc2_retirees, table_temp, "age65_plus_pflege SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_age65_plus_pflege"
for row in SC:
    sum_pflege = row.getValue(field_sum)
    print "SUM_age65_plus_pflege = " + str(int(round(sum_pflege,0)))

arcpy.Statistics_analysis(input_fc2_retirees, table_temp, "age65_plus_athome SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_age65_plus_athome"
for row in SC:
    sum_athome = row.getValue(field_sum)
    print "SUM_age65_plus_athome = " + str(int(round(sum_athome,0)))

#-------------------------------------------ANALYSIS 2---------------------------------------------#
#---------------------Distribute SUM_age65_plus_pflege to bkm (ogd_wohnpflege)----------------------#
print "\n--------------------------------ANALYSIS 2--------------------------------"
print "---------Distribute SUM_age65_plus_pflege to bkm (ogd_wohnpflege)---------\n"

# Process: Spatial Join
print "Spatial Join: ogd_wohnpflege to bkm"
output_spatialjoin = wsTemp + os.sep + "output_spatialjoin_wohnpflege"
arcpy.SpatialJoin_analysis(input_fc_retirees, input_fc3_retirees, output_spatialjoin, "JOIN_ONE_TO_
ONE", "KEEP_ALL", "", "INTERSECT", "0.5 Meters")

del_list = [TYPE_pflege, BEZ_pflege]
arcpy.DeleteField_management(input_fc_retirees, del_list)
arcpy.JoinField_management(input_fc_retirees, ID_bkm, output_spatialjoin, ID_bkm, "{}; {}".format(-
TYPE_pflege, BEZ_pflege))

# Process: Add Fields
print "Adding Fields"
arcpy.DeleteField_management(input_fc_retirees, ["w_factor_total_wohnpflege", "w_factor_percent_
wohnpflege", "p65_plus_wohnpflege"])
arcpy.AddField_management(input_fc_retirees, "w_factor_total_wohnpflege", "DOUBLE")
arcpy.AddField_management(input_fc_retirees, "w_factor_percent_wohnpflege", "DOUBLE")
arcpy.AddField_management(input_fc_retirees, "p65_plus_wohnpflege", "DOUBLE")

# Selecting Features (kiga) per BEZ
arcpy.MakeFeatureLayer_management (input_fc_retirees, "input_fc_f_layer")
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} IS NOT NULL".for-
mat(TYPE_pflege))

# Process: Summary Statistics (getting total w_factor of all buildings per BEZ)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC2 = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row2 in SC2:
    value = row2.getValue(field_sum)

# Process: Calculate Fields
print "Distribute p65_plus_wohnpflege to bkm (ogd_wohnpflege)"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_wohnpflege", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_wohnpflege", "[{}]/[w_factor_
total_wohnpflege]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p65_plus_wohnpflege", "{} * [w_factor_percent_
wohnpflege]".format(sum_pflege))

# Statistics
arcpy.Statistics_analysis(input_fc_retirees, table_temp, "p65_plus_wohnpflege SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_p65_plus_wohnpflege"
for row in SC:
    sum = row.getValue(field_sum)
    print "\nDistributed Population (p65_plus_wohnpflege) = " + str(int(round(sum,0)))

#------------------------------------------ANALYSIS 3--------------------------------------------#
#-----------------------Distribute SUM_age65_plus_athome to bkm per ZBEZ--------------------------#
print "\n-------------------------------ANALYSIS 3--------------------------------"
print "-------------Distribute SUM_age65_plus_athome to bkm per ZBEZ-------------\n"

# Process: Identity (get buildings per ZBEZ)
print "Identity (bkm, zbez)"
output_identity = wsTemp + os.sep + "output_identity4"
arcpy.Identity_analysis(input_fc_retirees, input_fc2_retirees, output_identity)
arcpy.AddField_management(output_identity, "w_factor3", "DOUBLE")
arcpy.CalculateField_management(output_identity, "w_factor3", w_factor)
```

```
# Add Fields
arcpy.AddField_management(output_identity, "w_factor_total_ret_athome", "DOUBLE")
arcpy.AddField_management(output_identity, "w_factor_percent_ret_athome", "DOUBLE")
arcpy.AddField_management(output_identity, "p65_plus_ret_athome", "DOUBLE")

print "\nDistribute p65_plus_ret_athome to Wohngebäude per ZBEZ:"
SC = arcpy.SearchCursor(input_fc2_retirees)
field_nr = OID
field_id = ID_zbez
field_pop = "age65_plus_athome"
for row in SC:
    f_nr = row.getValue(field_nr)
    f_id = row.getValue(field_id)
    pop_athome = row.getValue(field_pop)

    # Selecting Features (Wohngebäude) per ZBEZ
    arcpy.MakeFeatureLayer_management (output_identity, "input_fc_f_layer")

    use = living
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION",
                                            "{} = {} and {} = '{}'".format(ID_zbez, f_id, USE_bkm,
use))

    # Process: Summary Statistics (getting total w_factor of all buildings per ZBEZ)
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor3 SUM")

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_w_factor3"
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields
    print "Calculating Fields for ZBEZ " + str(f_id) + "(" + str(f_nr) + ")"
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_ret_athome", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_ret_athome", "[w_fac-
tor3]/[w_factor_total_ret_athome]")
    arcpy.CalculateField_management("input_fc_f_layer", "p65_plus_ret_athome", "{} * [w_factor_per-
cent_ret_athome]".format(pop_athome))

print "\nDissolve output_identity"
output_diss_identity = wsTemp + os.sep + "output_diss_identity3"
arcpy.Dissolve_management(output_identity, output_diss_identity, ID_bkm, "p65_plus_ret_athome SUM",
"MULTI_PART", "DISSOLVE_LINES")

print "Join p65_plus_ret_athome to bkm_output2"
arcpy.DeleteField_management(input_fc_retirees, ["SUM_p65_plus_ret_athome"])
arcpy.JoinField_management(input_fc_retirees, ID_bkm, output_diss_identity, ID_bkm, "SUM_p65_plus_
ret_athome")
```

```
# Check distributed population
arcpy.Statistics_analysis(input_fc_retirees, table_temp, “SUM_p65_plus_ret_athome SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_SUM_p65_plus_ret_athome”
for row in SC:
    pop = row.getValue(field_sum)

print “\nSUM_age65_plus_athome = “ + str(int(round(sum_athome,0)))
print “Distributed Population (SUM_p65_plus_ret_athome) = “ + str(int(round(pop,0)))

#--------------------------------------------------------------------------------------------------#
print “\n--------------------------------------------------------------------------”

print “\nOutput Data:”
print output_bkm2 + analysis_no + “ (polygon_fc)”
print output_zbez + analysis_no + “ (polygon_fc)”

print “\nOutput Fields in “ + output_bkm2 + analysis_no + “:”
print “p65_plus_wohnpflege”
print “SUM_p65_plus_ret_athome”

print “\nOutput Fields in “ + output_zbez + analysis_no + “:”
print “age65_plus_pflege”
print “age65_plus_athome”


end = datetime.datetime.now()

print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start)


####################################################################################################
print “\n##########################################################################\n\n”


#--------------------------------------------------------------------------------------------------#
#---------------------------------Distribute Patients to Hospitals---------------------------------#
#--------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print ‘start run: %0.19s\n’ % (start)

print “--------------------------------------------------------------------------”
print “---------------------Distribute Patients to Hospitals---------------------”
print “--------------------------------------------------------------------------\n”

input_fc_patients = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_table_patients = gdb + os.sep + table_patients # patients
```

```
print "Input data:"
print output_bkm2 + analysis_no + " (polygon_fc)"
print table_patients + " (table)"

#--------------------------------------------ANALYSIS--------------------------------------------#
#-----------------------------------Distribute Patients to Hospitals------------------------------#
print "\n---------------------------------ANALYSIS---------------------------------"
print "----------------------Distribute Patients to Hospitals--------------------\n"

# Add Fields
arcpy.DeleteField_management(input_fc_patients, ["w_factor_total2", "w_factor_percent2", "pa-
tients"])
arcpy.AddField_management(input_fc_patients, "w_factor_total2", "DOUBLE")
arcpy.AddField_management(input_fc_patients, "w_factor_percent2", "DOUBLE")
arcpy.AddField_management(input_fc_patients, "patients", "DOUBLE")

# Calculate patients
SC = arcpy.SearchCursor(input_table_patients)
field_id = ID_count3
field_name =  ID_hospital_pat
field_pop = POP_patients
for row in SC:
    f_id = row.getValue(field_id)
    name = row.getValue(field_name)
    pop_patients = row.getValue(field_pop)

    arcpy.MakeFeatureLayer_management (input_fc_patients, "input_fc_f_layer")
    arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(ID_hospital, name))

        # Process: Summary Statistics
    arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "w_factor SUM".format(w_factor_bkm))

    # Get Field Value
    SC2 = arcpy.SearchCursor(table_temp)
    field_sum = "SUM_{}".format(w_factor_bkm)
    for row2 in SC2:
        value = row2.getValue(field_sum)

    # Process: Calculate Fields (w_factor_percent2 already calculated in script "Distribute Working
Population (Hospitals, Univ, vhs)")
    print "Calculating Fields for " + str(name) + " (" + str(int(round(f_id,0))) + ")"
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total2", value)
    arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent2", "[{}]/[w_factor_to-
tal2]".format(w_factor_bkm))
    arcpy.CalculateField_management("input_fc_f_layer", "patients", "{} * [w_factor_percent2]".
format(pop_patients))

# Statistics
print "\nStatistics:"
arcpy.Statistics_analysis(input_table_patients, table_temp, "{} SUM".format(POP_patients))
```

```
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_{}”.format(POP_patients)
for row in SC:
    pop = row.getValue(field_sum)
    print “Number of patients = “ + str(int(round(pop,0)))

arcpy.Statistics_analysis(input_fc_patients, table_temp, “patients SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_patients”
for row in SC:
    distr_pop = row.getValue(field_sum)
    print “Number of distributed patients = “ + str(int(round(distr_pop,0)))

#--------------------------------------------------------------------------------------------------#
print “\n-------------------------------------------------------------------------”

print “\nOutput Data:”
print output_bkm2 + analysis_no + “ (polygon_fc)”

print “\nOutput Fields in “ + output_bkm2 + analysis_no + “:”
print “patients”


end = datetime.datetime.now()

print ‘\nfinished run: %0.19s\n’ % (end),
print ‘runtime: %0.7s\n’ % (end-start)


####################################################################################################
print “\n##########################################################################\n\n”


#--------------------------------------------------------------------------------------------------#
#----------------------------Create bkm_output3 and calculate total population---------------------#
#--------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print ‘start run: %0.19s\n’ % (start)

print “---------------------------------------------------------------------------”
print “------------Create bkm_output3 and calculate total population-------------”
print “--------------------------------------------------------------------------\n”

input_fc_total_pop = wsInput + os.sep + input_bkm # bkm
input_fc2_total_pop = wsOutput + os.sep + output_bkm2 + analysis_no # bkm_output2
input_fc3_total_pop = wsInput + os.sep + input_zbez # zbez_etrs
output_fc_total_pop = wsOutput2 + os.sep + output_name_total_pop # bkm_output3
input_table_patients = gdb + os.sep + table_patients # patients
input_table_wrlinien = gdb + os.sep + table_wrlinien_empl # besch_wr_linien
```

```
input_table_pendlersaldo = gdb + os.sep + table_pendlersaldo # pendlersaldo

print "Input data:"
print input_bkm + " (polygon_fc)"
print output_bkm2 + analysis_no + " (polygon_fc)"
print output_name_total_pop + " (polygon_fc)"
print table_patients + " (table)"
print table_wrlinien_empl + " (table)"
print table_pendlersaldo + " (table)"

#-------------------------------------------ANALYSIS---------------------------------------------#
#-------------------------Create bkm_output3 and calculate total population----------------------#
print "\n--------------------------------ANALYSIS--------------------------------"
print "------------Create bkm_output3 and calculate total population-------------\n"

# Create bkm_output3
print "Create bkm_output3"
arcpy.CopyFeatures_management(input_fc_total_pop, output_fc_total_pop)

# Join Fields
f0a = "p15_64_w_hosp"
f0b = "p15_64_w_univ"
f0c = "p15_64_w_vhs"
f0 = "p15_64_w_hosp_athome"
f1 = "p15_64_w_hosp_univ_vhs"
f2 = "SUM_{}".format(POP_15_64_firmenreg)
f3 = "p15_64_w_wrlinien_athome"
f4 = "SUM_p15_64_w_remaining_working_pop"
f5 = "p0_4_w_kiga"
f6 = "SUM_p0_4_w_athome"
f7 = "p15_64_w_unemployed"
f8 = "p5_14_w_pflichtschule"
f9 = "p10_19_w_ahs"
f10 = "p15_19_w_andere_hoehere"
f11 = "p15_19_w_berufsschule_schule"
f12 = "p15_19_w_berufsschule_betrieb"
f13 = "p20_29_studenten_univ"
f14 = "p20_29_studenten_athome"
f15 = "p65_plus_wohnpflege"
f16 = "SUM_p65_plus_ret_athome"
f17 = "patients"

join_list = [f0a, f0b, f0c, f0, f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15,
f16, f17]

print "Join Fields"
arcpy.JoinField_management(output_fc_total_pop, "code_bkm", input_fc2_total_pop, "code_bkm", join_
list)

# Replace <Null> with 0
print "Replace <Null> with 0"
```

```
expression = “myCalc(!p15_64_w_hosp_univ_vhs!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc_total_pop, “p15_64_w_hosp_univ_vhs”, expression, “PY-
THON_9.3”, codeblock)

expression = “myCalc(!SUM_{}!)”.format(POP_15_64_firmenreg)
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc_total_pop, “SUM_{}”.format(POP_15_64_firmenreg), expres-
sion, “PYTHON_9.3”, codeblock)

expression = “myCalc(!p15_64_w_wrlinien_athome!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc_total_pop, “p15_64_w_wrlinien_athome”, expression, “PY-
THON_9.3”, codeblock)

expression = “myCalc(!SUM_p15_64_w_remaining_working_pop!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc_total_pop, “SUM_p15_64_w_remaining_working_pop”, expres-
sion, “PYTHON_9.3”, codeblock)

expression = “myCalc(!p0_4_w_kiga!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc_total_pop, “p0_4_w_kiga”, expression, “PYTHON_9.3”, code-
block)

expression = “myCalc(!SUM_p0_4_w_athome!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
```

```
arcpy.CalculateField_management(output_fc_total_pop, "SUM_p0_4_w_athome", expression, "PYTHON_9.3",
codeblock)

expression = "myCalc(!p15_64_w_unemployed!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p15_64_w_unemployed", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!p5_14_w_pflichtschule!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p5_14_w_pflichtschule", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!p10_19_w_ahs!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p10_19_w_ahs", expression, "PYTHON_9.3",
codeblock)

expression = "myCalc(!p15_19_w_andere_hoehere!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p15_19_w_andere_hoehere", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!p15_19_w_berufsschule_schule!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p15_19_w_berufsschule_schule", expression,
"PYTHON_9.3", codeblock)

expression = "myCalc(!p15_19_w_berufsschule_betrieb!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p15_19_w_berufsschule_betrieb", expression,
"PYTHON_9.3", codeblock)

expression = "myCalc(!p20_29_studenten_univ!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p20_29_studenten_univ", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!p20_29_studenten_athome!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p20_29_studenten_athome", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!p65_plus_wohnpflege!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p65_plus_wohnpflege", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!SUM_p65_plus_ret_athome!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "SUM_p65_plus_ret_athome", expression, "PY-
THON_9.3", codeblock)

expression = "myCalc(!patients!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "patients", expression, "PYTHON_9.3", code-
block)
```

```
# Calculate total population (patients in hospitals not accounted) (no rest_bev)
print "Calculate total population (patients unaccounted)"
arcpy.AddField_management(output_fc_total_pop, "population_vienna_no_patients", "DOUBLE")
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna_no_patients",
    "[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]+[{}]".for-
mat(f0, f1, f2, f3, f4, f5,
       f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16))

# Check calculated population (patients in hospitals not accounted)
arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "population_vienna_no_patients SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_population_vienna_no_patients"
for row in SC:
    population = row.getValue(field_sum)

print "\nTotal Population (patients not accounted) = " + str(int(round(population,0)))

#--------------------------Calculate p5_64_w_rest_bev & total population--------------------------#

# Calculate p5_64_w_rest_bev
print "\nCalculate p5_64_w_rest_bev"
arcpy.AddField_management(output_fc_total_pop, "w_factor_total_rest_bev", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "w_factor_percent_rest_bev", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "p5_64_w_rest_bev", "DOUBLE")

# Selecting Features (Wohngebäude)
arcpy.MakeFeatureLayer_management (output_fc_total_pop, "input_fc_f_layer")

use = living
arcpy.SelectLayerByAttribute_management("input_fc_f_layer", "NEW_SELECTION", "{} = '{}'".for-
mat(USE_bkm, use))

# Process: Summary Statistics (getting total w_factor of all buildings)
arcpy.Statistics_analysis("input_fc_f_layer", table_temp, "{} SUM".format(w_factor_bkm))

# Get Field Value
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(w_factor_bkm)
for row in SC:
    value = row.getValue(field_sum)

# Get rest_bev
arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "population_vienna_no_patients SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_population_vienna_no_patients"
for row in SC:
    population = row.getValue(field_sum)

SC = arcpy.SearchCursor(input_table_wrlinien)
field = POP_15_64_wrlinien
for row in SC:
    employees_wr_linien = row.getValue(field)
    employees_wr_linien2 = employees_wr_linien/2

pop_plus_wrlinien = population + employees_wr_linien2

arcpy.Statistics_analysis(input_fc3_total_pop, table_temp, "{} SUM".format(POP_total))
SC = arcpy.SearchCursor(table_temp)
field = "SUM_{}".format(POP_total)
for row in SC:
    pop_total = row.getValue(field)

arcpy.Statistics_analysis(input_table_pendlersaldo, table_temp, "{} SUM".format(pendlersaldo_to-
tal))
SC = arcpy.SearchCursor(table_temp)
field = "SUM_{}".format(pendlersaldo_total)
for row in SC:
    pendlersaldo = row.getValue(field)

pop_total_pendler = pop_total + pendlersaldo
rest_bev = pop_total_pendler - pop_plus_wrlinien

# Process: Calculate Fields
print "Calculating Fields for p5_64_w_rest_bev"
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_total_rest_bev", value)
arcpy.CalculateField_management("input_fc_f_layer", "w_factor_percent_rest_bev", "[{}]/[w_factor_
total_rest_bev]".format(w_factor_bkm))
arcpy.CalculateField_management("input_fc_f_layer", "p5_64_w_rest_bev", "{} * [w_factor_percent_
rest_bev]".format(rest_bev))

expression = "myCalc(!population_vienna_no_patients!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna_no_patients", expression,
"PYTHON_9.3", codeblock)

expression = "myCalc(!p5_64_w_rest_bev!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "p5_64_w_rest_bev", expression, "PYTHON_9.3",
codeblock)

# Calculate total population (patients in hospitals not accounted)
print "Calculate total population (patients unaccounted)"
arcpy.AddField_management(output_fc_total_pop, "population_vienna_no_patients_restbev", "DOUBLE")
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna_no_patients_restbev",
"[population_vienna_no_patients]+[p5_64_w_rest_bev]")

# Statistics
arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "p5_64_w_rest_bev SUM")
SC = arcpy.SearchCursor(table_temp)
field = "SUM_p5_64_w_rest_bev"
for row in SC:
    distr_restbev = row.getValue(field)

arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "population_vienna_no_patients_restbev
SUM")
SC = arcpy.SearchCursor(table_temp)
field = "SUM_population_vienna_no_patients_restbev"
for row in SC:
    distr_totalbev = row.getValue(field)

distr_totalbev_inclWrlinien = distr_totalbev + employees_wr_linien2

print "\nTotal Population (+ pendlersaldo) = " + str(int(round(pop_total_pendler,0)))
print "Distributed Population incl. Wr Linien = " + str(int(round(pop_plus_wrlinien,0)))
print "rest_bev = " + str(int(round(rest_bev,0)))
print "Distributed rest_bev = " + str(int(round(distr_restbev,0)))
print "Distributed Total Population = " + str(int(round(distr_totalbev,0)))
print "Distributed Total Population incl. Wr Linien = " + str(int(round(distr_totalbev_inclWrlin-
ien,0))) + " (" + str(int(round(pop_total_pendler,0))) + ")"

#----------------------------------------------------------------------------------------------#

# Subtract Patients from buildings weighted by their number of residents
print "\nAdding Fields"
arcpy.DeleteField_management(output_fc_total_pop, ["w_factor_total_pop", "w_factor_percent_total_
pop", "minus_patients", "population_vienna_temp", "population_vienna"])
arcpy.AddField_management(output_fc_total_pop, "w_factor_total_pop", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "w_factor_percent_total_pop", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "minus_patients", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "population_vienna_temp", "DOUBLE")
arcpy.AddField_management(output_fc_total_pop, "population_vienna", "DOUBLE")

# Process: Summary Statistics
arcpy.Statistics_analysis(input_table_patients, table_temp, "{} SUM".format(POP_patients))

# Get Field Value
SC2 = arcpy.SearchCursor(table_temp)
field_sum = "SUM_{}".format(POP_patients)
for row2 in SC2:
    patients = row2.getValue(field_sum)

# Process: Calculate Fields
print "Calculate total population (patients accounted)"
arcpy.CalculateField_management(output_fc_total_pop, "w_factor_total_pop", distr_totalbev)
arcpy.CalculateField_management(output_fc_total_pop, "w_factor_percent_total_pop", "[population_vi-
enna_no_patients_restbev]/[w_factor_total_pop]")
arcpy.CalculateField_management(output_fc_total_pop, "minus_patients", "{} * [w_factor_percent_to-
tal_pop]".format(patients))
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna_temp", "[population_vien-
na_no_patients_restbev]-[minus_patients]")

expression = "myCalc(!population_vienna_temp!)"
codeblock = """def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)"""
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna_temp", expression, "PY-
THON_9.3", codeblock)
arcpy.CalculateField_management(output_fc_total_pop, "population_vienna", "[population_vienna_
temp]+[patients]")

# Check distributed population
arcpy.Statistics_analysis(output_fc_total_pop, table_temp, "population_vienna SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_population_vienna"
for row in SC:
    population = row.getValue(field_sum)

pop_plus_wrlinien = population + employees_wr_linien2

print "\nDistributed Total Population (patients accounted) = " + str(int(round(population,0)))
print "Distributed Total Population incl. Wr Linien = " + str(int(round(pop_plus_wrlinien,0)))

# Delete Fields
arcpy.DeleteField_management(output_fc_total_pop, ["w_factor_total_pop", "w_factor_percent_total_
pop", "minus_patients", "population_vienna_no_patients",
                                                    "population_vienna_temp"])

arcpy.DeleteField_management(output_fc_total_pop, ["w_factor_total_rest_bev", "w_factor_percent_
rest_bev", "population_vienna_no_patients_restbev"])

#-------------------------------------------------------------------------------------------------#
print "\n---------------------------------------------------------------------------"

print "\nOutput Data:"
print output_name_total_pop + " (polygon_fc)"

print "\nOutput Fields in " + output_name_total_pop + ":"
print "population_vienna"


end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s\n' % (end-start)
```

```
####################################################################################################
print “\n##########################################################################\n\n”


#--------------------------------------------------------------------------------------------------#
#----------------------------------------Aggregate population to raster----------------------------#
#--------------------------------------------------------------------------------------------------#

import datetime
start = datetime.datetime.now()
print ‘start run: %0.19s\n’ % (start)

print “--------------------------------------------------------------------------”
print “-----------------------Aggregate population to raster---------------------”
print “--------------------------------------------------------------------------\n”

import arcpy, os, sys
arcpy.env.overwriteOutput = True

input_fc = wsOutput2 + os.sep + output_bkm3 + analysis_no # bkm_output3
input_fc2 = wsInput + os.sep + input_grid # grid
input_fc3 = wsOutput + os.sep + output_name_wrlinien # public_transportation_buffer
input_table_wrlinien = gdb + os.sep + table_wrlinien_empl
output_fc = wsOutput2 + os.sep + output_name_aggregate # grid_aggregated

print “Input data:”
print output_bkm3 + analysis_no + “ (polygon_fc)”
print input_grid + “ (polygon_fc)”
print output_name_wrlinien + “ (polygon_fc)”

#------------------------------------------ANALYSIS 1----------------------------------------------#
#--------------------------------Aggregate population to raster------------------------------------#
print “\n--------------------------------ANALYSIS 1--------------------------------”
print “------------------------Aggregate population to raster--------------------\n”

# Make Feature Layer
arcpy.MakeFeatureLayer_management(input_fc, “f_layer”, “”, “”,
    “””p15_64_w_hosp p15_64_w_hosp VISIBLE RATIO;
           p15_64_w_hosp_athome p15_64_w_hosp_athome VISIBLE RATIO;
            p15_64_w_univ p15_64_w_univ VISIBLE RATIO;
           p15_64_w_vhs p15_64_w_vhs VISIBLE RATIO;
           p15_64_w_hosp_univ_vhs p15_64_w_hosp_univ_vhs VISIBLE RATIO;
           SUM_{0} SUM_{0} VISIBLE RATIO;
           p15_64_w_wrlinien_athome p15_64_w_wrlinien_athome VISIBLE RATIO;
           SUM_p15_64_w_remaining_working_pop SUM_p15_64_w_remaining_working_pop VISIBLE RATIO;
           p15_64_w_unemployed p15_64_w_unemployed VISIBLE RATIO;
           p0_4_w_kiga p0_4_w_kiga VISIBLE RATIO;
           SUM_p0_4_w_athome SUM_p0_4_w_athome VISIBLE RATIO;
           p5_14_w_pflichtschule p5_14_w_pflichtschule VISIBLE RATIO;
```

```
            p10_19_w_ahs p10_19_w_ahs VISIBLE RATIO;
            p15_19_w_andere_hoehere p15_19_w_andere_hoehere VISIBLE RATIO;
            p15_19_w_berufsschule_schule p15_19_w_berufsschule_schule VISIBLE RATIO;
            p15_19_w_berufsschule_betrieb p15_19_w_berufsschule_betrieb VISIBLE RATIO;
            p20_29_studenten_univ p20_29_studenten_univ VISIBLE RATIO;
            p20_29_studenten_athome p20_29_studenten_athome VISIBLE RATIO;
            p65_plus_wohnpflege p65_plus_wohnpflege VISIBLE RATIO;
            SUM_p65_plus_ret_athome SUM_p65_plus_ret_athome VISIBLE RATIO;
            patients patients VISIBLE RATIO;
            p5_64_w_rest_bev p5_64_w_rest_bev VISIBLE RATIO;
            population_vienna population_vienna VISIBLE RATIO""".format(POP_15_64_firmenreg))

# Process: Identity
print "Identity (buildings per grid cell)"
output_identity = wsTemp + os.sep + "identity_grid"
arcpy.Identity_analysis("f_layer", input_fc2, output_identity, "ALL", "", "NO_RELATIONSHIPS")

# Process: Spatial Join
print "Spatial Join (bkm from identity to grid)"
FID_bkm_output3 = "FID_" + output_bkm3 + analysis_no
arcpy.SpatialJoin_analysis(input_fc2, output_identity, output_fc, "JOIN_ONE_TO_ONE", "KEEP_ALL",
    """Shape_Length "Shape_Length" false true true 8 Double 0 0 ,First,#,{0},Shape_Length,-1,-1;
        Shape_Area "Shape_Area" false true true 8 Double 0 0 ,First,#,{0},Shape_Area,-1,-1;
        {3} "{3}" false true true 8 Double 0 0 ,First,#,{0},{3},-1,-1;
        {1} "{1}" true true false 4 Long 0 0 ,First,#,{1},{2},-1,-1;
        {4} "{4}" true true false 8 Double 0 0 ,First,#,{2},{4},-1,-1;
        {5} "{5}" true true false 8 Double 0 0 ,First,#,{2},{5},-1,-1;
        {6} "{6}" true true false 8 Double 0 0 ,Mean,#,{2},{6},-1,-1;
        {7} "{7}" true true false 8 Double 0 0 ,First,#,{2},{7},-1,-1;
        {8} "{8}" true true false 150 Text 0 0 ,First,#,{2},{8},-1,-1;
        {9} "{9}" true true false 50 Text 0 0 ,First,#,{2},{9},-1,-1;
        {10} "{10}" true true false 70 Text 0 0 ,First,#,{2},{10},-1,-1;
        p15_64_w_hosp "p15_64_w_hosp" true true false 8 Double 0 0 ,Sum,#,{2},p15_64_w_hosp,-1,-1;
        p15_64_w_hosp_athome "p15_64_w_hosp_athome" true true false 8 Double 0 0
,Sum,#,{2},p15_64_w_hosp_athome,-1,-1;
        p15_64_w_univ "p15_64_w_univ" true true false 8 Double 0 0 ,Sum,#,{2},p15_64_w_univ,-1,-1;
        p15_64_w_vhs "p15_64_w_vhs" true true false 8 Double 0 0 ,Sum,#,{2},p15_64_w_vhs,-1,-1;
        p15_64_w_hosp_univ_vhs "p15_64_w_hosp_univ_vhs" true true false 8 Double 0 0
,Sum,#,{2},p15_64_w_hosp_univ_vhs,-1,-1;
        {11} "{11}" true true false 250 Text 0 0 ,First,#,{2},{11},-1,-1;
        {12} "{12}" true true false 150 Text 0 0 ,First,#,{2},{12},-1,-1;
        {13} "{13}" true true false 250 Text 0 0 ,First,#,{2},{13},-1,-1;
        {14} "{14}" true true false 8 Double 0 0 ,Sum,#,{2},{14},-1,-1;
        p15_64_w_wrlinien_athome "p15_64_w_wrlinien_athome" true true false 8 Double 0 0
,Sum,#,{2},p15_64_w_wrlinien_athome,-1,-1;
        SUM_p15_64_w_remaining_working_pop "SUM_p15_64_w_remaining_working_pop" true true false 8
Double 0 0 ,Sum,#,{2},SUM_p15_64_w_remaining_working_pop,-1,-1;
        p15_64_w_unemployed "p15_64_w_unemployed" true true false 8 Double 0 0 ,Sum,#,{2},p15_64_w_
unemployed,-1,-1;
        p0_4_w_kiga "p0_4_w_kiga" true true false 8 Double 0 0 ,Sum,#,{2},p0_4_w_kiga,-1,-1;
        SUM_p0_4_w_athome "SUM_p0_4_w_athome" true true false 8 Double 0 0 ,Sum,#,{2},SUM_p0_4_w_
```

```
athome,-1,-1;
        p5_14_w_pflichtschule "p5_14_w_pflichtschule" true true false 8 Double 0 0
,Sum,#,{2},p5_14_w_pflichtschule,-1,-1;
        p10_19_w_ahs "p10_19_w_ahs" true true false 8 Double 0 0 ,Sum,#,{2},p10_19_w_ahs,-1,-1;
        p15_19_w_andere_hoehere "p15_19_w_andere_hoehere" true true false 8 Double 0 0
,Sum,#,{2},p15_19_w_andere_hoehere,-1,-1;
        p15_19_w_berufsschule_schule "p15_19_w_berufsschule_schule" true true false 8 Double 0 0
,Sum,#,{2},p15_19_w_berufsschule_schule,-1,-1;
       p15_19_w_berufsschule_betrieb "p15_19_w_berufsschule_betrieb" true true false 8 Double 0 0
,Sum,#,{2},p15_19_w_berufsschule_betrieb,-1,-1;
        p20_29_studenten_univ "p20_29_studenten_univ" true true false 8 Double 0 0
,Sum,#,{2},p20_29_studenten_univ,-1,-1;
        p20_29_studenten_athome "p20_29_studenten_athome" true true false 8 Double 0 0
,Sum,#,{2},p20_29_studenten_athome,-1,-1;
        p65_plus_wohnpflege "p65_plus_wohnpflege" true true false 8 Double 0 0 ,Sum,#,{2},p65_plus_
wohnpflege,-1,-1;
        SUM_p65_plus_ret_athome "SUM_p65_plus_ret_athome" true true false 8 Double 0 0 ,Sum,#,{2},-
SUM_p65_plus_ret_athome,-1,-1;
        patients "patients" true true false 8 Double 0 0 ,Sum,#,{2},patients,-1,-1;
       p5_64_w_rest_bev "p5_64_w_rest_bev" true true false 8 Double 0 0 ,Sum,#,{2},p5_64_w_rest_
bev,-1,-1;
        population_vienna "population_vienna" true true false 8 Double 0 0 ,Sum,#,{2},population_
vienna,-1,-1"""
        .format(input_fc2, FID_bkm_output3, output_identity, ID_grid, w_factor_bkm, ID_bkm, height_
bkm, vol_bkm, ID_hospital, NAME_hospital, USE_bkm, NAME_univ1, NAME_univ2, NAME_vhs, POP_15_64_fir-
menreg), "CONTAINS", "#", "#")

# Check if Spatial Join worked - if not (code_bkm = 0) print error message and exit
var = arcpy.da.SearchCursor(output_fc, ("{}".format(ID_bkm),)).next()[0]
if var == 0:
    print "ERROR IN FIELD MAPS"
    sys.exit()

#------------------------------------------ANALYSIS 2---------------------------------------------#
#-----------------------------Aggregate population (wrlinien) to raster---------------------------#
print "\n-------------------------------ANALYSIS 2--------------------------------"
print "------------------Aggregate population (wrlinien) to raster---------------\n"

# Make Feature Layer
arcpy.MakeFeatureLayer_management(input_fc3, "f_layer", "", "", "p15_64_wr_linien p15_64_wr_linien
VISIBLE RATIO")

# Process: Identity
print "Identity (p15_64_wr_linien per grid cell)"
output_identity = wsTemp + os.sep + "identity_grid_wrlinien"
arcpy.Identity_analysis("f_layer", input_fc2, output_identity, "ALL", "", "NO_RELATIONSHIPS")

# Process: Spatial Join
print "Spatial Join (bkm from identity to grid)"
output_join = wsTemp + os.sep + "join_grid_wrlinien"
arcpy.SpatialJoin_analysis(input_fc2, output_identity, output_join, "JOIN_ONE_TO_ONE", "KEEP_ALL",
    “””p15_64_wr_linien “p15_64_wr_linien” true true false 8 Double 0 0 ,Sum,#,{0},p15_64_wr_lin-
ien,-1,-1;
        {1} “{1}” true true false 8 Double 0 0 ,First,#,{0},{1},-1,-1”””
        .format(output_identity, ID_grid), “CONTAINS”, “#”, “#”)

# Check if Spatial Join worked - if not (p15_64_wr_linien = 0) print error message and exit
var = arcpy.da.SearchCursor(output_join, (“p15_64_wr_linien”,)).next()[0]
if var == 0:
    print “ERROR IN FIELD MAPS”
    sys.exit()

# Join Field p15_64_wr_linien
print “Join Field”
arcpy.JoinField_management(output_fc, ID_grid, output_join, ID_grid, “p15_64_wr_linien”)

# Replace <Null> with 0
print “Replace <Null> with 0”
expression = “myCalc(!population_vienna!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc, “population_vienna”, expression, “PYTHON_9.3”, code-
block)

expression = “myCalc(!p15_64_wr_linien!)”
codeblock = “””def myCalc (population):
    if population == None:
        return 0
    else:
        return (population)”””
arcpy.CalculateField_management(output_fc, “p15_64_wr_linien”, expression, “PYTHON_9.3”, codeblock)

# Calculating fields
print “Calculating fields”
arcpy.AddField_management(output_fc, “population_vienna_plus_wrlinien”, “DOUBLE”)
arcpy.CalculateField_management(output_fc, “population_vienna_plus_wrlinien”, “[population_vien-
na]+[p15_64_wr_linien]”)

# Statistics
arcpy.Statistics_analysis(input_fc, table_temp, “population_vienna SUM”)
SC = arcpy.SearchCursor(table_temp)
field_sum = “SUM_population_vienna”
for row in SC:
    population = row.getValue(field_sum)

SC = arcpy.SearchCursor(input_table_wrlinien)
field = POP_15_64_wrlinien
for row in SC:
    employees_wr_linien = row.getValue(field)
    employees_wr_linien2 = employees_wr_linien/2

pop_plus_wrlinien = population + employees_wr_linien2
print "\nTotal Population (bkm_output3) = " + str(int(round(pop_plus_wrlinien,0)))

arcpy.Statistics_analysis(output_fc, table_temp, "population_vienna_plus_wrlinien SUM")
SC = arcpy.SearchCursor(table_temp)
field_sum = "SUM_population_vienna_plus_wrlinien"
for row in SC:
    population = row.getValue(field_sum)
    print "Total Population (grid_aggregated) = " + str(int(round(population,0)))

#-----------------------------------------------------------------------------------------------#
print "\n---------------------------------------------------------------------------"

print "\nOutput Data:"
print output_name_aggregate + " (polygon_fc)"

print "\nOutput Fields in " + output_name_aggregate + ":"
print "p15_64_w_hosp"
print "p15_64_w_hosp_athome"
print "p15_64_w_univ"
print "p15_64_w_vhs"
print "p15_64_w_hosp_univ_vhs"
print "SUM_{}".format(POP_15_64_firmenreg)
print "p15_64_w_wrlinien_athome"
print "SUM_p15_64_w_remaining_working_pop"
print "p15_64_w_unemployed"
print "p0_4_w_kiga"
print "SUM_p0_4_w_athome"
print "p5_14_w_pflichtschule"
print "p10_19_w_ahs"
print "p15_19_w_andere_hoehere"
print "p15_19_w_berufsschule_schule"
print "p15_19_w_berufsschule_betrieb"
print "p20_29_studenten_univ"
print "p20_29_studenten_athome"
print "p65_plus_wohnpflege"
print "SUM_p65_plus_ret_athome"
print "patients"
print "p5_64_w_rest_bev"
print "population_vienna"
print "population_vienna_plus_wrlinien"


end = datetime.datetime.now()

print '\nfinished run: %0.19s\n' % (end),
print 'runtime: %0.7s' % (end-start)
print '\ntotal runtime: %0.7s\n' % (end-start1)
```