IIASA

International Institute for Applied Systems Analysis • A-2361 Laxenburg • Austria
Tel: +43 2236 807 • Fax: +43 2236 71313 • E-mail: info@iiasa.ac.at • Web: www.iiasa.ac.at

INTERIM REPORT IR-97-54 / December

Analysis of a Russian Landscape Map and Landscape Classification for Use in Computer-aided Forestry Research

Victor Wagner (vitus@agropc.msk.su)

Approved by
Sten Nilsson (nilsson@iiasa.ac.at)
Leader, *Forest Resources Project*

Contents

Foreword

IIASA, the Russian Academy of Sciences, and the Russian Federal Forest Service, in agreement with the Russian Ministry of the Environment and Natural Resources, signed agreements in 1992 and 1994 to carry out a large-scale study on the Siberian forest sector. The overall objective of the study is to focus on policy options that would encourage sustainable development of the sector. The goals are to assess Siberia's forest resources, forest industries, and infrastructure; to examine the forests' economic, social, and biospheric functions; with these functions in mind, to identify possible pathways for their sustainable development; and to translate these pathways into policy options for Russian and international agencies.

The first phase of the study concentrated on the generation of extensive and consistent databases for the total forest sector of Siberia and Russia.

In its second phase, the study has encompassed assessment studies of the greenhouse gas balances, forest resources and forest utilization, biodiversity and landscapes, non-wood products and functions, environmental status, transportation infrastructure, forest industry and markets, and socioeconomics.

This report, carried out by Dr. Victor Wagner of The Dokuchaev Soil Institute, Moscow, under the supervision of Profs. A. Shvidenko and S. Nilsson during his stay at IIASA in 1997, is a contribution to the analyses of the topic of mainly biodiversity. This work has been financially supported by the Swedish Council for Planning and Coordination of Research.

Analysis of a Russian Landscape Map and Landscape Classification for Use in Computer-aided Forestry Research

Victor Wagner

1 Introduction

The correct choice of territorial units is a very important question in all geographic and ecologic research. The major part of environmental information on Russia is given by administrative units, which sometimes occupies several natural zones.

Therefore utilization of existing physical-geographical data and concepts is critical for many forestry applications, e.g., for development of vegetational models at regional and global scale as well as evaluation of the main biogeochemical cycles.

Russian geography has accumulated a great experience in territorial unit classifications and regionalization of various scales. One of the most general approaches is the landscape approach. According to Berg[1], a geographical landscape is defined as *"a total or a group of objects and phenomena, with certain peculiarities concerning relief, climate, water, soil and vegetative cover, and animal habitat, as well as human activities, repeated in a harmonized way over a known land area"*.

Therefore the concept of landscapes represents aggregated knowledge of all natural conditions of a definite territory. A landscape is a relatively small territorial unit. Thus, typological classifications of landscapes are usually used for regional and subglobal maps.

There are several landscape maps existing for the total Russian territory. The two most recent maps are developed by Gudilin[2] and Isachenko[3].

The main problem with utilizing this information in computer-based analysis is that these maps, as well as other sources of landscape information, were intended for human reading, rather than GIS and database processing.

In this study we made an attempt to perform semiautomated analysis of the landscape map by Gudilin, in order to extract information related to the boreal forests in Russia.

2 Description of the Landscape Map and the Classification

The Dokuchaev Soil Institute in Moscow has produced a digital version of the landscape map developed by Gudilin (Rozhkov *et al.*[4]). This work was carried out in collaboration with the authors of the original map, and the main goal of the work was to reproduce the information from the original map as exactly as possible.

2.1 Level of details and properties of the map

The landscape map of Russia has the scale of 1:2,500,000 — a scale which allows to represent the whole of Russia on one map. It contains more than 27,000 polygons, which

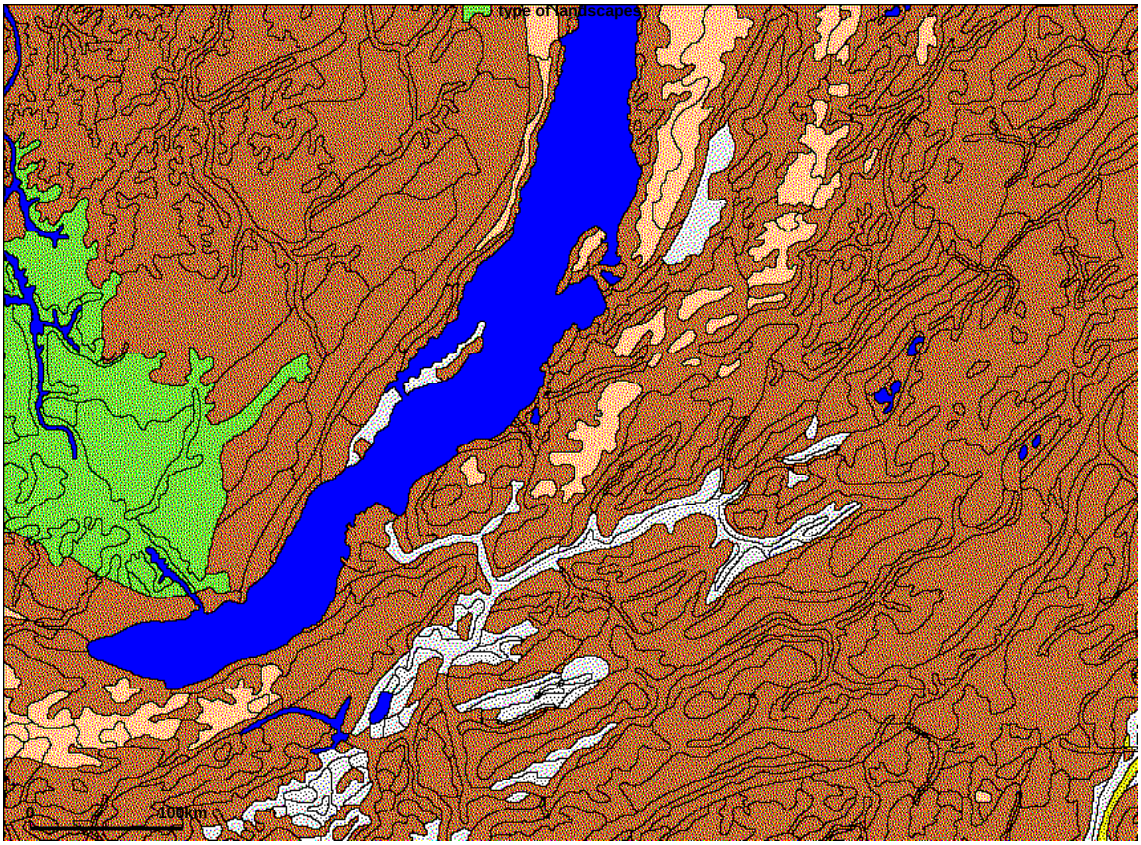

Figure 1: Fragment of the landscape map.

correspond to the most detailed level of landscape classification. Figure 1 shows a typical part of the map and illustrates the level of details.

The thematic information on the map is represented in three different ways:

1. Tabular legend, printed on the map itself. It shows the landscape classification at the *genera* level and the correspondence of the classification units with the cartographic signs.

2. Short legend, a book of 343 pages with a one-paragraph description of each of the landscapes. This legend is fully reproduced in electronic form. Landscapes listed in this legend are referenced by index, and plotted in each polygon of the map.

3. Long legend, consisting of four volumes, which includes some additional properties, not listed in the short legend (about half a page per landscape). Due to time and financial constraints, this legend was not included in the electronic version.

The paper map contains some shortcomings, which are unimportant for manual analysis, but may cause problems at GIS processing. Most important of these shortcomings are valley polygons.

Due to scale limitations, diversity of the river valleys could not always be represented appropriately on the map of the scale of 1:2,500,000. Thus, some of these polygons were originally left unclassified, with the remark that they belong to the same *subtype* of landscape as surrounding polygons.

During the development of the electronic version, special efforts have been made to overcome this uncertainty, and now these polygons have classification information attached to them to the *subtype* level.

These efforts lead to an increased number of polygons, from 25,000 in the scanned image to 27,000 in the final version, due to the fact that some valley polygons were subdivided into different subtypes.

Nevertheless, these polygons were excluded from this study, because the study is mainly based on information extracted from the so-called *kind* descriptions, rather than on the upper levels of classification.

The digitizing and further processing of the electronic version was carried out with an accuracy of 500 m (0.2 mm on the paper source). It is definitely higher than the accuracy of the source map, which could be up to 2 mm (5 km) according to Russian thematic cartographic standards.

The original map was plotted in Kavraysky conic equidistant projection, as most of the maps covering total Russia. Most operations with the electronic version were carried out by using Albers equal-area projection, due to the fact that it allows more accurate area calculations. Technical details about this map can be found in the Dokuchaev Soil Institute's on-line map catalogue[5].

The landscape classification has been translated into English and classifiers for all levels except *kind*, are available in both Russian and English (see Appendix A). However, descriptions of landscape *kinds* were not translated due to the big value, and this study was carried out using the Russian-language descriptions.

2.2 Structure of the classification

The map is based on a hierarchical landscape classification, which includes mainly climatic, geomorphological and vegetational aspects, although there are two levels which are distinguished via complex criteria — level of *variant* and level of *kind* (see Table 1).

The upper levels of the classification can be divided into two groups — geology/geomorphology and climate/biology levels. The levels of each group correlate with each other (usually, value of lower layer within a group can occur only in one *variant* of higher level), but groups themselves are almost orthogonal. The map design reflects this situation and the legend of the map is represented as a matrix, where rows correspond to the combination of climatic/vegetation levels and the column to the combination of geomorphology/geology levels.

Table 1: Classification of landscape levels.

Level	Area of concern	Main criteria	Number of items
Division	Geology	Tectonic conditions	2
Group	Climate	Climatic belts and sectors	18
Class	Geology	Megarelief	4
Type	Vegetation	Vegetation zones	35
Subtype	Vegetation	Vegetation types	99
Genera	Geomorphology	Type and genesis of relief	63
Variant	Complex	Large physical geographical regions	18
Kind	Complex	All factors, which do not match any other level	3446

I Kola–Karelian
II East European
III Middle Siberian
IV Kazakhstanian
V Central Asian
VI Middle Siberian
VII Novaya Zemlya–Uralian
VIII Severnaya Zemlya–Taymyrian
IX Tjan–Shan–Jungarian
X Altay–Sayanian
XI Cisbaykalian–Transbaikalian
XII North–Eastern
XIII Far East–Daurian
XIV Carpatian
XV Crimea–Caucasian
XVI Kopet–Dagian
XVII Pamirian
XVIII Kamchatka–Sakhalinian

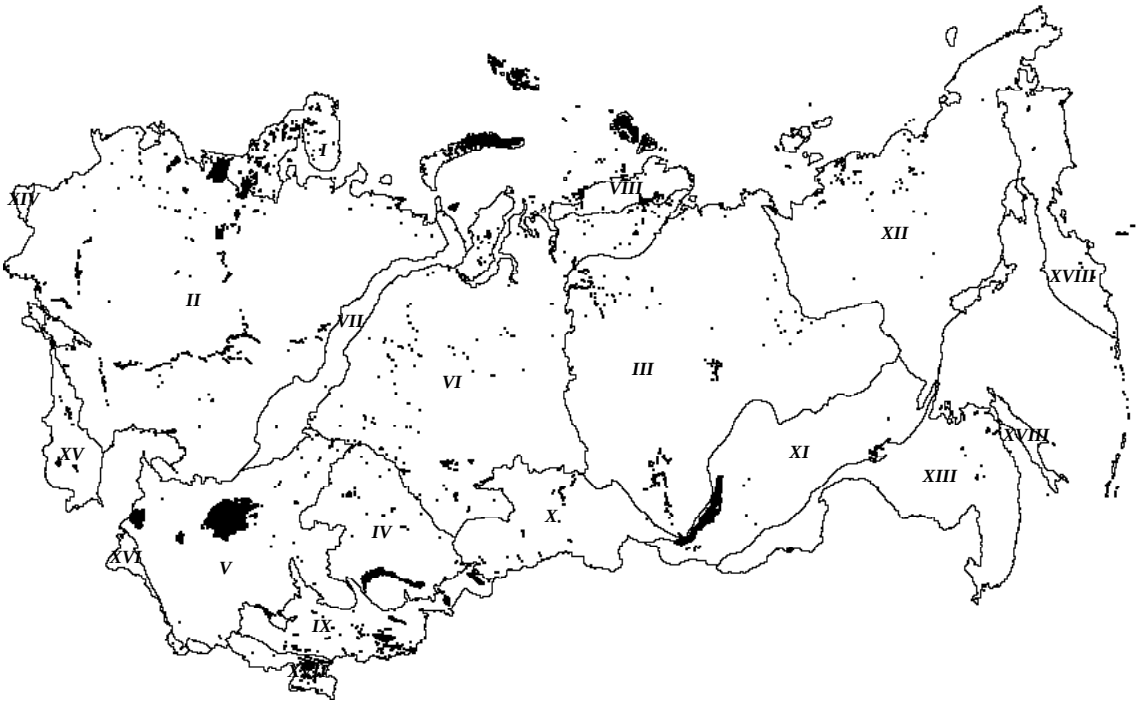

Figure 2: *Variants* of landscapes.

2.3 Description of landscape *kinds*

The lowest level of landscape classification has long textual descriptions, instead of short classification names. Typically, the description is one long sentence, which lists all properties of a given landscape, starting from relief type and ending with landcover and anthropogenic influence. Length of this description can vary from 44 to 475 characters.

One of the most important properties of the descriptions is that they consist of very limited sets of words. The description contains 3,446 landscapes using about 2,000 different Russian words and word forms for all properties of the landscape. Without a computer-based analysis of the description quantitative analyses would be impossible.

2.4 Regionalization at the level of *variant*

Although *kind* descriptions are the most valuable source of information for analysis, the second level of classification — *variant* — is important too. There are some differences between the large geographic regions of Russia, which are not reflected in *kind* descriptions. For example, there are three main species of *Larix*, which form the Siberian larch forests. In the descriptions all of them are addressed as "larch", but comparisons of the *variant* maps (Figure 2) with areas of those species show that they could be almost clearly distinguished on a *variant* basis.

Therefore a *variant*-based approach is important for linking of the landscape information with other sources of information, as described in Section 5.3.

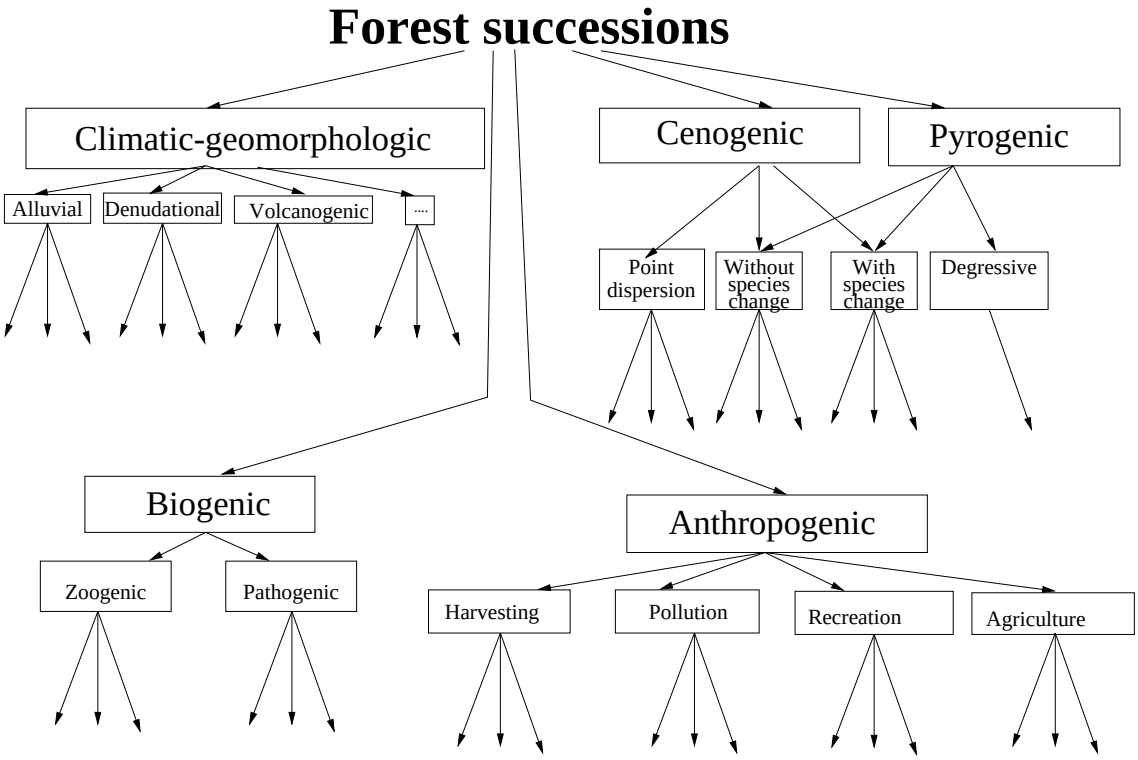

Figure 3: Classification of forest successions.

3 Forest Succession Database Applied to Landscapes

Availability of highly detailed information on natural conditions, which could be obtained from a landscape map, allows to develop spatial databases of forest types and possible trend of forest development, suitable for simulation models of forest growth and carbon balance. This information is also an important tool for biodiversity analyses.

Currently, information about the forest resources in Russia is available from the State Forest Account (SFA). This account presents data per enterprise. However, the forest enterprises can be large and occupy heterogeneous territory, especially in the less populated regions of Siberia.

Therefore it is essential to link statistical data obtained from this account to lesser and more homogeneous territorial units, i.e., landscape *kinds*.

Another source of forest specific information is the Forest Succession Database, developed by IIASA's Boreal Forest Resources Project. This latter database has almost no spatial references, but it is based on a forest succession classification, which is very close to the landscape classification in some aspects.

The upper level of this classification on successions is based on the cause of forest change, which can be either natural (e.g., climate change and geomorphological processes) or anthropogenic (e.g., forest harvesting and atmospheric pollution). This level of classification is shown in Figure 3. Some information for these levels, especially for "natural" (cenogenic and climate-geomorphologic) successions, can be easily obtained from the landscape *kind* description.

The next levels of succession classification present the type of development and tree species, which change during succession. There are three types of development: dynamical stability, progress, and degradation (Figure 4).

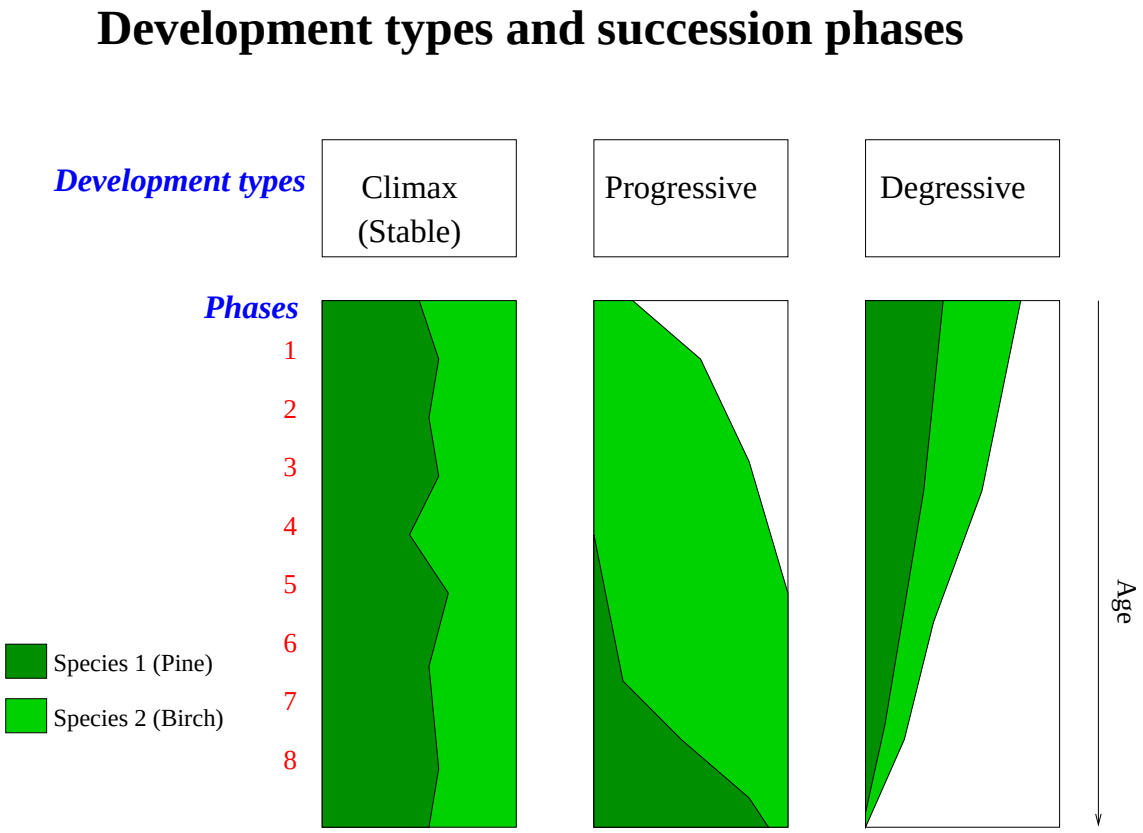

Figure 4: Dynamic of two tree species ratio during phases of different types of development.

Succession is described as a set of development phases. A phase is defined as a specific period of a succession process which has a definite morphological structure (e.g., secondary birch forest after fire in spruce-fir forest). A phase could be divided into age stages (young, middle-aged, immature, and mature). A complete succession may have a duration of up to a thousand years, including all phases.

Information, which could be obtained from the landscape descriptions, i.e., geomorphological process, types of forests, distribution of different species, allow us to select few succession types, which could be applicable to this kind of landscape from a long list of succession types, possible in a given ecoregion.

4 Analysis of Non-formalized Text

4.1 Problem of free text

The landscape classification was designed for human reading, not for computerized processing. Therefore most information is contained in descriptions, which are free-form text and have no explicit structure. This makes database and GIS-processing very difficult, if possible at all. The amount of information contained in the landscape *kind* description is enormous. The information includes dominant and subdominant landcover type, main vegetation associations, anthropogenic disturbances, relief type, parent material genesis and granulometry and many more attributes.

Therefore it is essential to convert the plain text description into some formalized tabular structure, which can be used for conventional database processing.

This conversion inevitably causes a loss of information, because information in human readable text is contained not only in words themselves, but also in the order of words. The amount of alternatives (for example, alternative tree species) is also important in order to estimate the significance of these alternatives.

But there are also some aspects that will simplify the problem of text formalization, and therefore make this work possible at all.

1. The various properties of the landscapes are listed in all descriptions in the same order (relief type first, then other geology and geomorphology information, followed by parent material genesis, then landcover and vegetation types from dominant to least dominant).

2. The set of words used in the descriptions is highly standardized. There are only 2,767 distinct words, including the grammatic forms.

3. The grammatic form of the words normally have a well-established meaning. This can be counted as benefit of the Russian language due to the fact that in Russian information can often be derived from word suffixes.

Therefore, the following steps of analysis are used:

1. Separation of the description into parts concerning geological and biological components of landscape.

2. Classification of words in each part and replacing words of written language in all their various forms with a fixed set of terms.

3. Creation of a list of terms, which occurs in each description with their relative weight. Sets of such lists conform some constraints which are usually applied to databases and thus can be used in GIS processing, and other computer-based applications.

The achieved result differs from the original goal of converting textual descriptions into a relational database, but have some advantages.

All the information of original description is retained, because all operations above are applied to the original legend text.

Not only structured information, but also technology and a set of software tools are provided, so if research with totally different goals is carried out, this approach could be applied again.

4.2 Separation of parts using sentence structure

Separation of geological and biological parts of descriptions is an important preliminary step for automated description analysis. It allows to put aside all information irrelevant to landcover and decreases the amount of words, which should be considered in later stages of analysis, by more than two times.

But the number of descriptions is too large to do this separation manually. At the same time our knowledge of the description structure was too limited at this stage of the analysis to define some algorithmic way to perform the separation.

Thus, a composite approach based on few heuristics with the following manual corrections was chosen.

Punctuation rules of the Russian language require that description of different properties should be separated by commas. Therefore, if geomorphological and landcover parts of the description are not separated by comma, it should be considered either as character mistake or as typo in the original book. Thus, if such cases were encountered during the manual checking of the separation results, corrections were made in the original text file.

With this assumption the problem of separation is reduced to the choice of the right comma. It can be done using keyword analysis, but at this stage no keyword analysis was performed. Thus, a simple table which estimates the number of commas separating the parts in the geomorphology description from the total number of such parts in the description was used.

These heuristics work in 86% of the cases and it is simplier to correct remaining cases manually than to develop more complex heuristics, which probably would be highly specific and not applicable to any text, other than the legend.

The results of this manual correction were stored separately from the legend text, which allows to regenerate separate parts in the case of correcting typos in the original legend.

A special user interface was developed for reviewing and correcting results of the separation (see Appendix B). In our opinion this approach — a close integration of manual and computer-based work — is fruitful under these conditions. A rough estimation is that this interface speeds up the process of reviewing the separation about ten times.

After this step of analysis we have two text files, one of them with geological and geomorphological descriptions and the other containing the vegetation and landcover description. It is possible to apply the same approach to the geomorphological description again, in order to separate relief data from parent material, but the landcover description has a more complex structure.

4.3 Word classification approach

The description of vegetation and landcover usually consists of several descriptions of the landcover types, which occur in the specific landscape *kind*. They can be separated not only by commas, but also by the word “and”. Commas can also occur inside the individual landcover type in constructs like “with . . . ”.

Therefore, a simplistic approach like the one described in Section 4.2 is impossible in this case. Several attempts were made to construct a true grammatic parser for this limited subset of written language, but they proved to be unnecessary complex.

Finally, we used an approach based on the semantic meaning of words. As a preliminary step, word frequency analysis was carried out. It shows that there are about 1,040 distinct words in the biological part of the descriptions.

Analysis of the frequencies of the word combinations shows that there are some word combinations which should be treated as one term. After this preliminary stage real word classification can be carried out.

4.3.1 Word classes

Word classes which were used for landcover analysis are listed in Table 2. Most important for the goals of our study are classes like “types”, which indicate landcover types such as forests, arable lands, etc., and “forests”, which indicate types of forest. There is also the class “multivalue”, which indicates words having different meaning depending on the context. Section 4.3.3 describes our approach in this situation. Initially this class was created for the single word “mixed”, which can mean primary forest type, if it occurs in

the forest description and by that has a meaning to use. It is totally irrelevant to our study, in steppe or desert landscapes. Further investigations prove that there are other words which could exhibit such behavior. Classification of words was carried out manually. A special environment was developed for this purpose to provide as much information about a given word as possible.

Table 2: Classes of words used for landcover information analysis.

Type	Criteria
types	Words indicating type of landcover
forest	Words indicating major forest species (possible types of forest)
vegetation	Vegetation species which can occur in more than one vegetation type
meaningless	Words which can be safely excluded from analysis
non-veget	Non-vegetation land types (bare rock, sand, etc.)
modifier	Words which affect relative weight of the following word (rare, often)
steppe	Steppe and desert vegetation species
tundra	Vegetation species and other information belonging to tundra landscapes and bogs
multivalue	Words which can indicate primary forest types, but only in certain contexts
disable-forest	Words indicating that this landcover type is not forest, regardless of the presence of words of the class "forest" ("with sparse trees")
zone	Words indicating natural zones (in combinations like "Arctic tundra")
formation	Types of forest stands
derived-from	Words indicating that the following words are applicable to the former, rather than the present, state of the landscape

The following information was taken into account:

- Total frequency of words in the text.
- List of word combinations in which the word can occur.[1]
- Frequency of word combinations, absolute and relative to both words in the combination.
- In ambiguous cases, the original text of descriptions where this word occurs was also consulted.

The environment where this classification was carried out also allows to declare multiword combinations as a single term and modify the list of used classes.

4.3.2 Grammatic forms of the words in the Russian language as guideline for significance estimation

Each word could occur in the description in different forms. Moreover, sometimes synonymous words were used by the authors of the descriptions. Therefore, during the word classification stage, various forms of the words were replaced by a single term. Sometimes one term was used for several synonymous words. These terms are referred to below as "values" of words.

[1] Consideration of possible combinations with the other word in a pair was often required.

But word suffixes can sometimes contain useful information. For example, name of a tree species can be used as adjective for the word “forests” and in constructs like “with . . . ”. This means that a given forest species has less relative weight than in the case where it is used as an adjective.

Each of these cases could easily be distinguished by using word suffixes. Practically, simple regular expressions were sufficient to correctly estimate the grammatic form in 90% of the cases.

This information was retained during the word classification stage and stored as a “subclass” of the word.

4.3.3 Multilevel structure of the description

From the analysis we want to obtain the following information as a first result. For each kind of landscape we should have a list of forest species along with their relative weight and total relative weight for all types of non-forest landcover.

The following procedure was used to compute the relative weight:

It was assumed that the landcover types are listed in the description in order of their importance. The same is applicable to the list of adjectives, describing forest species and list of parts of complex adjectives.[2]

For each of these three levels relative weights were estimated by counting the parts in the description and a sequential number of given parts by using Table 3. The final weight of the term was estimated as a product of relative weights at all three levels.

Table 3: Relative weights of parts of the landscape description.

Number of parts	Relative weights %							
1	100							
2	60	40						
3	50	30	20					
4	50	20	20	10				
5	40	20	20	10	10			
6	40	15	15	10	10	10		
7	30	15	15	10	10	10	10	
8	30	15	10	10	10	10	10	5

The types of landcover can easily be distinguished in this step of the analysis, because their description is ended by words of the class “types”. But a simple extraction of all forest types would not give a proper list of all land cover types with respect to their importance. There could be, for example, two distinct types of forests separated by the word “and”, which has only one word “forest” describing them.

Thus, first all words were divided into parts, endings by word type were identified and separated into forest types and non-forest types. The parts of the description which contained the terms “forest” and “sparse forest” were counted as forests as well as parts which contained at least one word of the class “forest” and did not contain words of the class “disable-forest” and “derived-from”.

[2]This is not evident, because the tradition in Russian earth sciences require that the most important part of a complex adjective should come at the end. But expert evaluation proves that this rule is not followed in the legend of this map.

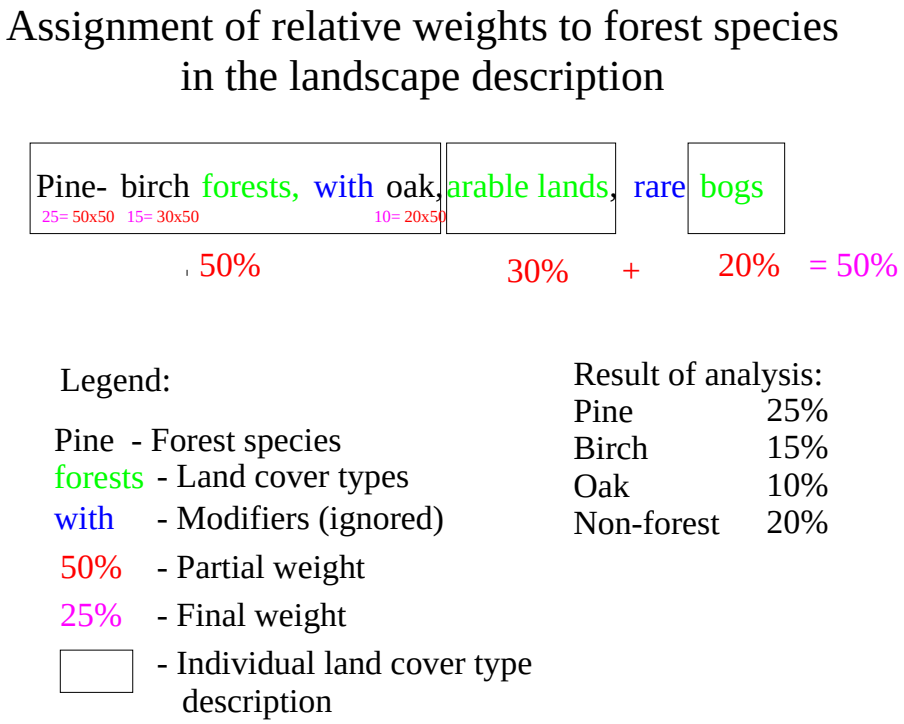

Figure 5: Description of the parsing scheme.

Parts containing words of the class “derived-from” were discarded in this step in order to avoid influences on the relative weight of the present vegetation cover types.

Then the forest types were analyzed by using the word *subtypes* (i.e., grammatic form of the words) to separate individual forest types.

In this step we counted the number of landcover types and the number of forest species (classes “forest” and “mixed”) in each part.

Thus, two of three levels of the description structure were effectively joined together and we computed relative weight as product of two factors, weight of the forest type in the total list of landcover types, and weight of forest species in the complex adjective, which could look like:

Pine-birch forests, with spruce and, seldom, cedar

This example contains one word of type “types”, four words of type “forest”, two words of class “modifier” and one word of class “meaningless”.

In the present system of analysis “modifiers” were treated just like “meaningless” words, but in the future it is possible to use them for refining the relative weights.

Figure 5 shows the scheme for description of the parsing and the relative weight assignment.

This stage of analysis produces a list of forest species for each kind of landscapes. This list can be used in GIS processing immediately, although it does not conform to the constraints of a relational database model.

The relative weights computed in this step do not represent the real area extent of forest species or wood reserves. They only indicate the importance of the species according to the original classification by the authors.

But these weights provide knowledge which allow us to distribute the SFA information over several landscape *kinds* inside all forest enterprises, taking into account the natural conditions of these landscapes as well as their area extent.

Unfortunately, there was no possibility to create a user interface for this step of the analysis, which would allow to construct a variety of queries. All scripts which currently exist are non-interactive and oriented toward a particular type of queries.

4.4 Technology of word analysis as alternative to traditional databases

Finally, we have developed a technology for computer-based analysis of certain types of written texts. This technology is not limited to this particular map legend. Although it is oriented to highly specialized forms of text — physical geography descriptions — it can be used to process large amounts of data, including field observations, collected during the pre-computerized age.

This technology has some benefits (described in the following) over converting such data into relational databases.

First, original text of description is preserved through the entire analysis. It means that no information loss occurs until we decide which information we want to process and which is irrelevant.

Second, any corrections (corrections of typographical errors) made to the original text are automatically propagated to the final results.

Third, if new data are acquired, they can easily be integrated into the existing database. If the new data follow the same outline as existing data, very few modifications are required for the word classification tables.

Although this technology is new and incomplete it can be recommended for use in other areas.

5 Results of Analysis

5.1 Distribution of tree species

The first information which can be derived from the landscape description is the distribution of individual forest species. This is a good test of the technology, because it can be easily compared with existing geobotanical maps. Figure 6 shows the distribution of larch forests in Russia.

The intensity of the green color shows the relative weight of larch species in a given point. As expected, larch forests are mainly located in Siberia.

5.2 Derivation of forest types from the landscape description

The second kind of information which can be extracted is the type of forests. In Figure 7 the type of forests is indicated by prevalent tree species, regardless of the relative weight of the same.

At a first glance, few inconsistencies can be found on this map, as well as on the map illustrating the forest cover (Figure 8). For example, in the Kola Peninsula the forests are extended almost to the coastline. Usually, the border between tundra and sparse forests is drawn further south. But Gudilin's map is based on a large amount of remote sensing information (aerial and satellite photos as well as aerovisual observations), so it could be even more accurate than other maps. These latter maps are usually based, especially

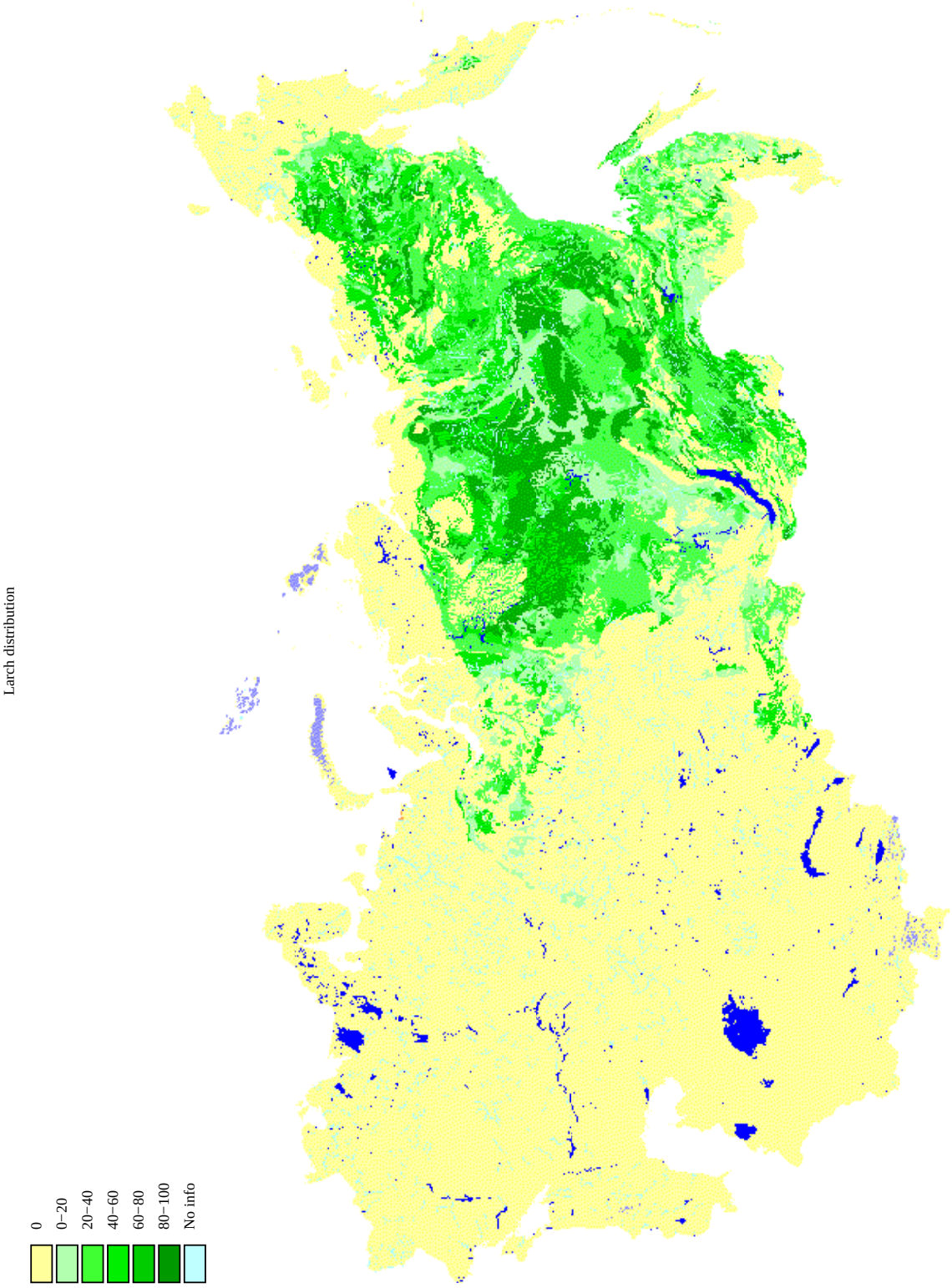

Figure 6: Distribution of larch forests.

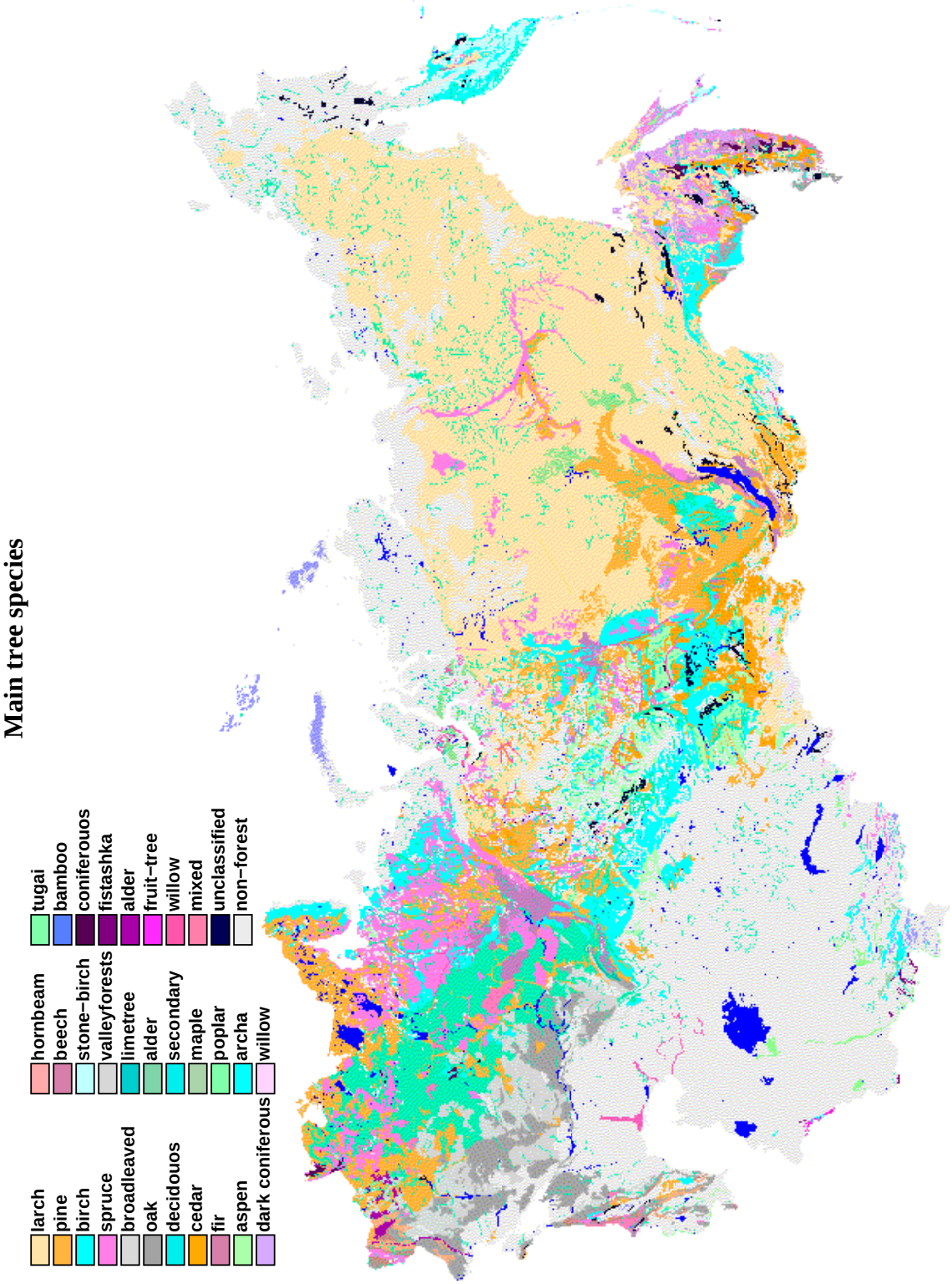

Figure 7: Map of prevalent forest species.

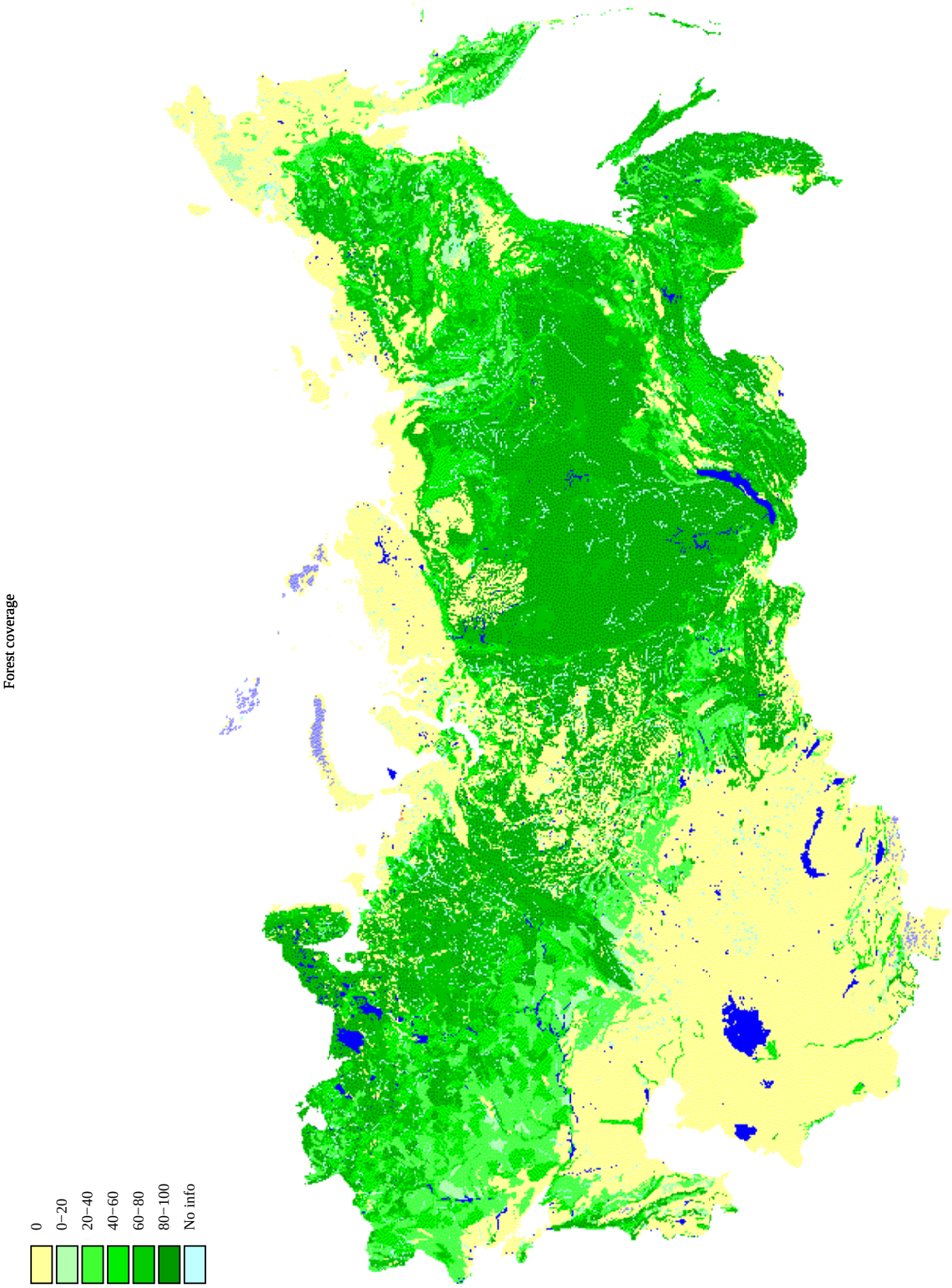

Figure 8: Distribution of forest lands.

in less populated polar regions, on old published works and general physical-geographic concepts.

The unusually large extent of non-forest patches in Western Siberia should be noticed. The explanation is that bog landscapes with some forest species, for example *spruce bogs and pine on ridges*, are classified as non-forest landscapes with our approach, although they can be classified as sparse forests using aerial photos or results of field investigations.

5.3 A regionalized approach of linking landscape descriptions with the State Forest Account (SFA)

Obviously, the next step of analysis would be to link the landscape information with SFA information and the succession database to replace relative weights with real figures of forest resources. This imposes several problems.

First, forest species names are used in the SFA, while landscape descriptions usually use less specific *genera* names. But it is possible to choose particular species within *genera*, using information for landscape *variant*.

Second, the real area extent as given in FSA databases are not necessary proportional to our relative weight, computed from the text description. Thus, the distribution of the forest resources information over landscapes should probably be carried out as an optimization problem. Due to the fact that the area of a landscape is usually larger than the borders of a given forest enterprise — particular enterprises cover several kinds of landscapes — this optimization should be performed by some larger territorial units. This would allow to impose additional constraints, such as that distribution of forests in a particular *kind* of landscape should be the same in all forest enterprises. There are two possible choices of territorial units for this optimization, namely the landscape variants and ecoregions.

While the landscape *variant* boundaries never cross boundaries of landscape *kind*, ecoregion boundaries never cross boundaries of the forest enterprises, so there are no geometric preferences in this choice. Landscape *variants* are sufficiently larger than ecoregions, so if it were possible to use the former, it would significantly decrease the amount of work. But it is possible that future investigations will prove that smaller territorial units are better for this kind of optimization. In this case ecoregions should be used.

6 Conclusion

Results of this study show that the landscape map of the USSR developed by Gudilin[2] can be used for GIS-based research. It is a valuable source of information because it contains the most detailed division of total Russia into homogeneous natural units, based on uniform classifications and concepts.

Its legend contains a large amount of information, relevant for forest research. In conjunction with other sources of information, such as the State Forest Account and the IIASA succession database, it can even be used as base for small-scaled forest mapping and forest simulation models.

The developed technology allows to extract information from text descriptions of landscapes and use it for mapping by GIS-systems and for joint analysis with other databases.

References

[1] Berg, L.S., 1930, *Landscape-geographical Zones of USSR*, Institute of Plant Growth, Leningrad, 2nd edition, pp. 369 [in Russian].

[2] Gudilin, I.S., 1987, *Explanatory Text to the Landscape Map of the USSR at the Scale of 1:2.5 Million,* Gidrospecgeologia, Moscow, pp. 102 [in Russian].

[3] Isachenko A.G., ed., 1988, *Landscape Map of USSR (for higher schools)*, GUGK, Moscow [in Russian].

[4] Rozhkov, V., Efremov, D., Nilsson, S. Sedych, V., Shvidenko, A., Sokolov, V., and Wagner, V., 1996, *Siberian Landscape Classification and a Digitized Map of Siberian Landscapes*, WP-96-111, International Institute for Applied Systems Analysis, Laxenburg, Austria.

[5] *Dokuchaev Soil Institute On-line Map catalogue*, 1997, `http://www.grida.no/prog/polar/ecoreg/dsi/english.html`

Appendix A: Landscape Classification by Gudilin (1987)

A.1 Biology related levels of classification

Group	Type	Subtype
Arctic		
	Arctic (polar) deserts	
		No differentiation
	Mountain arctic (polar) deserts	
		Arctic desert low mountains
	Arctic tundra	
		No differentiation
	Mountain arctic tundra	
		Arctic desert low mountains
		Arctic tundra low mountains
Subarctic moderate continental and continental		
	Subarctic tundra	
		Northern tundra
		Southern tundra
	Forest tundra	
		No differentiation
	Mountain tundra	
		Tundra low mountains
		Tundra and sparse forest low mountains
		Desert-tundra middle mountains
	Mountain sparse forests	
		Tundra and sparse forest low mountains
		Sparse forest low mountains
Subarctic severe continental		
	Subarctic tundra	
		Northern tundra
		Southern tundra
	Forest tundra	
		No differentiation
	Mountain tundra	
		Tundra and sparse forest-tundra middle mountains
		Bare top uplands
		Tundra low mountains
		Tundra and sparse forest low mountains
		Tundra-bare top middle mountains
	Mountain sparse forests	
		Sparse forest and tundra-sparse forest low mountains
Boreal suboceanic(atlantic)		
	Taiga forests	
		Subtaiga (with mixed broadleaved and coniferous forests)
Boreal moderate continental		
	Mountain meadows	
		Meadow and tundra-meadow middle mountains

Group	Type	Subtype
	Taiga forests	
		Northern taiga
		Middle taiga
		Southern taiga
		Subtaiga (with mixed deciduous, broadleaved, and coniferous forests)
	Mountain taiga forests	
		Sparse taiga low mountains
		Taiga low mountains
		Forested low mountains (with mixed broadleaved and coniferous and deciduous forests)
Boreal continental		
	Mountain meadows	
		Meadow middle mountains
		Meadow uplands
	Mountain tundra and subtundra sparse forests	
		Meadow-tundra middle mountains
		Tundra and bare top-tundra uplands
	Taiga forests	
		Northern taiga
		Middle taiga
		Southern taiga
		Subtaiga (with mixed deciduous and coniferous forests and coniferous forests)
	Mountain taiga forests	
		Sparse taiga low mountains
		Taiga low mountains
		Sparse taiga middle mountains
		Taiga middle mountains
		Low mountains with exposition-dependent forests
Boreal severe continental		
	Mountain tundra and subtundra sparse forests	
		Bare top uplands
		Tundra-sparse forest middle mountains
		Tundra uplands
	Taiga forests	
		Northern taiga
		Middle and southern taiga
	Mountain taiga forests	
		Sparse taiga low mountains
		Taiga low mountains
		Sparse taiga middle mountains
		Taiga middle mountains
		Low mountains with exposition-dependent forests
Subboreal suboceanic (atlantic)		
	Broadleaved forests	
		No differentiation

Group	Type	Subtype
	Mountain forests (mixed coniferous and broadleaved)	
		Meadow-forested low mountains (with coniferous and broadleaved forests)
		Forested middle mountains (with coniferous and mixed broadleaved and coniferous forests)
	Mountain meadows (subalpine)	
		Meadow middle mountains
Subboreal moderate continental		
	Broadleaved forests	
		No differentiation
	Forest-steppe	
		No differentiation
	Steppe	
		Typical (true) steppes
		Dry steppes
	Mountain meadows	
		Meadow uplands
	Mountain forest-steppes	
		Forest steppe low mountains
	Mountain forests (broadleaved and mixed)	
		Forested low mountains (with broadleaved forests)
		Forested middle and low mountains (with broadleaved and broadleaved and coniferous forests)
	Subnival landscapes	
		Stony subnival uplands (intermediate mountainous landscape)
Subboreal continental		
	Forest-steppe	
		No differentiation
	Steppe	
		Typical (true) steppes
		Dry steppes
	Semideserts	
		No differentiation
	Deserts	
		Northern desert
		Southern desert
	Mountain steppes and deserts	
		Desert-steppe low mountains
		Steppe low mountains
		Meadow-forest and forest-meadow-steppe low mountains (with coniferous, deciduous and broadleaved forests)
		Steppe and desert-steppe middle mountains
	Mountain meadowsteppes and exposition-dependent forests	
		Meadow-forest and forest-meadow-steppe low mountains (with coniferous, deciduous and broadleaved forests)
		Meadow-forest and forest-meadow-steppe middle mountains (with coniferous, deciduous and broadleaved forests)

Group	Type	Subtype
	Subnival landscapes	
		Stony subnival uplands (intermediate mountainous landscape)
	Mountain meadowsteppes (subalpine and alpine)	
		Meadow and meadow-steppe uplands
	Mountain forests (broadleaved)	
		Forested low and middle mountains(with broadleaved forests)
Subarctic suboceanic and oceanic (pacific)		
	Mountain sparse forests and elvinwood	
		Elfin wood and tundra-elfin wood low mountains
		Elfin wood and sparse forest low mountains
	Subarctic tundra	
		Northern tundra
		Southern tundra
	Forest tundra	
		Suppressed and deformed forests and elfin wood
		Subtundra sparse forests
	Mountain tundra	
		Sparse forests and tundra-elfin wood low mountains
		Elfin wood-tundra middle mountains
		Tundra low mountains
		Tundra-bare top middle mountains
Boreal suboceanic (pacific)		
	Mountain tundra and subtundra sparse forests	
		Sparse forests and tundra-elfin wood middle mountains
		Tundra-bare top uplands
	Taiga forests	
		Northern taiga
		Middle taiga
		Southern taiga
		Subtaiga (with mixed deciduous and coniferous forests)
	Mountain taiga forests	
		Taiga low mountains
		Elfin wood and sparse taiga low mountains
		Taiga and sparse taiga middle mountains
		Forested low mountains (with mixed broadleaved and coniferous forests)
Boreal oceanic (pacific)		
	Deciduous forests and meadows	
		No differentiation
	Mountain deciduous forests and elvinwoods	
		Elfin wood low mountains
		Low mountains with deciduous forests
	Mountain tundra and elvinwood	
		Elfin wood-tundra middle mountains
		Meadow-tundra middle mountains
		Tundra-bare top uplands

Group	Type	Subtype
	Taiga forests	
		Subtaiga (with mixed broadleaved and coniferous forests)
		Southern taiga
	Mountain taiga forests	
		Taiga low mountains
		Forested low mountains (with mixed broadleaved and coniferous forests)
Subboreal severe continental		
	Steppe	
		Dry steppes
	Semideserts	
		No differentiation
	Subnival landscapes	
		Stony subnival uplands (intermediate mountainous landscape)
	Mountain meadowsteppes (subalpine and alpine)	
		Meadow-steppe and steppe uplands
	Mountain steppes	
		Desert-steppe low mountains
		Desert-steppe middle mountains
	Mountain tundra and steppe	
		Tundra-steppe and steppe uplands
	Mountain cold deserts	
		Desert uplands
Subboreal suboceanic (pacific)		
	Broadleaved forests	
		No differentiation
	Mountain forests (broadleaved)	
		Forested low mountains (with broadleaved forests)
Subtropical suboceanic (atlantic)		
	Mountain meadows (subalpine)	
		Meadow uplands
	Mountain forests (broadleaved and mixed)	
		Forested low mountains (with broadleaved forests with evergreen understory)
		Forested middle mountains (with mixed broadleaved and coniferous forests)
	Broadleaved liana forests	
		No differentiation
Subtropical moderate continental		
	Forest-steppe	
		No differentiation
	Mountain meadows	
		Meadow uplands
	Mountain forest-steppes	
		"Shiblyak" low mountains
	Mountain forests (broadleaved and mixed)	
		Forested middle and low mountains (with broadleaved and broadleaved and coniferous forests)

Group	Type	Subtype
		Forested low mountains (with Mediterranean forests)
Subtropical continental		
	Semideserts	
		No differentiation
	Mountain meadowsteppes and exposition-dependent forests	
		Meadow-forest and forest-meadow-steppe middle mountains and uplands (with broadleaved forests)
	Subnival landscapes	
		Stony subnival uplands (intermediate mountainous landscape)
	Mountain meadowsteppes (subalpine and alpine)	
		Meadow-steppe uplands
	Mountain deserts, steppe and xerophytic sparse forests	
		Desert-steppe low mountains
		Steppe and desert-steppe middle mountains
		Steppe (semi-savanna) low mountains
		Desert-steppe uplands
		Xerophytic bushes and sparse forests of middle and high mountains

A.2 Geomorphology related levels

Division	Class	Genera
Landscapes of plains(platforms)		
	Plains	
		Accumulative-denudational sea deposits
		Bog accumulative-denudational
		Delta accumulative
		Deluvial-proluvial accumulative
		Deluvial-proluvial accumulative-denudational
		Denudational carst plains
		Denudational plains with hard rock basement
		Denudational stratum plains
		Denudational trappes
		Denudational volcanogenic
		Alluvial accumulative
		Denudational-erosion structural
		Denudational-erosion plains with hard rock basement
		Denudational-erosion stratum plains
		Denudational-erosion structural
		Denudational-erosion trappes
		Denudational-erosion volcanogenic
		Alluvial accumulative-denudational
		Denudational-structural
		Eol accumulative deposits
		Eol accumulative-denudational
		Alluvial-proluvial accumulative
		Volcanic matter plains with hard rock basement
		Fluvioglacial accumulative
		Fluvioglacial accumulative-denudational
		Glacial accumulative
		Glacial accumulative-denudational
		Glacial and fluvioglacial accumulative
		Glacial and fluvioglacial accumulative-denudational
		Alluvial-proluvial accumulative-denudational
		Lake accumulative deposits
		Lake accumulative-denudational
		Lake-alluvial accumulative
		Lake-alluvial accumulative-denudational
		Loess accumulative
		Loess accumulative-denudational
		Mixed origin accumulative
		Mixed origin accumulative-denudational
		Sea accumulative deposits
		Sea-glacial accumulative
		Sea-glacial accumulative-denudational
		Solontchak accumulative
		Solontchak accumulative-denudational

Division	Class	Genera
		Bog accumulative
	Mountains inside platform	
		Denudational-erosion block-flexure
		Denudational-erosional flexure regions
		Blocked denudational-erosion
		Blocked volcanic matter
Landscapes of mountain belts		
	Foothill and intermountain plains	
		Accumulative-denudational sea deposits
		Bog accumulative-denudational
		Deluvial-proluvial accumulative
		Deluvial-proluvial accumulative-denudational
		Denudational plains with hard rock basement
		Denudational stratum plains
		Denudational volcanogenic
		Alluvial accumulative
		Denudational-erosion structural
		Denudational-erosion block-flexure
		Denudational-erosion plains with hard rock basement
		Denudational-erosion stratum plains
		Denudational-erosion volcanogenic
		Alluvial accumulative-denudational
		Denudational-structural
		Eol accumulative deposits
		Eol accumulative-denudational
		Volcanic matter block-flexure
		Alluvial-proluvial accumulative
		Volcanic matter plains with hard rock basement
		Volcanic matter
		Fluvioglacial accumulative
		Fluvioglacial accumulative-denudational
		Glacial accumulative
		Glacial accumulative-denudational
		Glacial and fluvioglacial accumulative
		Glacial and fluvioglacial accumulative-denudational
		Glacial and fluvioglacial
		Alluvial-proluvial accumulative-denudational
		Lake accumulative deposits
		Lake accumulative-denudational
		Lake-alluvial accumulative
		Lake-alluvial accumulative-denudational
		Loess accumulative
		Loess accumulative-denudational
		Mixed origin accumulative
		Mixed origin accumulative-denudational
		Sea accumulative deposits

Division	Class	Genera
		Blocked denudational
		Blocked denudational-erosion
		Blocked volcanic matter
		Bog accumulative
	Mountains	
		Denudational flexure
		Denudational-erosion and volcanic matter (plains with hard rock basement)
		Denudational-erosion and volcanic matter
		Denudational-erosion block-flexure
		Denudational-erosion volcanic
		Denudational-erosion volcanos (plains with hard rock basement)
		Denudational-erosional flexure regions
		Erosion block-flexure
		Erosion flexure
		Erosional block-flexure
		Erosional blocked
		Volcanic matter block-flexure
		Blocked denudational
		Blocked denudational-erosion
		Blocked volcanic matter

A.3 Variants of landscapes

Code	Variant
I	Kola-Karelian
II	East European
III	Middle Siberian
IV	Kazakhstanian
V	Central Asian
VI	Middle Siberian
VII	Novaya Zemlya-Uralian
VIII	Severnaya Zemlya-Taymyrian
IX	Tjan-Shan-Jungarian
X	Altay-Sayanian
XI	Cisbaikalian-Transbaikalian
XII	North-Eastern
XIII	Far East-Daurian
XIV	Carpatian
XV	Crimea-Caucasian
XVI	Kopet-Dagian
XVII	Pamirian
XVIII	Kamchatka-Sakhalinian

Appendix B: Software Used for the Analysis

B.1 General choice of tools

The requirements of this study are far beyond the capabilities of usual end-user software (spreadsheets, databases), thus scripting language was a logical choice.

We need both processing power and flexibility of string processing and highly customized user interface. There are several freely available programs which offer such functionality, namely Tcl, Perl, and Python.

Tcl have several advantages:

- There is a larger amount of tools and extensions available for Tcl than for Python.
- Tcl is fully transportable. There are versions for Unix, Windows, and Macintosh.
- Tcl scripts can be modified during execution. This was essential for debugging word-analysis scripts, because it takes considerable time to collect statistical information about words in the legend.
- We have a GIS extension which allows us to visualize and print maps directly from Tcl script.

Thus, Tcl/Tk was chosen as base language for this study. TkTable extension was also used to provide easy tabular interface for the word-classification programs.

All these tools are freely available and can be downloaded from the internet.

Tcl core is located at `http://www.sunscript.com`.

TkTable extension is at `http://www.cs.uoregon.edu/research/tcl/capp/`.

B.2 User guide

All scripts are designed to simplify debugging and modification. Usually they are organized in two parts, namely startup script and procedure library. The startup script usually contains few lines to load procedure library and desired data files. Procedure library is responsible for all other work. Most of the scripts allow to reload procedure library without loosing already processed data.

B.2.1 `table` script

`table` script performs the part separation (see Section 4.2). It loads the original legend which is kept in the text file with fields separated by the "`|`" character, strips out higher classification levels, and presents results of separation in a tabular form.

The geomorphological part is plotted by black text and the biological part by blue text. By default, the boundary between two parts is in the middle of the screen, although the table can be scrolled in both directions.

Each cell in the table represents a comma-separated part of the description. This part can be moved from one side to another by double-clicking the left mouse button. If you click on the left of the boundary, all cells between the one you click and the boundary will be moved to the right side and vice versa.

This manual correction does not affect the basic heuristics. They are stored in a separate file named `minmax.idx`. This file contains entries only for manually modified rows. It is saved automatically at exit, and can also be saved manually using the "Save" button. At startup this file is read and all stored corrections are applied.

The "Reload" button allows to reread the data file in case of correction of typos.

The “Generate files” button writes the results of the separation into two files `bio.txt` and `geo.txt`, which are used for subsequential word analysis.

There is also the “Goto” entry which allows to enter the number of landscape kinds without manual scrolling of the table. It proved to be useful when errors in the separation step were discovered during the word analysis.

There is a non-interactive version of this script, `regen-bio-geo`, which recreates `bio.txt` and `geo.txt` from `legend` and existing `minmax.idx`. It is intended for correcting typos in the original legend during the word analysis. A common procedure library for these scripts is kept in the file `table.tcl`.

B.2.2 Word analysis scripts

There are actually two scripts which share the common data handling library `readdata.tcl`. `wordanalysis` computes statistical information and allows to analyze and assign classes to words. The procedure library for this script is stored in the file `wordanal.tcl`.

`classify` presents words on a per-class basis and allows to assign values and subclasses (procedure library `classclass.tcl`).

First it computes the statistical information and presents the word table. At the time a row of this table is selected, a small table of word combinations appears in the upper-left corner of the window.

A double-click on the word combination allows to find other words from this combination. The combination can be converted to a term by pressing the appropriate button. Statistics are not recalculated automatically when a new term is defined, due to the considerable time required. The “Recalc” button should be pressed for recalculations.

A middle-button click on the word with original texts of the descriptions where this word can occur pops up in the window.

By clicking the right button on the “class” column of the table a menu of already defined classes pops up and allows to assign one of them to a word. New classes can be defined from the same menu.

You should distinguish between “Reload” and “Reread” buttons. The former reloads the procedure library and the latter reloads data files.

The results of this program are kept in files with the same prefix as the source file. For example, if the data file is `bio.txt`, list of classes is kept in `bio.class`, and classes assigned to words in the `bio.txt.class`. The latter has the form of a Tcl script and is sourced by script on startup and data rereading.

The name of the data file is hardcoded into the startup script. Nothing prevents you from creating several startup scripts for different data files. After all, the startup script is six lines long.

The `classify` script presents classes one at a time and allows to assign values to words. It is mostly done by using standard tkTable key and the mouse function. However, a third column of a table can contain regular expressions. If you enter a value into the second column and a regular expression into the third, and then double-click on value, all words that match this regular expression are given below.

A double click on the word in the first column copies it into both value and regular expression fields, although you probably want to edit both of them before applying.

When you finish entering a value into the second column, you should press the Return key or double-click on this cell, otherwise the value can be lost if you change the word type immediately.

By pressing the right button on the first column a menu of classes pops up and allows to send this word into other class.

Subclasses are usually assigned automatically, by using a predefined set of regular expressions in the file `bio.subclass`. However, it is possible to change subclasses manually by using the pop-up menu.

There are buttons for saving results, reloading data and program, and exiting from the program. The button "Classify" does nothing.

Note: `classify` and `wordanalysis` should not run simultaneously. They modify the same files and each of them can overwrite changes made by the other.

B.2.3 Relative weight assignment

This script (`weights.tcl`) is not interactive. It simply reads data from files prepared by previous scripts and produces the `forlist.txt`.

This file has the following structure:

- Each line is a correct Tcl list.
- The first element of the list is a landscape *kind* number.
- All other elements are two-element lists, with species name (or some reserved word like "non-forest" or "unclassified") as first element and relative weight of this species as second element.

This script also produces a list of used species names and keywords to stdout.

There are several small scripts — `forest.tcl`, `larch.tcl` and `specie.tcl`, which extract information from `forlist.txt` to produce maps. They are all straightforward, so only `forest.tcl` is included here as an example. They produce files with the same names as the script, and extension `.tab` which contains the tcl list of number pairs. The first number in each pair is the landscape number, and the second is the number of legend entry to assign to this polygon.

B.3 Program texts

B.3.1 Part separation

TABLE

```
#!/usr/local/bin/wish4.2
source table.tcl
layout
proc message {msg} {
.menu.progress config -text $msg
update
}
read_all
```

REGEN_BIO_GEO

```
#!/usr/local/bin/tclsh7.6
proc message {msg} {
puts $msg
}
```

```
source table.tcl
read_all
write_files
```

TABLE.TCL

```
#! /usr/local/bin/wish4.2
if {"$tcl_platform(platform)"=="unix"} {
source $env(HOME)/.wishrc
set  font 6x10
} else {
 set font {}
}
package require Tktable
#median - number of boundary column
set median 14
# table event handling
proc go_to {} {
set line [.menu.goto get]
.f.t yview [expr $line - 2]
}
proc toggle_bio {index} {
global tab median max min changed new_changes
set row [.f.t index $index row]
set col [.f.t index $index col]
#puts "th($row) was $th($row)."
if {$col<$median} {
    set shift [expr $median-$col]
    for {set i $max($row)} {$i>=$min($row)} {incr i -1} {
set tab($row,[expr $i+$shift]) $tab($row,$i)
unset tab($row,$i)
    }
    incr max($row) $shift
    incr min($row) $shift
} else {
   set shift [expr $median-$col-1]
   for {set i $min($row)} {$i<=$max($row)} {incr i} {
set tab($row,[expr $i+$shift]) $tab($row,$i)
unset tab($row,$i)
    }
    incr max($row) $shift
    incr min($row) $shift
}
set changed($row) 1
set new_changes 1
}
proc save_results {} {
global  changed min new_changes
if !$new_changes return
set f [open minmax.idx w]
```

```
foreach i [lsort -integer [array names changed]] {
puts $f "$i $min($i)"

}
close $f
set new_changes 0
}

proc save_and_quit {} {
pack forget .f .hs .menu
label .wait -text "Wait please while I'm saving results" -wraplength 160
pack .wait
update
save_results
destroy .
}

proc unquote {text} {
regsub {, *$} $text {} text
set debug 0
#if [regexp "\}" $text] {set debug 1;puts "-------------"}

if $debug {puts $text}
regsub -all "\} \{" $text {, } text
if $debug {puts $text}
regsub -all "\{\{" $text {({} text
if $debug {puts $text}
regsub -all "\}\}" $text {)} text
if $debug {puts $text}
return $text
}

proc write_files {} {
global tab min median max lndid
message "Writing files..."
update
save_results
set bio [open bio.txt w]
set geo [open geo.txt w]
for {set row 1} {$row<=3446} {incr row} {
  set geotext {}
  for {set j $min($row)} {$j<$median} {incr j} {
     append geotext "$tab($row,$j), "
  }
  puts $geo "$lndid($row)|[unquote $geotext]"
  # in real script there is handling of special case of bog landscapes
  # it is omitted because it is highly specific and contain Cyrillic
  # regular expressions
  set biotext ""
```

```
  for {set j $median} {$j<=$max($row)} {incr j} {
     append biotext "$tab($row,$j), "
  }
  puts $bio "$lndid($row)|[unquote $biotext]"
}
close $bio
close $geo
message "Files written"
}

proc read_all {} {
global lndid lineno tab median changed min max new_changes lnd_id index
message "Loading..."
#Reading already saved changes
if [file exists minmax.idx] {
    set f [open minmax.idx]
    while {![eof $f]} {
       set ll [gets $f]
       if [llength $ll]<2 continue
       set min([lindex $ll 0]) [lindex $ll 1]
       set changed([lindex $ll 0]) 1
    }
}
set new_changes 0
#reading legend
array set threshold { 2 2 3 3 4 4 5 4 6 5 7 5}
set f [open ../legend]
set lineno 0
while {![eof $f]} {
 set line "\{[gets $f]\}"
 incr lineno
 regsub -all {\|} $line "\} \{" line
 set lndid($lineno) [lindex $line 0]
 if {$lndid($lineno)>3499} break
 # Do not bother ourselves with unclassified river valleys.
 set line "\{[lindex $line 11]\}"
 # legend has 12 fields
 regsub -all {\(} $line "\{\{" line
 regsub -all {\)} $line "\}\}" line
 regsub -all {, *} $line "\} \{" line
# if [ catch {set count [llength $line]}] {puts "$lineno:$line"}
# if ![info exists table($count)] {
#   set table($count) 1
# } else {
#  incr table($count)
# }
set colno 0
set tab($lineno,0) $lndid($lineno)
set s 1
```

```
if [array exists index] {unset index}
foreach i $line {
# Following regular expression uses Cyrillic "S", not latin "C"
# Semantically, it means "^with .*"
if [regexp "^c .*" $i] {
incr s
set index($s) $colno
}
incr colno
}
if [info exists threshold($s)] {set t $index($threshold($s))} else {
set t $colno}
if {$median<=$t} {set t [expr $median -1]}
if ![info exists min($lineno)] {
set min($lineno)  [expr $median - $t]
}
set i $min($lineno)
foreach s $line {
 set tab($lineno,$i) $s
 incr i
}
set max($lineno) [expr $i -1]

}
close $f
message "Loading complete"
catch {
#This commands work only in graphic environment
.f.t configure -rows $lineno
}
}

proc layout {} {
global median font
frame .menu
button .menu.reload -text "Reload" -command read_all
button .menu.save -text "Save" -command save_results
button .menu.write -text "Generate files" -command write_files
button .menu.quit -text "Exit" -command save_and_quit
pack .menu.reload .menu.save .menu.write -padx 10 -side left
label .menu.goto_label -text Goto:
entry .menu.goto -width 6
pack .menu.goto_label .menu.goto -side left
bind .menu.goto <Key-Return> go_to
label .menu.progress -text ""
pack .menu.progress -side left
pack .menu.quit -padx 30 -side right
pack .menu -expand y -fill x
frame .f
```

```
table .f.t -colwidth 45 -font $font -roworigin 0 -cols [expr 2*$median]\
-rows 1000 -titlecols 1 -yscrollcommand ".f.s set"\
-xscrollcommand ".hs set" -variable tab -anchor w -titlerows 1
for {set i 1} {$i<2*$median} {incr i} {set tab(0,$i) $i}
.f.t width 0 6
scrollbar .f.s -orient vert -command ".f.t yview"
scrollbar .hs -orient horiz -command ".f.t xview"
.f.t tag configure bio -foreground blue -font $font
.f.t tag configure both -foreground red -font $font
.f.t tag configure title -anchor c
.f.t xview [expr $median -3]
for {set i 0;set j $median} {$i<$median} {incr i;incr j} {
   .f.t tag col bio $j
}
pack .f.t -side left -fill both -expand y
pack .f -fill both -expand y
pack .f.s -expand y -fill y -side left
pack .hs -expand y -fill x
bind .f.t <Double-1> "toggle_bio @%x,%y;break"
wm protocol . WM_DELETE_WINDOW save_and_quit
}
```

Frequency analysis

WORDANALYSIS

```
#!/usr/local/bin/wish4.2
source wordanal.tcl
readfile bio.txt
layout
update
fill_tables
```

WORDANAL.TCL

```
#!/usr/local/bin/wish4.2
if {"$tcl_platform(platform)"=="unix"} {
if [file exists ~/.wishrc] {source ~/.wishrc}
set font 8x13
set MyEvent <Button-2>
} else {
set dir $tcl_pkgPath
if [file exists $dir/pkgIndex.tcl] {source $dir/pkgIndex.tcl}
set font {-*-ROL:KOI8/Courier-medium-r-normal--12-*-*-*-*-*-*-*}
set MyEvent <Double-3>
}
package require Tktable

source readdata.tcl

proc process_array {} {
```

```
    global descr wordcount statustext  wordtab table badwords
    set t 0
    catch {unset table}
    catch {unset wordtab}
    foreach i [array names descr] {
set text $descr($i)
set count [llength $text]
set p 0
for {set i 0;set j 1} {$j<$count} {incr i;incr j} {
    set w1 [lindex $text $i]
    set w2 [lindex $text $j]
    if [info exists badwords($w1)] continue
    if [info exists wordtab($w1)] {
incr wordtab($w1)
    } else {
set wordtab($w1) 1
    }
   if {[info exists badwords($w2)]} {
      set index $w1
   } else {
   set index "$w1 $w2"
   }
   incr p
   if [info exists table($index)] {
      incr table($index)
   } else {
       set table($index) 1
   }
       }
       incr t $p
       update
    }
set wordcount [llength [array names wordtab]]
set statustext "Total pairs scanned:$t
Different words found:[llength [array names table]]
words analysed $wordcount"
.u.menu.stat config -height 3 -text $statustext
.u.t config -rows [expr $wordcount+1]
}
proc save_classes {} {
global class workfile wordtab
set f [open $workfile.class w]
foreach {i j} [array get class] {
  if [info exists wordtab([lindex [split $i ","] 1])] {
   puts $f [list set class($i) $j]
  } else { unset class($i) }
}
close $f
}
```

```
proc save_and_exit {} {
save_classes
exit
}

proc fill_tables {} {
  global font class table wordtab tab
  toplevel .process
  wm geometry .process +300+300
  label .process.l -text "Working..." -font $font
  pack .process.l
  label .process.stage -text "Counting words..." -font $font
  pack .process.stage
  update
  process_array
  .process.stage config -text "Sorting words"
  update
  #sort out words
  set list {}
  foreach {word count} [array get wordtab] {
      lappend list [list [format "%05d" $count] $word]
  }
  set j 0
  foreach l [lsort -ascii -dec $list] {
      incr j
      .process.stage config -text "Filling table row $j"
      update
      set word [lindex $l 1]
      regsub {^0*} [lindex $l 0] {} count
      set tab($j,0) $word
      set tab($j,1) $count
      set tab($j,list) [concat [array names table "* $word"]\
      [array names table "$word *"]]
      set tab($j,2) [llength $tab($j,list)]
      for {set i 5} "\$i<[.t.t cget -cols]" {incr i} {
  catch {unset tab($j,$i)}
      }
      foreach cls [array names class "*,$word"] {
  set l [split $cls ","]
  set tab($j,[lindex $l 0]) $class($cls)
      }
      if !$tab($j,2) {
catch {unset tab($j,3)}
catch {unset tab($j,4)}
continue
      }
      set c 0
      foreach pair $tab($j,list) {
if $table($pair)>$c {
```

```
   set c $table($pair)
   set p $pair
}
      }
      set tab($j,3) $p
      set tab($j,4) [format "%6.2f" [expr $c*100.0/$count]]
  }
destroy .process
}

proc show_pairs {index} {
  global pairtab tab table wordtab
  set row [.t.t index $index row]
  set count $tab($row,1)
  set list {}
  foreach pair $tab($row,list) {
    lappend list [list [format "%05d" $table($pair)] $pair]
  }
  set j 1
  foreach tt  [lsort -dec $list]  {
     regsub {^0*} [lindex $tt 0] {} c
     set pair [lindex $tt 1]
     set pairtab($j,0) $pair
     set pairtab($j,1) $c
     set pairtab($j,2) [format "%6.2f" \
    [expr 100.0*$c/$wordtab([lindex $pair 0])]]
     if [llength $pair]>1 {
     set pairtab($j,3) [format "%6.2f" \
     [expr 100.0*$c/$wordtab([lindex $pair 1])]]
     } else {catch {unset pairtab($j,3)}}
     incr j
  }
  .u.t config -rows $j
}
set list {}
# reloads program without destroying global data structures
proc reload {} {
foreach w [pack slaves .] {
 destroy $w
}
set result [catch {uplevel source wordanal.tcl} msg]
set savedInfo errorInfo
uplevel layout
if $result {error $msg $savedInfo}
}

# interface design

proc layout {} {
```

```
global MyEvent font tab pairtab wordcount statustext
if ![info exists wordcount] {set wordcount 1000}
if ![info exists statustext] {set statustext ""}
set tab(0,0) "Word"
set tab(0,1) "Count"
set tab(0,2) "Combinations"
set tab(0,3) "Most frequent"
set tab(0,4) "%"
set tab(0,5) "Class"
set tab(0,6) "Value"
set tab(0,7) "Subclass"
frame .u

frame .u.menu
label .u.menu.stat -width 40 -height 4 -text $statustext
pack .u.menu.stat
frame .u.menu.b
button .u.menu.b.term -text "Term" -command make_term -font $font
button .u.menu.b.repl -text "Replace" -command add_replacement -font $font
button .u.menu.b.calc -text "Recalc" -command fill_tables -font $font
button .u.menu.b.read -text "Reread" -command "re_read; fill_tables"\
     -font $font
pack .u.menu.b.term .u.menu.b.repl .u.menu.b.calc .u.menu.b.read\
      -side left
pack .u.menu.b
pack .u.menu.b -fill x -expand y
frame .u.menu.b1
button .u.menu.b1.save -text "Save" -command save_classes -font $font
button .u.menu.b1.reload -text Reload -command reload -font $font
button .u.menu.b1.find -text "Find" -command find_word -font $font
button .u.menu.b1.exit -text "Exit" -command save_and_exit -font $font
pack .u.menu.b1.save .u.menu.b1.find .u.menu.b1.reload .u.menu.b1.exit\
       -side left

pack .u.menu.b1 -fill x -expand y
pack .u.menu -expand y -fill both -side left
table .u.t -variable pairtab -cols 4 -colwidth 6 -titlerows 1 -height 8\
-yscrollcommand ".u.s set" -anchor e -font $font
.u.t width 0 30
bind .u.t <Double-1> "find_other_word @%x,%y"
set pairtab(0,0) "Word combination"
set pairtab(0,1) "Count
set pairtab(0,2) "% 1st"
set pairtab(0,3) "% 2nd"
.u.t tag configure txt -anchor w
.u.t tag col txt 0
scrollbar .u.s -orient vert -command ".u.t yview"
pack .u.t .u.s -side left -fill y -expand y
pack .u
```

```
frame .t
table .t.t -variable tab -cols 8 -rows [expr $wordcount+1] -height 35\
     -titlerows 1 -yscrollcommand ".t.s set" -anchor w -maxwidth 1024\
     -font $font
if {[winfo screenheight .t.t]<768} {.t.t configure -maxheight 400}
.t.t tag configure num -anchor e
.t.t tag col num  1 2 4
.t.t width 0 20
.t.t width 1 10
.t.t width 2 10
.t.t width 3 30
.t.t width 4 10
.t.t width 5 10
.t.t width 6 10
.t.t width 7 10
bind .t.t <Button-1> "show_pairs @%x,%y"
bind .t.t $MyEvent "show_occurences @%x,%y"
bind .t.t <Button-3> "assign_class @%x,%y %X %Y"
scrollbar .t.s -orient vert -command ".t.t yview"
pack .t.t -side left -expand y -fill both
pack .t.s -side left -expand y -fill y
pack .t -fill both -expand y
load_classes .classmenu 5 bio.class
load_classes .subclassmenu 7 bio.subclass
load_classes .valuemenu 6 bio.values
}

proc list_numbers {word} {
global descr
foreach {i j} [array get descr] {
 if [lsearch -exact $j $word]!=-1 {lappend list $i}
}
if [info exists list] {
return $list
} else { return "No occurences of '$word' found"
}
}

proc show_occurences {index } {
global font tab descr
set word $tab([.t.t index $index row],0)
foreach i [list_numbers $word] {
append text "$i $descr($i)" "\n"
}
if ![winfo exists .show_descr] {
toplevel .show_descr
frame .show_descr.f
text .show_descr.f.t -width 80 -height 14 -xscrollcommand\
     ".show_descr.s set"  -yscrollcommand ".show_descr.f.s set"\
```

```
      -font $font
scrollbar .show_descr.f.s -orient vert -command ".show_descr.f.t yview"
pack .show_descr.f.t -side left
pack .show_descr.f.s -side left -expand y -fill y
pack .show_descr.f
scrollbar .show_descr.s -orient horiz -command ".show_descr.f.t xview"
pack .show_descr.s -expand y -fill x
button .show_descr.close -text Dismiss -command {destroy .show_descr}
pack .show_descr.close
} else {
if {"[wm state .show_descr]"!="normal"} {wm deiconify .show_descr}
.show_descr.f.t delete 0.0 end
raise .show_descr
}
.show_descr.f.t insert 0.0 $text
}

proc find_other_word {index} {
global tab pairtab
set row [.u.t index $index row]
set pair $pairtab($row,0)
set thisword $tab([.t.t index active row],0)
regsub " *$thisword *" $pair {} word
if [llength $word]==2 {error "Pair list doesn't match active word"}
for {set i 1} "\$i<[.t.t cget -rows]" {incr i} {
   if {"$word"=="$tab($i,0)"} {
    .t.t see $i,0
    .t.t activate $i,0
    break
   }
}
}
puts find_other_word
proc make_term {} {
 global descr pairtab workfile terms
 set row [.u.t index active row]
 set pair $pairtab($row,0)
 regsub -all " " $pair "_" term
 foreach i [array names descr] {
   regsub -all $pair $descr($i) $term descr($i)
 }
 set terms($pair) $term
 set f [open $workfile.subst a+]
 puts $f $pair
 close $f
}
puts make_term

proc find_word {} {
```

```
global find_regex font
toplevel .find
wm title .find "Find"
entry .find.e -width 40 -textvariable "find_regex" -font $font
pack .find.e
focus .find.e
bind .find.e <Key-Return> {do_find; destroy .find}
}
puts find_word

proc do_find {} {
global find_regex tab
 if [catch {.t.t index active row} row] {set row 1}
set end [.t.t cget -rows]
incr row
while {$row<$end&&![regexp $find_regex $tab($row,0)]} {incr row}
if {$row<$end} {
 .t.t activate $row,0
 .t.t see $row,0
} else {bell}
}
puts do_find

proc assign_class {index rootx rooty} {
  global tab classified_word class_row class_level column_menus
  set row [.t.t index $index row]
  set class_row $row
  set classified_word $tab($row,0)
  set col [.t.t index $index col]
  if [info exists column_menus($col)] {
    $column_menus($col) post $rootx $rooty
  }
}
puts assign_class

proc put_class {column name} {
global tab classified_word class class_row
set tab($class_row,$column) $name
set class($column,$tab($class_row,0)) $name
}
puts put_class

proc load_classes {menu column filename} {
global column_menus font
catch {destroy $menu}
menu $menu -font $font
$menu add command -label "Cancel" -command {}
$menu add command -label "New class" -command \
   "create_class $menu $column $filename"
```

```
$menu add separator
if ![catch {open $filename} f] {
   while {![eof $f]} {
       set line [gets $f]
       if ![string length $line] continue
       $menu add command -label $line -command [list put_class\
     $column $line]
   }
   close $f
}
set column_menus($column) $menu
}
puts load_classes

proc create_class {menu column filename} {
global font
toplevel .new_class
wm title .new_class "Define new class"
entry .new_class.e -width 40 -font $font
bind .new_class.e <Key-Return> ".new_class.button.ok invoke"
bind .new_class.e <Key-Escape> ".new_class.button.cancel invoke"
frame .new_class.button
button .new_class.button.ok -text "Ok" -command "define_class\
   $menu $column $filename \[.new_class.e get\]; destroy .new_class"\
   -font $font
button .new_class.button.cancel -text Cancel -command\
     {destroy .new_class} -font $font
pack .new_class.button.ok .new_class.button.cancel -side left
pack .new_class.e .new_class.button
}
puts create_class

proc define_class {menu column filename name} {
$menu add command -label $name -command [list put_class $column $name]
set f [open $filename "a+"]
puts $f $name
close $f
put_class $column $name
}

puts define_class
```

READDATA.TCL

```
# read the description and transform it into word list
# should be sourced at global level

# Constant words to skip are leaved out in this listing, because
```

```
# they all Cyrillic
array set badwords {
....
}
proc readfile {filename} {
global terms workfile class
catch {unset terms}
global workfile
set workfile $filename
if ![catch {set f [open $filename.subst]}] {
while {![eof $f]} {
set pair [gets $f]
if ![string length $pair] continue
 regsub " " $pair "_" terms($pair)
}
close $f
}
if [file exists $workfile.class] {source $workfile.class}
re_read
}
proc re_read {} {
global workfile descr terms
set f [open $workfile]
set t 0
while {![eof $f]} {
set list [split [gets $f] |]
if [llength $list]<2 continue
    regsub -all {([^ ])([,;.()])} [lindex $list 1] {\1 \2} text
    regsub -all {([,;.()-])([^ ])} $text {\1 \2} text
foreach t [array names terms] {
  regsub -all $t $text $terms($t) text
}
set descr([lindex $list 0]) $text
}
close $f
}
```

CLASSIFY

```
#!/usr/local/bin/wish4.2
source classclass.tcl
readdata
layout
```

CLASSCLASS.TCL

```
source ~/.wishrc
package require Tktable
proc reload {} {
global errorInfo
```

```
foreach w [pack slaves .] {
 destroy $w
}
rename layout layout_old
set result [catch {uplevel source classclass.tcl} msg]
set savedInfo $errorInfo
if [catch {uplevel layout} msg1] {
 set Info1 $errorInfo
 rename layout_old layout
    foreach w [pack slaves .] {
     destroy $w
    }
 uplevel layout
 error $msg1 $Info1
}
rename layout_old {}
if $result {error $msg $savedInfo}
}

proc save_and_exit {} {
 save
 destroy .
}

proc save {} {
puts Saving
global class
set f [open bio.txt.class w]
foreach {i j} [array get class] {
   puts $f [list set class($i) $j]
}
close $f
}
proc read_subclass {} {
global subclass
set f [open bio.subclass]
catch {destroy .subclassmenu}
menu .subclassmenu
foreach i [split [read -nonewline $f] "\n"] {
  set subclass([lindex $i 0]) [lindex $i 1]
  .subclassmenu add command -label [lindex $i 0] -command\
  "set_subclass [lindex $i 0]"
}

}

proc set_subclass {subclname} {
global tab class curword currow
set tab($currow,3) $subclname
```

```
set class(7,$curword) $subclname
}

proc post_value_menu {x y row } {
global curclass

}

# Reads definition of value from active row and applies
# given regexp to all rows in the table
proc classify {} {
global class tab
set row [.t.t index active row]
set word $tab($row,0)
set value $tab($row,1)
set class(6,$word) $value
.t.t tag cell green $row,1

if [catch {set regexp $tab($row,2)}] return

for {set i [expr $row+1]} "\$i<[.t.t cget -rows]" {incr i} {
set word $tab($i,0)
.t.t tag cell {} $i,1
if [info exists class(6,$word)] {
.t.t tag cell green $i,1
continue
}
if [regexp "^$regexp\$" $word] {
  set class(6,$word) [set tab($i,1) $value]
  .t.t tag cell green $i,1
}
}
}
proc readdata {} {
global class lists subclass
read_subclass
puts "Reading data"
catch {unset class}
source bio.txt.class
recalc_classes
foreach k [array names class 5,*] {
    set word [lindex [split $k ","] 1]
    if ![info exists class(7,$word)] {
foreach {i j} [array get subclass] {
    if [regexp $j $word] {
set class(7,$word) $i
    }
}
    }
```

```
}
}
proc recalc_classes {} {
global class lists
puts "Recalculating classes"
catch {unset lists}
foreach i [array names class 5,*] {
 set word [lindex [split $i ,] 1]
 lappend lists($class($i)) $word
}
foreach i [array names lists] {
set lists($i) [lsort $lists($i)]
}
}

proc reclass_to {new_class} {
global curword tab class oldclass
set word $tab($curword,0)
puts "Reclassing word $word"
puts "Old class $class(5,$word)"
set class(5,$word) $new_class
puts "New class $class(5,$word)"
recalc_classes
set oldclass {}
change_class {} {} w
}
set oldclass {}
proc change_class {name index access} {
global curclass tab lists class oldclass
puts "Changing class to $curclass"
if {"$oldclass"=="$curclass"} {return}

set oldclass $curclass
catch {unset tab}
set j 1
set tab(0,0) ""
set tab(0,1) " "
set tab(0,2) "regexp"
set tab(0,3) ""
foreach i $lists($curclass) {
 set tab($j,0) $i
 if [info exists class(6,$i)] { set tab($j,1) $class(6,$i)
   .t.t tag cell green $j,1
 } else {.t.t tag cell  {} $j,1}
 if [info exists class(7,$i)] { set tab($j,3) $class(7,$i) }
 incr j
}
.t.t config -rows $j
```

```
}
proc menu_man {x y X Y} {
global curclass tab curword currow
set row [.t.t index @$x,$y row];
set col [.t.t index @$x,$y col]
switch $col {
0 {set curword $row;.classmenu post $X $Y}
1 {post_value_menu $X $Y $row}
2 {return}
3 {set curword $tab($row,0);set currow $row;.subclassmenu post $X $Y}
default return
}
}
# Copies word into columns two and three to make final value and
# regular expression from it
proc propagate {index} {
global tab
set row [.t.t index $index row]
set tab($row,2) [set tab($row,1) $tab($row,0)]
}

proc copy {} {
global clipboard
set clipboard [.t.t curvalue]
.top.copy configure -state normal
}

proc paste {} {
global clipboard tab
if ![info exists clipboard] return
foreach index [.t.t curselection] {
set tab($index) $clipboard
set list [split $index ","]
set row [lindex $list 0]
set col [lindex $list 1]
if {$col==1} {set class(6,$tab($row,0)) $clipboard

} elseif {$col==3} {set class(7,$tab($row,0)) $clipboard
}

}

}

proc double_man {index} {
switch [.t.t index $index col] {
0 {propagate $index}
1 {classify}
2 {classify}
```

```
}
}

#button release in table
proc fill_block {} {
global tab
set cells [.t.t cursel]
.t.t flush
if [llength $cells]<=1 return
set value [.t.t get [lindex $cells 0]]
foreach i $cells  {
set tab($i) $value
set list [split $i ","]
set col [lindex $list 1]
if {$col==1} {
 set word $tab([lindex $list 0],0)
 set class(6,$word) $value
}
}
}
proc checkcell {row col} {
if {$col==0} {return 0}
if {$col==1} {.t.t flush}
return 1
}
proc layout {} {
global curclass tab
frame .top
button .top.reread -text "Reread data" -command readdata
button .top.reload -text "Reload program" -command reload
button .top.save -text "Save" -command save
button .top.classify -text "Classify" -command fill_block
frame .t
table .t.t -anchor w -rows 50 -cols 6 -colwidth 20 -yscrollcommand\
       ".t.s set" -var tab -titlerows 1 -selectmode extended\
-validate y -validatecommand "checkcell %r %c"
.t.t tag configure green -background green
scrollbar .t.s -orient vert -command ".t.t yview"
label .top.l -text "Current class"
set f [open bio.class]
set classlist [split [read $f] "\n"]
close $f
trace var curclass w change_class
eval tk_optionMenu .top.classes curclass $classlist
catch {destroy .classmenu}
menu .classmenu
foreach i $classlist {
.classmenu add command -label $i -command "reclass_to $i"
}
```

```
bind .t.t <Button-3> "menu_man %x %y %X %Y"
bind .t.t <Double-1> "double_man @%x,%y"
button .top.exit -text Exit -command save_and_exit
wm protocol . WM_DELETE_WINDOW save_and_exit
pack .top.reread .top.reload .top.save .top.classify .top.l\
.top.classes -side left
pack .top.exit -side right
pack .top -fill x -expand y
pack .t.t -side left
pack .t.s -side left -fill y -expand y
pack .t
}
```

WEIGHTS.TCL

```
#!/usr/local/bin/tclsh7.6
# first, reading of data
source readdata.tcl
set tab(1) 100
set tab(2) {60 40}
set tab(3) {50 30 20}
set tab(4) {50 20 20 10}
set tab(5) {40 20 20 10 10}
set tab(6) {40 15 15 10 10 10}
set tab(7) {30 15 15 10 10 10 10}
set tab(8) {30 15 10 10 10 10 10 5}
set max_level 8

#
# If set to nonzero, script would generate error if forest vegetation
# type found without forest species in its description.
# With zero value it would be processed as ‘‘unclassified’’
#
set suspisious 0
set skip(meaningless) 1
# for a while
set skip(modifier) 1

# Classes which ARE used in forest description
set goodclasses(forest) 1
set goodclasses(multivalue) 1


proc end_level term {
global level
add_list $term
incr level
}
proc add_list term {
global list level value_list
```

```
lappend list($level) $term
if [info exists value_list($term)] {
 incr value_list($term)
} else {
 set value_list($term) 1
}
}
proc subdivide part {
global suspisious class level list goodclasses
set flag 0
set counter 0
foreach word $part {
   if ![info exists goodclasses($class(5,$word))] continue
   switch -exact $class(7,$word) {
       dominant { if $flag {incr level}
  set flag 1
       }
       subdominant {
  if $flag {incr level}
  set flag 0
       }
       default {}
   }
   add_list $class(6,$word)
   incr counter
}
if !$counter {
  if $suspisious {
   return -code error -errorcode PARSE\
         "Something wrong with landcover type: $part"
  } else {
   add_list unclassified
  }
}
}
proc process_descr {descr {trace 0}} {
    global level list badwords class skip weight tab max_level errorCode
    set level 0
    catch {unset list}
    set list(0) {}
    set part {}
    set isforest 0
    foreach word $descr {
      if [info exists badwords($word)] continue
      if [info exists skip($class(5,$word))] continue
      lappend part $word
      if {$class(5,$word)=="forest"} {
   set isforest 1
      } elseif {$class(5,$word)=="types"} {
```

```
if {[lsearch -exact {...}\
    $class(6,$word)]!=-1} {
      set isforest 1
}
         if $isforest {
             if {[catch {subdivide $part} msg]} {
                return -code error -errorcode $errorCode $msg
             }
             incr level
         } else {
           end_level "non-forest"
         }
             set part {}
             set isforest 0
     }
   }
    if {$level==0} {return [list [list non-forest 100.00]]}
    if {$level>$max_level} {
     return -code error -errorcode PARSE "Description too complex: $level"
    }
    catch {unset res}
    for {set i 0} {$i<$level} {incr i} {
if [llength $list($i)]>$max_level {
  return -code error -errorcode PARSE\
                "Description too complex: [llength $list($i)]"
}
set j 0
set w [lindex $tab($level) $i]
set tt $tab([llength $list($i)])
foreach k $list($i) {
     set ww [expr [lindex $tt $j]*$w]
     if [info exists res($k)] {
set res($k) [expr $res($k)+$ww]
     } else {
set res($k) $ww
     }
             incr j
}
    }

    set r ""
    foreach k [array names res] {
       if {$k!="non-forest"} {
       lappend r [list [format "%8.2f" [expr $res($k)/100.0]] $k]
      }
    }
    set result {}
    foreach w [lsort -decr $r] {
       lappend result [list [lindex $w 1] [string trim [lindex $w 0]]]
```

```
    }
    if [info exists res(non-forest)] {
       lappend result [list non-forest [format "%.2f" \
                     [expr $res(non-forest)/100.0]]]
    }
    if ![llength $result] {
       set result [list [list non-forest 100.00]]
    }
    return $result
}
proc scan_all {} {
global value_list errorInfo descr errorCode
catch {unset value_list}
set f [open "forlist.txt" w]
puts stderr "Processing descrtiptions"
set errors {}
foreach i [lsort -integer [array names descr]] {
 #if {$i%100==0} { puts -nonewline stderr "\rProcessing $i";flush stderr}
 if [catch {process_descr $descr($i)} res] {
    if {$errorCode=="PARSE"} {
      puts stderr "$i:$res"
      lappend errors $i
    } else {
      puts stderr $res
      puts stderr $errorInfo

      exit
    }
 } else {
    puts $f [concat $i $res]
 }
}
close $f
parray value_list
}
puts stderr "Reading data"
readfile bio.txt
scan_all
```

FOREST.TCL

```
#!/usr/local/bin/tclsh7.6
set f [open forlist.txt]
proc incrcount {index} {
global count
if [info exists count($index)] {
  incr count($index)
} else {
  set count($index) 1
}
```

```
}
set species non-forest
while {![eof $f]} {
   set list [gets $f]
   if [eof $f] continue
   if [set index [lsearch -glob $list "$species *"]]!=-1 {
     set weight [lindex [lindex $list $index] 1]
     if {$weight==100} {set class 1
       incrcount 100
     } else { set class [expr int(floor((100-$weight)/20))+2]
     incrcount $weight
    }
   } else {
     set class 6
     incrcount 120

   }
   incrcount $class
   lappend table [list [lindex $list 0] $class]
}
close $f
for {set i 3500} {$i<3998} {incr i} {lappend table [list $i 7]}
lappend table {3998 253} {3999 254}
set f [open forest.tab w]
puts $f [list set foresttab $table]
close $f
foreach i [lsort -real [array names count]] {
 puts [format "%6.2f %6d" $i $count($i)]
}
set sum 0
foreach i {1 2 3 4 5 6} {
 incr sum $count($i)
}
puts $sum
```