GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Masterarbeit

Web-basierte Visualisierung von Klimaszenarien

Web-based Visualization of Climate Scenarios

Sabine Flügel

Studienrichtung: Forstwissenschaften,
Waldökosystemanalyse und Informationsverarbeitung

Gutachter:

Prof. Dr. Winfried Kurth
(Universität Göttingen)

Dr. Jürgen Nagel
(Nordwestdeutsche forstliche Versuchsanstalt)

28. Juli 2010

Abstract

Climate change happens and will affect growth and survival of tree species. To prevent silviculture from ecological and economical disasters, it is necessary to quantify future climate conditions and estimate their silvicultural importance. The web-based Decision Support System “Forest and Climate Change” (DSS WuK) deals with the development of adaption strategies for a sustainable forest management under climate change in Germany, considering the main trees species oak, beech, Norway spruce, Douglas fir and Scots Pine. Users of the DSS-WuK should not only be given advice about the treatment of forest stocks, but should also understand the underlying models and data. Therefore the main objective of this thesis is the web-based visualization of climate variables with silvicultural importance like the mean daily temperature and sum of precipitation within and beyond the growing season and the length of the growing season to enable Users of the DSS-WuK an easy understanding of the underlying climate data and to help them drawing their own conclusion with respect to changing site conditions.

A set of Open Source Software is used to pre-process prospective climate data derived from global and local climate models using the emission scenarios A1B and B1. The database management system PostgreSQL is needed to aggregate approximately 868 Mio datasets. Further it helps to calculate the length of the growing season which is tree species-dependant. Out of these aggregated datasets raster images are produced and included into UMN Mapserver. The JavaScript-Libraries OpenLayers, Ext JS and GeoExt are used to build a web-mapping application presenting maps received from UMN Mapserver. By the use of this web-mapping application the User will be able to investigate the change of climate variables in four different climate periods, beginning in 1971 and ending in 2100, all over Germany. To present a yearly chronological sequence of the climate variables for every grid cell the Web-Development-Framework Django is used to access the database and select all datasets belonging to a User-specified coordinate. Django processes these datasets and delivers them to JavaScript Function based on the Google Visualization API which in turn draws a line chart. By retrieving a set of such line charts including all climate variables, scenarios and tree species a User will find himself able to analyze the climate change and its influence on the main tree species at a specific site. The developed web-application is easy to use, simple to understand and provides all the information necessary for gaining insight to the data the DSS-WuK builds its suggestions on and for a better understanding of changing site conditions.

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Jahrtausendelange Klimaänderungen (BLÜMEL 2002) und menschliche Einflüsse prägen die heute in Deutschland anzutreffenden Waldlandschaften (SCHULZE et al. 2010). Dieses Zusammenspiel hat sich bis heute nicht verändert, wohl aber der Einfluss des Klimas auf den Menschen. Wo noch vor einigen hundert Jahren die Bevölkerung durch klimatische Verhältnisse zum Verlassen bestimmter Landstriche gezwungen war (BLÜMEL 2002), ist die Gesellschaft heute dermaßen fortschrittlich, dass negative Klimawirkungen in vielen Bereichen erfolgreich bekämpft werden können. In der Landwirtschaft kann beispielsweise mit Gewächshäusern gearbeitet werden, um optimale Wuchsbedingungen zu schaffen und schädigende Witterungseinflüsse zu umgehen.

Die Möglichkeit, auf Klimaänderungen reagieren und ihnen standhalten zu können, muss auch in der Forstwirtschaft ausgenutzt werden. Laut KÖLLING et al. (2008) besitzen Wälder das Potenzial sich im Zuge klimatischer Änderungen umzustrukturieren, was sich jedoch auf den Zustand und die Gesundheit des Waldes auswirken würde. Weiterhin sollen sie nicht „den Anforderungen der Forstwirtschaft [und] der Volkswirtschaft […] genügen".

Dass sich das Klima auch in jüngster Zeit im Wandel befindet, zeigen Temperaturmessungen seit 1850. Erst während des 2. Drittels des 20. Jahrhunderts treten globale Temperaturanstiege auf (CLIMATE RESEARCH UNIT 2008) und erst seit kurzem ist wohl bekannt, dass die Erhöhung der Globaltemperatur nicht zu einer natürlichen Schwankung gehört, sondern dauerhaft bleiben wird. Auch das Niederschlagsverhalten wies in den letzten Jahrzehnten auf allen Teilen der Erde Anomalien auf (WORLD METEOROLOGICAL ORGANIZATION 1997), die darauf schließen lassen, dass ein klimatischer Umbruch im Gange ist. Auch projizieren verschiedenste Klimamodelle und Szenarien recht deutliche klimatische Veränderungen innerhalb einer kurzen Zeitspanne.

Während die einen dem Klimawandel gelassen entgegensehen und davon ausgehen dass „die Forstwirtschaft […] eher von den zu erwartenden Änderungen profitieren" kann (KROPP et al. 2009), wird an anderer Stelle die Anfälligkeit der Baumarten gegenüber einer Klimaänderung hervorgehoben und „neue Planungsgrundlagen und Entscheidungsmodelle" werden gefordert (AMERELLER et al. 2009).

Im Zuge des Klimawandels hat das Bundesministerium für Bildung und Forschung (BMBF) die Fördermaßnahme „klimazwei – Forschung für den Klimaschutz und Schutz vor Klimawirkungen“ ins Leben gerufen. Dabei werden zum einen Projekte gefördert, in deren Rahmen Möglichkeiten untersucht werden, wie die Emmission klimaerwärmender Gase reduziert werden könnte (INSTITUT DER DEUTSCHEN WIRTSCHAFT KÖLN 2010). Diese Projekte stammen aus den unterschiedlichsten Bereichen, z.B. werden Low-Emmission-Ships entwickelt und Biotreibstoffe untersucht etc. (MAHAMMADZADEH 2009). Zum anderen werden Anpassungsstrategien an den bereits eingetretenen und auch weiterhin nicht mehr aufzuhaltenden Klimawandel entwickelt. Hierfür geförderte Projekte befassen sich z. B. mit einem nachhaltigen Grundwassermanagement, regionalen Akteuren, die in Bezug auf Klimaschutz und Anpassung an den Klimawandel mobilisiert werden sollen, Tourismusentwicklung, etc. (MAHAMMADZADEH 2009).

Das Projekt „DSS-WuK – Decision Support System Wald und Klimawandel: Anpassungsstrategien für eine nachhaltige Waldbewirtschaftung unter sich wandelnden Klimabedingungen“ zählt ebenfalls zu den Projekten, die sich mit den Anpassungsstrategien befassen. Im Rahmen des Projektes wird ein web-basiertes Decision Support System entwickelt, welches die kleinräumigen standörtlichen Veränderungen, die im Zuge des Klimawandels zu erwarten sein werden, berücksichtigt. Daraus werden ökonomische und waldbauliche Einschätzungen für die fünf Hauptbaumarten Eiche, Buche, Fichte, Douglasie und Kiefer abgeleitet (JANSEN et al. 2008).

Die Notwendigkeit dieses Projektes ergibt sich daraus, dass der Standortfaktor Klima, der bisher als konstant angenommen wurde, ab sofort in all seiner Variabilität erfasst werden und umgehend in waldbauliche und ökonomische Entscheidungen miteinbezogen werden muss. In einigen deutschen Forstbetrieben wurden bereits Waldumbauprogramme initialisiert. Als Beispiel kann hier die bayerische Forstwirtschaft genannt werden, der mehrere Millionen Euro für den Waldumbau zur Verfügung gestellt wurden (HAHN 2009). Durch Nutzung des DSS-WuK lassen sich diese Waldumbauprogramme bezüglich der Baumartenwahl noch optimieren, bzw. können deutschlandweit Standorte auf ihre Notwendigkeit zum Umbau überprüft werden.

1.1 Zielstellung

Ziel dieser Arbeit ist die web-basierte Visualisierung von Klimavariablen, die es den Nutzern des DSS-WuK erleichtert, die ökonomischen und ökologischen Einschätzungen nachzuvollziehen. Außerdem wird es den Nutzern ermöglicht, eigene Schlüsse aus den zukünftigen klimatischen Bedingungen zu ziehen, so dass sie sich nicht blind auf die durch das DSS-WuK gelieferten Einschätzungen verlassen müssen.

Dargestellt werden forstlich relevante Klimaparameter, die einen Bezug zu den im DSS-WuK implementierten Modellen aufweisen. Die Klimadaten werden entsprechend aufbereitet und in einer Datenbank vorgehalten. Die visuelle Darstellung im Browser soll zum einen deutschlandweite Übersichtskarten umfassen, wobei keine baumartenspezifischen Unterschiede sichtbar sein und über Klimaperioden gemittelte Werte gezeigt werden. Zum anderen sollen für jeden möglichen Punkt in Deutschland detaillierte Informationen abrufbar sein. Dabei werden Liniendiagramme mit jährlichen Mittelwerten erstellt, so dass sich Unterschiede verschiedener Klimaszenarien und Baumarten erkennen lassen.

1.2 Relevante Klimavariablen

Die in dieser Arbeit entwickelten Visualisierungen sollen Forstpraktikern helfen, die Auswirkungen des Klimawandels auf ihre Bestände besser zu verstehen. Die interaktiven Karten und Liniendiagramme werden in die Internetplattform DSS-WuK eingebunden. Dementsprechend müssen Klimavariablen für die Darstellung ausgewählt werden, die für den Forstpraktiker aussagekräftig sind und in Beziehung zu den im DSS-WuK verwendeten Modellen stehen.

Bereits vor 8200 Jahren führte eine Abkühlung des europäischen Klimas um 1.7°C der jährlichen Durchschnittstemperatur zu einer veränderten Baumartenzusammensetzung (TINNER und LOTTER 2001). Auch heute beeinflusst die Temperatur das Verhalten der Bäume. Nach FANG und LECHOWICZ (2006) hängt die geografische Verteilung der Buche vorrangig von der in der Vegetationsperiode verfügbaren Wärmemenge ab, wobei auch hohe Wintertemperaturen das Buchenvorkommen mitbestimmen können, da sie sich auf den Austrieb und die Konkurrenzkraft auswirken. Nach SOLBERG et al. (2009) beeinflussen mittlere Temperaturen von Mai bis

August den Zuwachs von Kiefer und Fichte, welcher jedoch auch auf Veränderungen in der Länge der Vegetationszeit zurückzuführen sein kann. Ein weiteres erwähnenswertes Temperaturereignis stellen Spätfröste dar. Bei der Buche haben sie, sofern sie nach dem Blattaustrieb stattfinden, erheblichen negativen Einfluss auf den Zuwachs (DITTMAR et al. 2006).
REICH und OLEKSYN (2008) zeigen, dass Wachstum und Überleben der Kiefer hauptsächlich von der Jahresmitteltemperatur beeinflusst werden. Laut JUMP et al. (2006) geht bei unverändertem Jahresniederschlag und gleichzeitig einer Erhöhung der Jahresmitteltemperatur der Zuwachs der Buche zurück. JUMP et al. schließen, dass die dadurch entstehende Trockenheit für die Zuwachsverluste verantwortlich ist.
Den Zusammenhang zwischen Temperatur und Niederschlag machte sich KÖLLING (2007) zunutze und leitet Klimahüllen für verschiedene Baumarten ab, bei denen lediglich aufgrund von Jahresniederschlag und Jahresmitteltemperatur auf das Vorkommen einer Baumart geschlossen werden kann. Diese äußerst simple Form einer Klimahülle wurde stark kritisiert (BOLTE et al. 2008, PEARSON und DAWSON 2003). Hauptkritikpunkt ist, dass das Überleben und die Produktivität einer Baumart nicht von Jahresmittelwerten abhängt. Vielmehr bestimmen Witterungsextreme wie die bereits angesprochenen Spätfröste, Trockenheit, Hitze (BOLTE et al. 2008) oder Stürme (BOLTE und IBISCH 2007) das Vorkommen der Baumarten. Auch LINDNER et al. (2010) zeigen, dass Stürme, Trockenheit, Überflutungen und Hitzewellen die meisten Gefahren für Bäume bergen, wobei sie sich hierbei eher auf maritime Gebiete beziehen. Die Auswirkungen extremer Trockenheit waren im Jahr 2003 zu spüren, als in Bayern frühzeitiger Laubabwurf einsetzte (RASPE et al. 2004).

Es zeichnet sich ab, dass Temperatur, Niederschlag und Wind die bedeutendsten klimatischen Standortfaktoren für das Baumwachstum darstellen.
Der **Niederschlag** ist primär für das pflanzenverfügbare Bodenwasser verantwortlich. Benötigt wird Wasser von den Bäumen jedoch nur während der Vegetationsperiode. Für Mitteleuropa wird eine Abnahme der Sommerniederschläge und eine Zunahme der Winterniederschläge für möglich gehalten (ROECKNER et al. 2006), also eine Verschiebung des Hauptniederschlags aus der Vegetationsperiode in den Winter. Um abzuschätzen, mit welchen Graden an Trockenheit während der Vegetationsperiode zu rechnen ist, lohnt sich die Betrachtung des gesamten in der Vegetationsperiode auftretenden Niederschlags. Außerdem ist interessant, wie die Nieder-

schläge verteilt sind, um Aufschluss darüber zu erhalten, ob eine konstante Wassernachlieferung über die Niederschläge gewährleistet ist.

Nach Beendigung der Vegetationsperiode muss der Bodenwasservorrat in Vorbereitung auf die nächste Vegetationsperiode wieder aufgefüllt werden. Dies geschieht über den im Winter anfallenden Niederschlag. Reichen die Niederschlagsmengen nicht aus, um den Bodenwasservorrat zu maximieren, so ist dieser im Laufe der nächsten Vegetationsperiode schneller erschöpft und es wird früher und häufiger zu Trockenstresssituationen kommen. Niederschlagssummen außerhalb der Vegetationsperiode eignen sich, um diesen Sachverhalt darzustellen.

Die Einflüsse der **Temperatur** auf die Baumarten werden künftig deutlich zu spüren sein, da eine Temperaturerhöhung bereits im Gange ist (vgl. 1.1). Dabei ist anzunehmen, dass gerade die Temperaturverhältnisse in der Vegetationsperiode am meisten ins Gewicht fallen, denn steigende Temperaturen beeinflussen die Wasserverfügbarkeit durch höhere Verdunstungsraten negativ. Der Gesamtvorrat an pflanzenverfügbarem Wasser wird durch das Gesamtaufkommen an Wärme beeinflusst und nicht durch einzelne, sehr heiße Tage. Nun lässt sich darüber streiten, ob die Jahresmitteltemperatur oder die mittlere Temperatur in der Vegetationsperiode am besten geeignet ist, um das Ansteigen der Wärmemenge anzuzeigen. Wird aber gleichzeitig eine Betrachtung des Niederschlages in der Vegetationsperiode vorgenommen, lässt sich ein Überblick über die zu erwartende Trockenheit erhalten.

In Mitteleuropa soll „die mittlere Stärke der Winterstürme um etwa 10%“ zunehmen (ROECKNER et al. 2006). Für die mittlere **Wind**geschwindigkeit in Nordrhein-Westfalen werden dagegen in den nächsten Jahrzehnten nur sehr geringe Änderungen projiziert, die „klimatisch keine Bedeutung“ haben (GERSTENGARBE et al. 2004). Außerdem konnten im letzten Jahrhundert keine Langzeittrends für Windgeschwindigkeiten über 20m/s festgestellt werden und die Häufigkeit von Gewittern, die auch mit heftigen Windböen verbunden sind, nahm sogar ab (HEINO et al. 1999). Für steigende Sturmschäden lassen sich nicht unbedingt steigende Windgeschwindigkeiten verantwortlich machen, sondern Landnutzungsänderungen, Veränderungen des Bestandesalters und der Bestandesstruktur und auf klimatischer Ebene auch milde Winter, die dazu führen, dass die Böden nicht mehr so häufig gefroren sind und dadurch die Verankerung der Bäume geschwächt wird (SCHLYTER et al. 2006).

Nun spielen mittlere Windgeschwindigkeiten kaum eine Rolle in Bezug auf etwaige Windschäden. Wenn sich ein Zusammenhang zwischen der mittleren Windgeschwindigkeit und den Minimal- und Maximalwerten unterstellen ließe, können die Änderungen der Windgeschwindig-

keit in Nordrhein-Westfalen darauf hindeuten, dass es zu keiner oder nur einer sehr geringen Zunahme an hohen Windgeschwindigkeiten kommen wird. Dass sich im letzten Jahrhundert keine signifikanten Veränderungen hoher Windegeschwindigkeiten feststellen ließen, könnte ebenfalls darauf hindeuten, dass auch weiterhin keine oder nur leichte Veränderungen eintreten. Die in Zukunft zu erwartenden Sturmschäden dürften demzufolge eher andere Ursachen haben, wie der bereits erwähnte Rückgang der Bodenfröste. Dieser lässt sich gut durch steigende winterliche Temperaturen abbilden.

Eine bedeutende, aber bisher noch nicht angesprochene Gefahr für alle Baumarten ist der Schadinsektenbefall. Beeinflusst wird das Auftreten von Schadinsekten allgemein durch Windschäden und Witterungsverhältnisse wie Temperatur und Niederschlag. Doch die artspezifischen Einflüsse des Klimas auf Schadinsekten sind noch weitgehend unbestimmt (PETERCORD et al. 2008). Daher wird dieser Faktor nicht weiter berücksichtigt.

Aufgrund dieser Überlegungen wurde entschieden, folgende Klimavariablen im Visualisierungsprozess zu verwenden:

- die Länge der Vegetationsperiode (length)
- die Tagesmitteltemperatur innerhalb der Vegetationsperiode (temp_vp)
- die Tagesmitteltemperatur außerhalb der Vegetationsperiode (temp_nvp)
- die Niederschlagssumme innerhalb der Vegetationsperiode (prec_vp)
- die Niederschlagssumme außerhalb der Vegetationsperiode (prec_nvp)

Die Länge der Vegetationsperiode ebenfalls darzustellen bietet sich an, da die Länge der Vegetationsperiode die Niederschlagssummen beeinflusst. Z. B. würde eine Verlängerung der Vegetationsperiode zu höheren Niederschlagssummen in der Vegetationsperiode führen. Eine gemeinsame Betrachtung dieser Größen ist also empfehlenswert.

1.2.1 Berechnung der Vegetationsperiode

Als Vegetationsperiode könnten prinzipiell feste Zeiten angenommen werden (z.B. Mai-September). Da die Länge der Vegetationsperiode jedoch von klimatischen Bedingungen (in gängigen Modellen von der Temperatur) abhängt, kann eine Änderung der Länge der Vegetationsperiode erwartet werden. SPEKAT et al. (2006) erwarten z. B. eine Verlängerung der Vegetationsperiode in Nordrhein-Westfalen um ca. 14 Tage.

Im DSS-WuK wird die Länge der Vegetationsperiode ebenfalls nicht als konstant angenommen. Der Beginn der Vegetationsperiode wird nach einem Ansatz von MENZEL (1997) berechnet und das Ende geht auf einen Ansatz von VON WILPERT (1990) zurück. Diese beiden Modelle werden auch in dieser Arbeit verwendet.

Beginn der Vegetationsperiode

Der Zeitpunkt des Blattaustriebs hängt von dem Kältereiz (gemessen in der Anzahl an Kältetagen) ab, der im Winter erfahren wird. Ein starker Kältereiz, d.h. viele Kältetage, führt zu einem früheren Blattaustrieb, während wenig Kältetage zu einer Verzögerung des Blattaustriebes beitragen. Über den Kältereiz lässt sich eine kritische Wärmesumme bestimmen, die für den Beginn des Blattaustriebes erreicht werden muss (Formel 1).

$$TS_{crit} = a + b \cdot \ln(CD) \qquad (1)$$

mit TS_{crit} Wärmesumme, die zum Austrieb benötigt wird

CD Anzahl Kältetage ab 01.11. des Vorjahres bis Beginn des Blattaustriebes (Tabelle 1)

a, b baumartenspezifische Konstanten (Tabelle 1)

Die Berechnung der tatsächlichen Wärmesumme beginnt ab dem 01. Februar (Formel 2).

$$TS = \sum_{1.2.}^{n} T_{mittel} - T \qquad \text{wenn } T_{mittel} > T \qquad (2)$$

mit TS tatsächliche Wärmesumme

T_{mittel} Tagesmitteltemperatur

T baumartenspezifischer Schwellenwert (Tabelle 1)

Sobald die tatsächliche Temperatursumme die kritische Wärmesumme erreicht, beginnt der Blattaustrieb.

Tabelle 1: Parameter (fortan bezeichnet als Menzelparameter) für die Temperaturschwellen und Konstanten für den Beginn der Vegetationsperiode (aus: VON WILPERT 1990)

	Temperatur-schwelle Kältetage	Temperatur-schwelle Wärmesumme	a	b
Eiche	9	4	1748	-298
Buche	9	6	1922	-348
Fichte/ Douglasie	9	4	1848	-317
Kiefer	9	5	1395	-223

Ende der Vegetationsperiode

VON WILPERT (1990) geht davon aus, dass es nach dem 5. Oktober keine Kambiumaktivität mehr gibt und das Wachstum auch unter ansonsten „optimalen Wuchsbedingungen aufgrund der Tageslänge eingestellt wird“. Außerdem wird das Wachstum eingestellt, wenn das 7-tägige gleitende Mittel der Tagesmitteltemperaturen 5 mal unter 10°C liegt. Das Wachstum kann danach auch wieder aufgenommen werden, wenn das 7-tägige gleitende Mittel der Tagesmitteltemperaturen 5 mal über 10°C liegt. Im DSS-WuK wird mit der Suche nach dem Ende der Vegetationsperiode ab dem 01. August begonnen und die Vegetationsperiode endet, sobald das 7-tägige gleitende Mittel der Tagesmitteltemperaturen 5 mal unter 10°C liegt oder der 05. Oktober erreicht wird. Eine temperaturbedingte Wiederaufnahme des Wachstums ist jedoch nicht vorgesehen.

2 Daten

Wetter- und Klimaforschung begann bereits vor mehreren Hundert Jahren (LE TREUT et al. 2007). Als sich abzeichnete, dass ein Wandel des Klimas im Gange ist, wurde 1988 das International Panel on Climate Change (IPCC) gegründet, zu dessen Aufgaben es unter anderem gehören sollte, noch bestehende Wissenslücken bezüglich des Klimawandels zu schließen (IPCC 2004). Außerdem sollte der menschliche Einfluss auf das Klima und die damit zusammenhängende Änderung des Klimas abgeschätzt werden. Dies führte zu der Entwicklung von möglichen Emissionsszenarien auf deren Grundlage Projektionen des zukünftigen Klimas durchgeführt werden konnten. Bei der Erarbeitung der Emissionszenarien wurden zu Beginn vier sogenannte Storylines verfasst, die beschreiben, wie sich die Zukunft auf demografischer, sozialer, ökonomischer, technologischer und politischer Ebene entwickeln könnte (IPCC 2000). Da es nicht möglich war, einen kurzen, prägnanten Namen für jede Storyline zu finden, gab man ihnen die Namen A1, A2, B1 und B2.

Die Storylines A1 und A2 stellen die ökonomische Orientierung und das Streben nach Reichtum in den Vordergrund. Ökologische Aspekte werden eher vernachlässigt. Diese werden dafür in den B1 und B2 Storylines berücksichtigt, da auf globaler Ebene Umweltbewusstsein entwickelt und ökologische Nachhaltigkeit angestrebt wird.

Globalisierung und die Annäherung der Sozialprodukte und Pro-Kopf-Einkommen zwischen entwickelten Ländern und Entwicklungsländern ist in die Storylines A1 und B1 eingearbeitet. Die Möglichkeit, dass sich unterschiedliche Regionen der Welt in verschiedene Richtungen entwickeln können, liegt den Storylines A2 und B2 zugrunde.

Um tatsächliche Ausprägungen im Rahmen dieser Storylines zu quantifizieren, wurden 6 Computermodelle genutzt um 40 verschiedene Szenarien zu ermitteln. Diese decken einen großen Bereich dessen ab, was an zukünftigen Treibhausgasemissionen möglich ist. Sie sind in Anlehung an die Storylines zu Szenarienfamilien zusammengefasst (A1, A2, B1, B2). In der Familie der A1-Szenarien existieren die drei Untergruppen A1FI (Energieversorgung aus fossilen Rohstoffen), A1T (Energieversorgung aus nicht fossilen Rohstoffen) und A1B (ausgeglichene Energieversorgung aus fossilen und nicht-fossilen Rohstoffen). In den 40 Szenarien kommt es zwischen 1990 und 2100 zu kumulativen CO_2-Emissionen von 773-2538 Gigatonnen Kohlenstoff (GtC).

Für jede Szenarienfamilie wurde ein Marker-Szenario ausgewählt, welches die Charakteristiken der jeweiligen Szenarienfamilie am Besten wiedergibt. Diese Marker-Szenarien bilden jeweils die Grundlage für die Klimaprojektionen. Die kumulativen CO_2-Emissionen belaufen sich im Marker-Szenario für A1B auf 1499 GtC und für B1 auf 983 GtC.

Die in dieser Arbeit verwendeten globalen Klimaprojektionen wurden mit ECHAM5/MPI-OM durchgeführt. ECHAM5 und MPI-OM wurden am Max-Planck Institut für Meteorologie entwickelt. Es handelt sich bei ECHAM5/MPI-OM um ein globales Atmosphärenmodell und ein Ozean-Zirkulationsmodell, die aneinander gekoppelt wurden. ECHAM5 besitzt eine horizontale Auflösung von ca. 200km und die Auflösung von MPI-OM variiert zwischen 10-150km (HOLLWEG et al. 2008). Für eine differenzierte Betrachtung kleinräumiger klimatischer Gegebenheiten werden die globalen Projektionen herunterskaliert („Downscaling“).

Im DSS-WuK werden Klimaprojektionen verwendet, die mit dem Climate Local Model (CLM) herunterskaliert wurden. Das CLM basiert auf dem Local Model (LM) des Deutschen Wetterdienstes (DWD) und wurde von der CLM-Community (ein internationales Netzwerk von Wissenschaftlern und wissenschaftlichen Einrichtungen) entwickelt. Angetrieben wurde es bisher mit dem Output von ECHAM5/MPI-OM für die Szenarien A1B und B1. Die herunterskalierten Projektionen liegen zum einen in einem rotierten Gitter (Datenstrom 2) und zum anderen in einem geografischen Gitter (Datenstrom 3, Auflösung 0.2°) für Europa vor. Die Daten werden in einer Datenbank über ein Internetportal zur Verfügung gestellt und lassen sich durch Angabe von Koordinaten für einen bestimmten Teil von Europa herunterladen. Für die Visualisierung der oben genannten Klimavariablen werden mittlere Lufttemperaturen in 2m Höhe sowie Niederschlagssummen auf Tagesbasis aus Datenstrom 3 benötigt. Die Daten existieren zum einen für die Szenarien A1B und B1 aus jeweils zwei Simulationsläufen für den Zeitraum 2001-2100 und zum anderen für das 20. Jahrhundert aus 3 Simulationsläufen von 1960-2000.

Das Format aller Daten ist NetCDF. Dieses Format enthält Dimensionen (z.B. Zeit, Längengrad…), Variablen (Array mit Daten gleicher Typen zur Speicherung des Hauptanteils an Daten) und Attribute (Metadaten) (REW et al. 2010). Aus diesem Format werden die Datensätze extrahiert, umorganisiert und in einer Datenbank in Form von Tabellen (Anhang A) abgelegt.

3 Struktur der Anwendung

3.1 Datenhaltung

Sämtliche Klimadaten werden in einer PostgreSQL-Datenbank gehalten. Bei PostgreSQL handelt es sich um ein objektrelationales Datenbankmanagementsystem (HARTWIG 2001) und gilt in diesem Bereich als eines der erfolgreichsten Open-Source Produkte (MATTHEW und STONES 2005).
Bereits im Jahr 1977 wurde an der Universität Berkeley mit der Entwicklung des Datenbankmanagementsystems Ingres begonnen, welches in späteren Jahren zu der Entwicklung von PostgreSQL führte (HARTWIG 2001). Zurzeit liegt es in der Version 8.4.3 vor. Die grafische Benutzeroberfläche pgAdmin III ermöglicht einen schnell erlernbaren Umgang mit PostgreSQL. Außerdem besitzt PostgreSQL Schnittstellen zu diversen Programmiersprachen wie C, C++, ODBC, Java, Perl, Tcl/TK, Python und HTML (MOMJIAN 2001). Durch die Erweiterung PostGIS unterstützt PostgreSQL die Arbeit mit Geodaten. PostGIS enthält über 3000 räumliche Referenzsysteme und erlaubt die Speicherung von Geometrien wie Punkte, Linien, Polygone, Multipolygone etc. Zahlreiche räumliche Abfragen und Operationen sind möglich.

Eine Datenbank lässt sich als eine „Sammlung von Relationen“ betrachten. Dabei kann man sich eine Relation als Tabelle vorstellen. Es gibt Zeilen, die als Tupel und Spalten, die als Attribute bezeichnet werden (ELMASRI und NAVATHE 2002). Tupel werden in dieser Arbeit auch als Datensätze bezeichnet. Jedes Tupel repräsentiert eine Entität („Objekt oder Konzept aus der realen Welt“ (ELMASRI und NAVATHE 2002)) und jedes Attribut eine Eigenschaft der Entität. Zusammenhängend mit dieser Arbeit könnte der Niederschlag als Entität bezeichnet werden und die Koordinate, an der er auftritt, bzw. seine Höhe, als Attribut.
In PostgreSQL existieren physische und virtuelle Tabellen. Physische Tabellen werden als Relation und virtuelle Tabellen als View bezeichnet. Ein View stellt nur eine Sicht auf die Daten in den physischen oder anderen virtuellen Tabellen dar. Demzufolge unterliegt er im Gegensatz zu physischen Tabellen gewissen Beschränkungen. Es ist beispielsweise nicht möglich, einem View einen Primärschlüssel oder einen Index hinzuzufügen.
Über einen Primärschlüssel lässt sich jeder Datensatz in einer Tabelle eindeutig identifizieren. Bei Erstellen eines Primärschlüssels für eine Tabelle wird automatisch ein Index auf den Primär-

schlüssel erstellt (MOMJIAN 2001). Ein Index lässt sich als Tabelle beschreiben, die ein bestimmtes Attribut oder mehrere Attribute einer Tabelle enthält sowie Referenzen auf den Speicherort der Datensätze mit diesen Attributen. Wird auf eine Tabelle eine SELECT-Anfrage mit einer WHERE-Klausel ausgeführt, können die entsprechenden Datensätze bei Existenz eines Indexes schnell gefunden werden, wenn der Index auf das Attribut erzeugt wurde, das in der WHERE-Klausel verwendet wird (HARTWIG 2001). Ein Index wird jedes Mal aktualisiert, wenn die Datensätze der zugrunde liegenden Tabelle geändert werden.
Um Daten zu definieren, zu aktualisieren und anzufragen hat sich SQL (Structured Query Language) zu einem weit verbreiteten Standard in den relationalen Datenbanksystemen entwickelt (ELMASRI und NAVATHE 2002). Als deklarative Sprache ist SQL auf einen Anfrageoptimierer angewiesen, welcher den schnellsten Weg ermittelt, die Anfrage auszuführen.
PostgreSQL stellt einige Built-in Funktionen zur Verfügung wie z.B. sqrt(x) zur Berechnung der Quadratwurzel. Daneben existieren Aggregatfunktionen zur Berechnung von Durchschnittswerten (avg(x)) oder Summen (sum(x)) etc. Außerdem können eigene Funktionen erstellt werden, die unter anderem in den Sprachen SQL, PL/pgsql, PL/Tcl, PL/Perl und C geschrieben werden können (MOMJIAN 2001).
Für Datenbanken existieren verschiedene Grade an Normalisierung um redundante Datenhaltung zu vermeiden und dadurch die Konsistenz der Datenbank zu erhöhen. Ein Normalisierungsprozess war im Rahmen dieser Arbeit jedoch nicht notwendig, da die Datenbank sehr einfach gehalten ist und keine komplexen Zusammenhänge in den Daten bestehen.

Mit dem Datenbankmanagementsystem PostgreSQL sind ca. 868 Mio. Datensätze zu erfassen, die zu den benötigten Klimavariablen aufbereitet werden (Abbildung 1). Alle Datensätze müssen zunächst eingelesen werden. Da noch keine Punktgeometrien existieren, sind diese zu erzeugen und in einer separaten Tabelle zu speichern. Aus den Temperaturdaten wird dann für jedes Jahr die Vegetationszeit berechnet. In Abhängigkeit von der Vegetationszeit lassen sich Niederschlagssummen und Temperaturmittelwerte berechnen. Diese werden in einer Tabelle vereint, die als Grundlage für die Django-Applikation dient (vgl. Kapitel 3.3). Außerdem dient diese Tabelle als Ausgangspunkt für die weitere Aggregation zu Klimaperioden (je 30 Jahre). Bei diesem Schritt werden die Koordinaten hinzugefügt und es werden Views erzeugt, die die Daten für jeweils eine Übersichtskarte enthalten. Das erleichtert die Erstellung von GeoTIFFs. Die Trennung nach Baumarten wird in den Views aufgehoben.

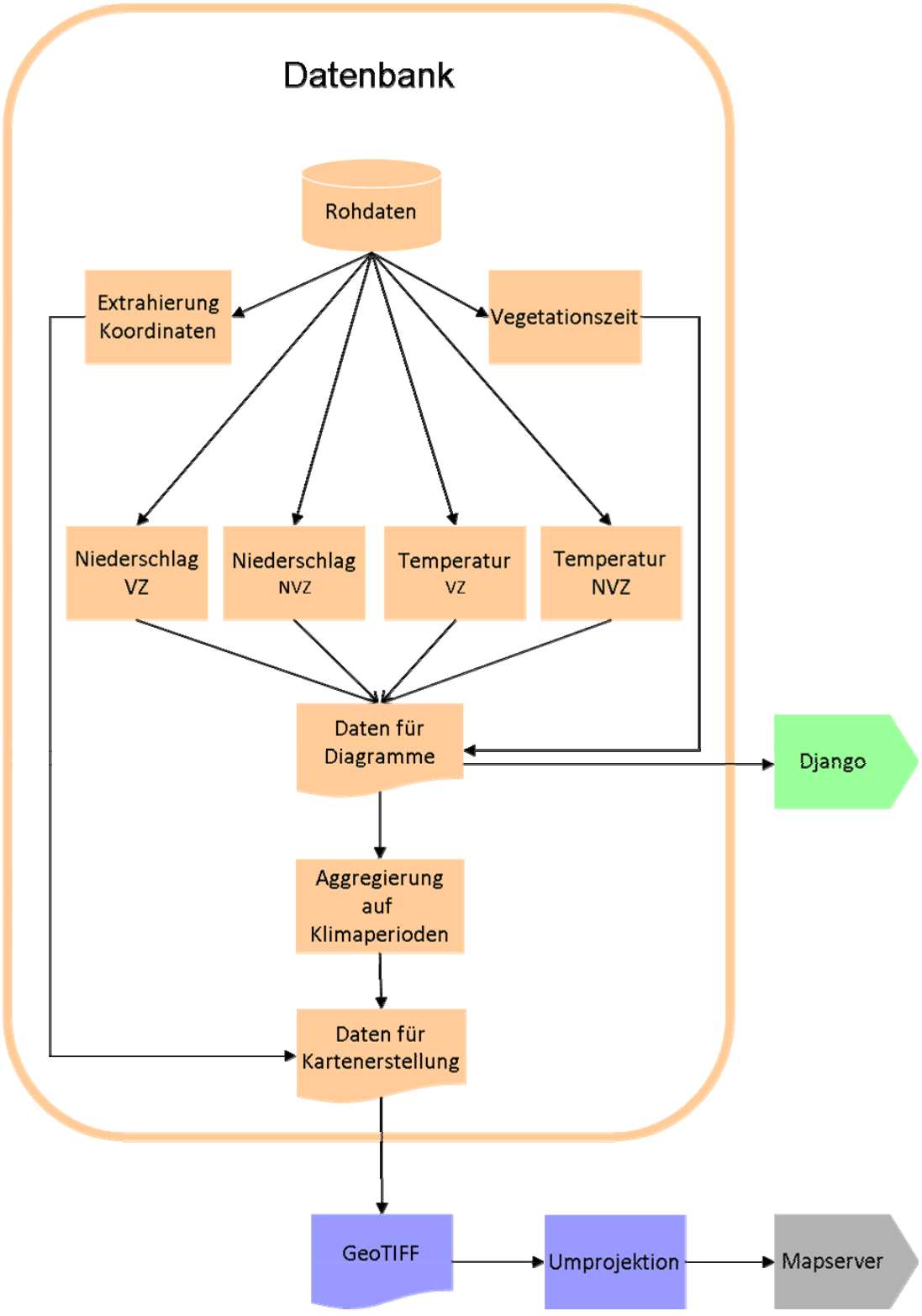


Abbildung 1: Ablauf der Datenaufbereitung in der Datenbank

3.2 Interaktive Karten

Die Bereitstellung von Karten im Internet wird auch als Web-Mapping bezeichnet. Dabei sind verschiedene Grade an Funktionalität möglich, ausgehend von einem statischen Bild bis hin zum Analysieren und Verändern der den Karten zugrunde liegenden Daten. Für Web-Mapping wird ein Zusammenspiel von verschiedenen Technologien auf Server- sowie Clientseite benötigt. Serverseitig müssen Webserver, Kartenserver und Geodaten zum Einsatz kommen (MITCHELL 2005). Alle auf Clientseite benötigte Technologie ist bereits in modernen Web-Browsern vorhanden und lässt sich durch JavaScript-Bibliotheken erweitern.

Wie Abbildung 1 und 2 erläutern, werden GeoTIFFs, die aus der Datenbank erzeugt wurden, in das Spherical Mercator Koordinatensystem projiziert, um eine Überlagerung eines Google-Maps-Layers mit den Übersichtskarten zu ermöglichen.

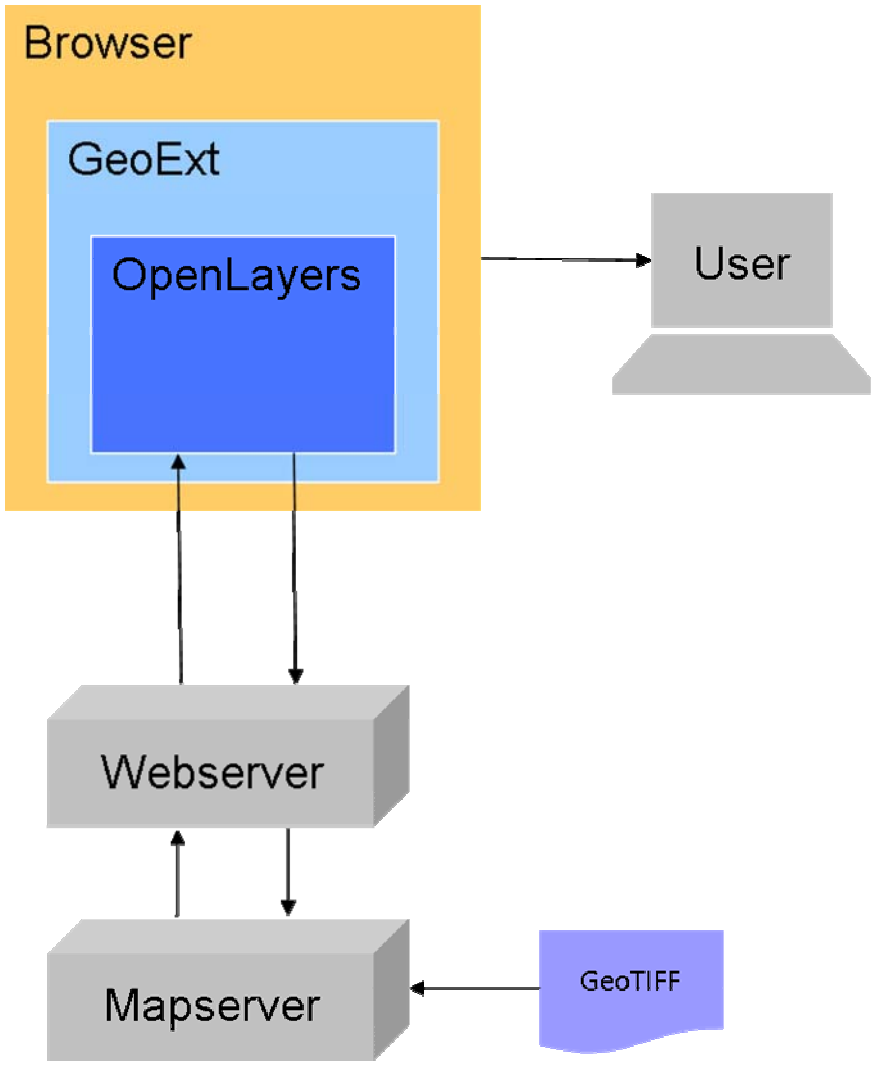


Abbildung 2: Die Web-Mapping Anwendung

Der UMN Mapserver wird als WMS-Dienst konfiguriert und erhält als Datenquelle die GeoTIFFs. Da OpenLayers wiederum WMS-Dienste als Datenquelle akzeptiert, werden durch OpenLayers Map-Requests zur Erzeugung der gewünschten Kartenausschnitte über einen Web-Server an den UMN Mapserver geschickt, welche dann in OpenLayers gemeinsam mit einem Google-Maps-Layer dargestellt werden. Die Einbindung der durch OpenLayers erzeugten Web-Mapping-Komponente in GeoExt ermöglicht zum einen den automatischen Abruf zum Kartenauschnitt zugehöriger Legenden und zum anderen einen komfortablen Auswahlbaum für alle anzeigbaren Layer.

Das GeoTIF-Format basiert auf dem TIFF. In den Metatags des GeoTIFFs können z.B. Angaben über die Koordinaten und das Referenzsystem gespeichert werden (RITTER und RUTH 2000). Für die Erzeugung und Umprojektion der GeoTIFFs werden die FW Tools genutzt. Der Begriff FW Tools beschreibt eine Sammlung von Open-Source-GIS-Software, die von Frank Warmerdam entwickelt wurde und für Windows zurzeit in der Version 2.4.7 vorliegt. Den Kern der FW Tools bildet das Programmpaket GDAL/OGR. Hierbei handelt es sich um Bibliotheken und einige kommandozeilenbasierte Dienstprogramme, mit deren Hilfe ca. 100 Raster- und ca. 50 Vektordatenformate gelesen und ausgeschrieben werden können. Während des Verarbeitens der Daten stehen unter anderem Möglichkeiten zur Umprojektion, Analyse, Visualisierung und Interpolation zur Verfügung (WARMERDAM 2010).
Zum Erstellen der GeoTIFFs wird auf das Programm gdal_grid zurückgegriffen, welches über die FW Tools Shell aufgerufen werden kann. Das Programm erzeugt ein Gitter (Raster). Die Anzahl der Zellen in x- und y-Richtung sowie die Ausdehnung des Gitters lassen sich vorgeben. Die Werte der Gitterknoten werden aus den Daten interpoliert.
Der Programmaufruf sieht wie folgt aus:

gdal_grid –ot *datentyp* –of *outputformat* –txe *xmin xmax* -tye *ymin ymax* –outsize *x y* –a *algorithmus* –a_srs *epsg* –zfield *attribut* –l *tabelle* "*datenbankverbindung*" *filename*

Zur Projektion der erzeugten GeoTIFFs in das Google Mercator System wird das Programm gdalwarp verwendet:

gdalwarp –a_srs *epsg* –t_srs *epsg inputfile outputfile*

Weitere Informationen zu den Eingabeparametern für gdal_grid und Gdalwarp finden sich bei WARMERDAM 2009a und 2009b.

Der UMN Mapserver ist ein plattformunabhängiger Kartenserver, der vor ca. 15 Jahren an der Minnesota Universität entwickelt wurde. Ein Kartenserver erzeugt aus Nutzeranfragen Karten aus Geodaten, welche an den Nutzer zurückgegeben werden. Der UMN Mapserver unterstützt zahlreiche OGC-Standards und verschiedenste Raster- und Vektordatenformate und noch einiges mehr (UNIVERSITY OF MINNESOTA 2010a). Die aktuelle Version des UMN Mapserver ist 5.6.3. Für Windows existiert ein Installationspaket des UMN Mapserver, welches dessen schnelles und unkompliziertes Einrichten ermöglicht. Mitgeliefert wird auch der Web-Server Apache.
Um den UMN Mapserver nutzen zu können, muss eine Konfigurationsdatei geschrieben werden: Das Mapfile. Es ist eine einfache Textdatei (MITCHELL 2005) und besteht aus verschiedenen Objekten mit dazugehörigen Parametern (UNIVERSITY OF MINNESOTA 2010b). Zu den wichtigsten Objekten gehört das Map-Objekt, welches einige allgemeine Angaben und alle weiteren Objekte enthält. Das gesamte Mapfile stellt das Map-Objekt dar. Außerdem existiert das Layer-Objekt. Es enthält alle zu verarbeitenden Geodaten. Für jede Datenquelle muss ein weiteres Layer-Objekt initialisiert werden. Das Legend-Objekt liefert die Legende für die Layer-Objekte. Innerhalb eines Layers kann das Class-Objekt genutzt werden, um die Daten zu klassifizieren. Das Legend-Objekt greift diese Klassifizierung auf und erstellt entsprechende Legendeneinträge. Die Legendensymbole lassen sich über das Style-Objekt formatieren, welches nur innerhalb eines Class-Objektes verwendet werden kann. Zur Gestaltung von Beschriftungen (z.B. in der Legende) steht das Label-Objekt zur Verfügung. Die Ausgabeprojektion der Daten wird über das Projection-Objekt geregelt. Zur Verwendung einer epsg-Projektion („init=epsg:xxxx“) muss diese im epsg-file (\ms4w\proj\nad\epsg) definiert sein. Zur Nutzung der Google-Mercator Projektion wurde das epsg-file um folgendes ergänzt:

```
<900913> +proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +no_defs <>
```
(BUTLER et al. 2010)

Das letzte wichtige Objekt ist das Web-Objekt. Es spielt vor allem dann eine Rolle, wenn der UMN Mapserver als WMS genutzt werden soll. Dieses Objekt enthält die Metadaten „wms_srs, wms_title und wms_onlineresource“ die eine Rolle für den Aufbau eines XML-Dokumentes im Rahmen einer GetCapabilities-Anfrage spielen. Außerdem sollten im Map-Objekt und auf Layer-Ebene der Parameter NAME und das Projection-Objekt enthalten sein, die ebenfalls für die GetCapabilities-Anfrage, bzw. für GetMap und GetFeatureInfo-Requests, von Bedeutung sind.
Weitere Informationen zu den Objekten und ihren Parametern sind bei UNIVERSITY OF MINNESOTA 2010c einzusehen. Dort ist auch die Konfiguration als WMS explizit beschrieben.

OpenLayers (Version 2.9) ist ein Web-Mapping Client. Dahinter versteckt sich eine Javascript-Bibliothek, mit deren Hilfe Karten in Web-Browsern dargestellt werden können (OSGEO 2010a). Zahlreiche Datenformate können in OpenLayers eingebunden werden (GoogleMaps, WMS, WFS, Mapserver, OpenStreetMap, Vektorlayer, etc). OpenLayers empfiehlt sich, um auf schnelle, unkomplizierte Weise Karten in Web-Anwendungen einzubinden. Deshalb wird OpenLayers als Web-Mapping Client genutzt.
OpenLayers bietet 2 Klassen an, die für die Einbindung einer Karte Grundvoraussetzung sind: Map und Layer. Map gibt die Rahmenbedingungen für die interaktive Karte vor (Koordinatensystem, Ausdehnung, etc.) und ermöglicht die Einbindung von Steuerelementen (z.B. Bewegen der Karte) und Layer ist die Datenquelle, aus der die Karte gebildet werden soll. (OSGEO 2010c).
OpenLayers bietet zwar ein hohes Maß an Funktionalität; die Gestaltung einer ansprechenden Benutzeroberfläche lässt sich jedoch durch GeoExt besser realisieren. GeoExt (Version 0.7) ist eine JavaScript Bibliothek, die auf Ext JS aufbaut. Ext JS ist ebenfalls eine JavaScript-Bibliothek, welche für die Erstellung mächtiger Internetanwendungen genutzt wird (SENCHA INC. 2010a).
Die mit GeoExt gestaltete Benutzer-Oberfläche basiert auf der Nutzung der Klassen Ext.Panel, Ext.TreePanel und Ext.Tree.TreeNode aus der Ext JS – Bibliothek und deren Erweiterungen GeoExt.MapPanel, GeoExt.LegendPanel und GeoExt.tree.LayerNode aus der GeoExt – Bibliothek. Den GeoExt-Erweiterungen stehen alle Optionen, Eigenschaften, Events und Methoden der jeweiligen Ext JS – Klassen zur Verfügung.
Ein Panel ist ein Container, dem verschiedene Inhalte und Layouts zugewiesen werden können (SENCHA INC 2010b). So kann ein TreePanel einen Auswahlbaum, ein MapPanel ein Kartenob-

jekt (aus OpenLayers) und ein LegendPanel eine Legende enthalten (GEOEXT COMMUNITY 2010).

3.3 Dynamische Grafiken

Stellt der Nutzer eine Koordinatenanfrage an das System, müssen entsprechende Liniendiagramme erstellt und zurückgeliefert werden (Abbildung 3). Zur Verarbeitung der Anfrage wird das Web-Development-Framework Django verwendet. Die URL, von dem die Anfrage stammt, wird an die Datei URLconf (ein Python-File) übergeben. Darin ist definiert, welche URL welchen View aufruft. Ein View ist ein Python-file, welches einen Request entgegen nimmt und den Inhalt einer html-Seite zurückgibt. Die html-Seite, in die der erzeugte Inhalt eingefügt werden soll, liegt als Template bereit. Für die Erstellung von Liniendiagrammen ist es notwendig, dass der View Zugang zur Datenbank erhält und die Klimavariablen abfragen kann. Dazu nutzt er das Model (ebenfalls ein Python-file). Prinzipiell regelt das Model alles, was mit der Datenbank zusammenhängt: Speichern, Abfragen und Ändern von Daten. Die aus der Datenbank erhaltenen Klimadaten werden an eine Javascript-Funktion übergeben, deren Aufruf im Template bereitgestellt wird. Ist das Template gerendert und an den Nutzer übersandt, wird die Funktion ausgeführt und erstellt ein Liniendiagramm.

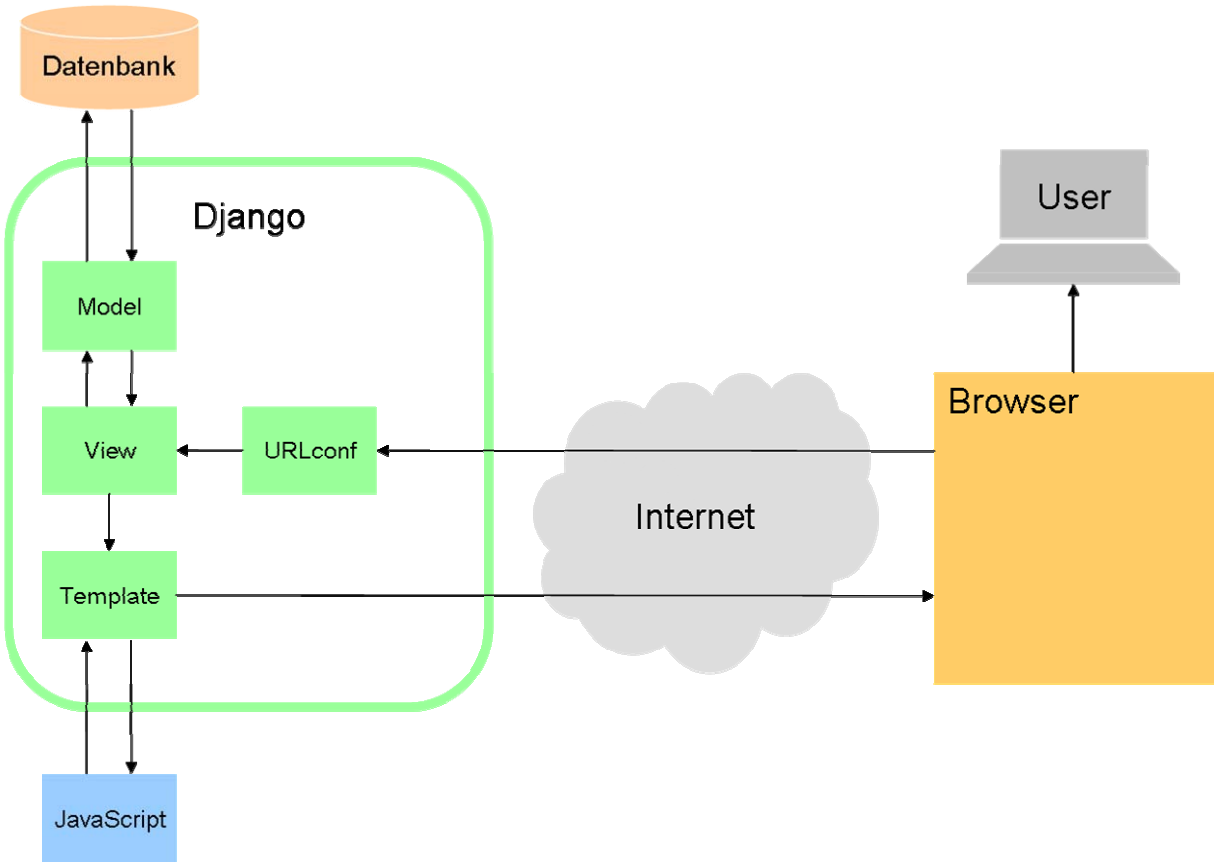


Abbildung 3: Ablauf bei der Erzeugung dynamischer Liniendiagramme

Mit Django (aktuelle Version 1.2) lassen sich in kurzer Zeit komplexe Webseiten erstellen. Es handelt sich dabei um eine Sammlung von Python-Bibliotheken, die wiederum durch die Skriptsprache Python genutzt werden. Entwickelt wurde es im Jahre 2003 durch Mitarbeiter von Lawrence Journal-World. Im Sommer 2005 wurde es als Open Source Software freigegeben (HOLOVATY und KAPLAN-MOSS 2009). Durch das add-on GeoDjango lassen sich auch Web-Anwendungen mit geografischem Bezug entwickeln (DJANGO SOFTWARE FOUNDATION 2010). Der Einstieg in Django fällt bei Vorkenntnissen in Python leicht.

Django bietet Abkürzungen für wiederholt anfallende Programmieraufgaben und unterstützt in Anlehnung an das MVC-Pattern (Model-View-Controller) das „loose coupling“. Dabei wird jeder Programmkomponente eine bestimmte Aufgabe zugeteilt und sie kann ausgetauscht werden, ohne die anderen Komponenten zu beeinflussen. (HOLOVATY und KAPLAN-MOSS 2009). Mit Django lässt sich also das Design einer Webseite von ihrem Inhalt trennen. Diese Trennung ist in Django über das MTV-Pattern (Model-Template-View) implementiert.

Die Google Chart Tools stellen vielfältige Möglichkeiten zur Verfügung, Diagramme aller Art in die Webseite einzubinden. Dabei werden interaktive Grafiken durch die Google Visualization API (Java-Script Bibliothek) erzeugt. Statische Grafiken lassen sich auf einem von Google bereitgestellten Chartserver über GET- oder POST-Requests oder unter zusätzlicher Ausnutzung der Google Visualization API erstellen (GOOGLE 2010).

4 Datenhaltung

Im Bereich der Datenhaltung lassen sich 4 Arbeitsgebiete abgrenzen:
a) Die Vorbereitung der Datenbank. Im Laufe der weiteren Datenverarbeitung war es notwendig, auf bestimmte Informationen (z.B. Menzelparameter) zuzugreifen. Deshalb werden in diesem Gebiet alle im weiteren Verlauf benötigten Informationen in Tabellen hinterlegt. Außerdem zeigte sich, dass eine bestimmte Struktur der Rohdaten von Vorteil war. Dementsprechend wurden die Rohdaten zunächst umstrukturiert.
b) Die Berechnung der Vegetationsperiode. In diesem Gebiet werden Funktionen entworfen, mit denen die Vegetationszeit berechnet werden kann.
c) Die Aggregierung der Rohdaten. Die einzelnen Tageswerte müssen bis zur Ebene der Liniendiagramme oder der Übersichtskarten zusammengefasst werden.
d) Die Umstrukturierung der Ergebnisse. Der Zugriff von außerhalb auf die Ergebnisse soll so unkompliziert wie möglich sein. Deshalb werden die Ergebnisse entsprechend neu strukturiert.

Im Gesamtablauf sind drei Punkte zu beachten. Der komplette Prozess der Datenaufbereitung hat innerhalb weniger Tage abzulaufen. Deshalb sollen die im Aufbereitungsprozess notwendigen Datenanfragen mit minimalem Zeitaufwand durchgeführt werden. Außerdem wirkt sich jede in der Datenbank gespeicherte Information auf den Speicherplatz aus. Es soll versucht werden, den Speicherbedarf so niedrig wie möglich zu halten. Des Weiteren muss im Auge behalten werden, dass eventuell weitere Klimadaten leicht in den bestehenden Prozess miteingebunden werden können. Manchmal treten Widersprüche zwischen diesen drei Punkten auf, die sich nicht vermeiden lassen. So sind z.B. Indices notwendig, um die Laufzeit zu optimieren, doch gleichzeitig erhöhen sie auch den Speicherplatz. In solchen Fällen muss abgewogen werden, was wichtiger ist.

Ein Überblick über die in der Datenbank enthaltenen Rohdaten befindet sich in Anhang A. Die zur Aufbereitung entwickelten Skripte sind in Anhang B-E hinterlegt.

Die Skripte werden am Szenario A1B für die Aufbereitung von 2x130 Jahren entwickelt, welche sich mit wenigen Änderungen auf das Szenario B1 übertragen lassen. Vor beide Szenarien werden jeweils die Simulationen des 20. Jahrhunderts gesetzt, so dass stets ein Zeitraum von 130

Jahren existiert. Die meisten der dieser Arbeit beigefügten SQL-Skripte beziehen sich nur auf die beiden mit CLM ausgeführten Modelläufe des Szenarios A1B.

4.1 Vorbereitung

In den Rohdaten sind Längen- und Breitengrade (gespeichert als Integer, um Speicherplatz zu sparen) enthalten, die jedoch noch keine Geometrie darstellen und keinem Koordinatensystem zugeordnet sind. Aus einer beliebigen Tabelle der Rohdaten werden Längen- und Breitengrade extrahiert und zu Punktgeometrien transformiert. Sie werden dem geografischen Referenzsystem „World Geodetic System“ (WGS 84) zugeordnet.
Die Temperatur-Rohdaten werden in eine Form gebracht (Anhang B), die sich für die Berechnung der Vegetationsperiode gut nutzen lässt. Hierfür werden zwei Tabellen erschaffen (temperature1 und temperature2), die Tagesmittelwerte von 1969-2100 erhalten:

```
CREATE TABLE temperature1 AS
        SELECT "T_2M_AV", time, lon, lat FROM a1b_1_temp
        UNION
        SELECT "T_2M_AV", time, lon, lat FROM c20_2_temp WHERE time >= '1969-01-01';
```

Die Notwendigkeit von temperature1 ergibt sich aus 2 Gründen. Erstens ist es nicht möglich, einen Tabellenname als Parameter an eine Funktion zu übergeben. D. h. alle Funktionen, die bisher auf temperature1 zugreifen, müssten dupliziert werden und je einmal auf a1b_1_temp und zum anderen auf c20_2_temp zugreifen. Nun ist das an sich noch kein ausreichender Grund, um den immens höheren Speicherplatzverbrauch, der durch temperature1 entsteht, zu rechtfertigen. Jedoch muss aufgrund der duplizierten Funktionen geregelt werden, wann welche der Funktionen verwendet werden sollen. D. h. es stellt sich die Frage, wann eine Funktion, die auf a1b_1_temp und wann eine Funktion, die auf c20_2_temp zugreift, verwendet wird. Diese Unterscheidung muss innerhalb der Funktion veggi() (vgl. Kapitel4.2) getroffen werden. Das wiederum heißt, dass bei jedem Iterationsschritt derjenigen Schleife, die über die Jahre iteriert (vgl Kapitel 4.2), mit einer IF-Bedingung geprüft werden muss, ob das Jahr <= 2000 oder >2000 ist. Im Rahmen der Entwicklung der Funktion veggi() wurde die Erfahrung gemacht, dass IF-Bedingungen zeitaufwendig sind, weshalb sie soweit irgendwie möglich, umgangen werden. Auch wenn dabei von Millisekunden die Rede ist, wird diese IF-Bedingung in jedem Iterationsschritt, d. h. 130 mal, sowie für jede Koordinate (ca. 2500) und für 4 Szenariendurchläufe ausge-

führt. Daraus ergibt sich dass eine IF-Bedingung ca. 1.300.000 mal geprüft wird. Abhängig von der Prozessorleistung kann dies kein Problem darstellen oder den Aufwand ins Unermessliche treiben.

Außerdem muss bei Erreichen des Jahres 2000 im selben Iterationsschritt auf beide Tabellen (a1b_1_temp und c20_2_temp) zugegriffen werden. Dies erhöht wieder die Komplexität der Funktion veggi() und sorgt für gesteigerten Zeitaufwand.

Der Zugriff auf temperature1 gestaltet sich im Zuge der Vegetationsperiodenberechnung also einfacher als der Zugriff auf a1b_1_temp und c20_2_temp. Der Tabelle temperature1 werden weitere Attribute hinzugefügt,

```
ALTER TABLE temperature1
        ADD column jahr int4, ADD column monat int4, ADD column tage int4;

UPDATE temperature1
        SET jahr = EXTRACT(YEAR FROM time),
            monat = EXTRACT(MONTH FROM time),
            tage = EXTRACT(DAY FROM time);
```

die später Teil eines Index sein werden. Offensichtlich wird durch diese Vorgehensweise der benötigte Speicherplatz immens erhöht, doch steht zunächst die zügige Berechnung der Vegetationszeit im Vordergrund.

Für die Niederschlagsdaten werden Views erzeugt (precipitation1 und precipitation2), deren Datensätze denselben Zeitraum umfassen wie die Tabellen temperature1 und temperature2. Dadurch lassen sich SQL-Anweisungen, die für die Aggregierung der Daten zu den Klimavariablen anhand der Temperaturdaten entwickelt wurden, leichter auf die Aggregierung der den Niederschlag betreffenden Klimavariablen übertragen (Anhang E).

Außerdem wird eine Tabelle mit den Menzelparametern angelegt, die zur Berechnung der Vegetationszeit benötigt werden. Für Fichte und Douglasie werden dieselben Parameter genutzt.

Ein Multipolygon, welches Deutschland absteckt, ermöglicht später das Entfernen von Daten, die nicht innerhalb von Deutschland liegen.

4.2 Berechnen der Vegetationsperiode

Die Vegetationsperiode wird über die Funktion veggi() berechnet (Anhang C). Diese Funktion liefert selbst keinen Rückgabewert, sondern dient dazu, die Tabellen veg_per1 und veg_per2

mit den Daten über die Vegetationsperiode aufzufüllen. Innerhalb von veggi() wird auf eine Reihe von SQL-Funktionen zurückgegriffen. Die SQL-Funktionen und veggi() existieren für A1B in 2-facher Ausführung; je einmal für die Berechnung der Vegetationsperiode in Lauf 1 und Lauf 2. Diese Notwendigkeit ist wieder darauf zurückzuführen, dass Tabellen nicht als Parameter an eine Funktion übergeben werden können. Deshalb müssen in allen Funktionen fixe Namen für Tabellen stehen.

Das Grundgerüst der Funktion veggi() beinhaltet vier ineinander geschachtelte Schleifen über die lon/lat-Werte, die Baumart und das Jahr für das jeweils die Vegetationsperiode berechnet werden soll, sowie ein INSERT- und ein UPDATE-Statement für die Tabelle veg_per:

```
CREATE OR REPLACE FUNCTION veggi() RETURNS void AS $$

    BEGIN
    FOR lons IN lowest_lon()..highest_lon() BY 2 LOOP

        BEGIN
        FOR lats IN lowest_lat()..highest_lat() BY 2 LOOP

            BEGIN
            FOR menzel_id IN lowest_menzel()..highest_menzel() LOOP

                BEGIN
                FOR year IN lowest_year()..(highest_year()-1) LOOP
                …
                …
                …
                UPDATE veg_per1
                SET next_begins = date_veg_begin where years = year AND lon = lons AND lat =
                lats AND species= menzel_id;
                …
                …
                …
                INSERT INTO veg_per1(years, this_begins, this_ends, length, species, lon, lat)
                VALUES(year+1, date_veg_begin, date_veg_ende, length_veg_per, menzel_id,
                lons, lats);

                END LOOP;
                END;
            END LOOP;
            END;
        END LOOP;
        END;
  END LOOP;
  END;

$$ LANGUAGE 'plpgsql';
```

Die Nutzung eines INSERTS- und eines UPDATE-Statements ermöglicht es, Informationen aus 2 verschiedenen Jahren in einer Zeile zu vereinen. Dies ist nötig, da für die Berechnung der Klimavariablen außerhalb der Vegetationszeit das Ende der Vegetationszeit (this_ends) im aktuellen Jahr und der Beginn der Vegetationszeit des folgenden Jahres (next_begins) bekannt sein müssen (Anhang E).

Die Berechnung der Vegetationsperiode beginnt am 01.11. (vgl. Kapitel 1.2.1) des aktuellen Jahres und endet spätestens am 05.10. des Folgejahres. Daraus ergibt sich, dass in veggi() manchmal „year+1“ zu sehen ist, und in den SQL-Funktionen „jahr=$x+1“.

Die Summierung aller Kältetage im November, Dezember und Januar über eine einfache Funktion

```
colddays := count_colddays(lons, lats, year, menzel_temp_colddays);
```

führte dazu, dass die Zählvariable „tag“, die dazu gedacht war, die Nummer des Tages im Jahr (DOY) anzugeben, erst am 01. Februar mit 0 initialisiert wurde. Wenn also der DOY benötigt wurde, musste „tag“ entweder um 31 oder 32 erhöht werden.

4.3 Aggregierung zu Klimavariablen

Mit Vorliegen der Informationen über die Vegetationszeit lassen sich die vier weiteren Klimavariablen für jedes Jahr berechnen (Anhang D). Die Temperaturmittelwerte und Niederschlagssummen außerhalb der Vegetationszeit basieren zwar auf 2 aufeinander folgenden Jahren, müssen aber einem Jahr zugeordnet werden. Da ein Zusammenhang zwischen den Ereignissen außerhalb der Vegetationsperiode und der darauf folgenden Vegetationsperiode besteht (der Niederschlag außerhalb der Vegetationsperiode beeinflusst z. B. die Wasserverfügbarkeit in der darauf folgenden Vegetationsperiode), werden die Klimavariablen außerhalb der Vegetationsperiode dem Jahr der kommenden Vegetationsperiode zugeordnet.

Das Ergebnis wird zunächst für jede Variable und jeden Lauf in einer separaten Tabelle gespeichert und dann in eine gemeinsame Tabelle (diagramme_a1b) überführt. Diese beinhaltet nun alle Klimavariablen, die für die Visualisierung durch die Liniendiagramme benötigt werden.

Die Jahreswerte lassen sich weiter zu Mittelwerten in den Klimaperioden (klimaperioden_a1b) zusammenfassen und bekommen die Punktgeometrien zugewiesen. In jeder Klimavariablen werden alle Werte, die außerhalb von Deutschland liegen, auf 0 gesetzt. Dazu wird die geomet-

rische Funktion ST_DISJOINT verwendet, welche „TRUE“ liefert, wenn sich die Geometrien räumlich nicht überschneiden.

```
UPDATE klimaperioden_a1b
  SET prec_vp_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
```

Wichtig ist, dass die Zahl 0 verwendet wird und nicht “NULL”, welches für einen nicht existierenden Wert steht. Denn wenn die GeoTIFFs erstellt werden, lässt sich mit dem Interpolationsalgorithmus „Nearest Neighbour“ der Gitterzelle 0 zugewiesen.

Alle nicht mehr benötigten Tabellen zum Speichern von Zwischenergebnissen können gelöscht werden, um Speicherplatz freizugeben.

4.4 Umstrukturierung der Ergebnisse

Die Klimavariablen auf Ebene der Übersichtskarten liegen alle in einer Tabelle und nach Baumarten, Modellläufen und Klimaperioden getrennt vor. Während der Erstellung der GeoTIFFs können SQL-Anfragen ausgeführt werden, um die Trennung nach Baumarten und Läufen aufzuheben und die GeoTIFFs für die verschiedenen Klimaperioden zu erzeugen. Diese Anfragen lassen sich bereits in der Datenbank in Form von Views hinterlegen (Anhang E).

Der Zugriff mit Django auf die Daten für die Liniendiagramme lässt sich leichter durchführen, wenn die Daten der beiden Szenarien A1B und B1 in einer Tabelle vorliegen. Nachdem die SQL-Skripte für B1 angepasst und ausgeführt sind, wird aus den Tabellen diagramme_a1b und diagramme_b1 eine Tabelle (diagramme) erzeugt, die die gesamte Information aus beiden Szenarien enthält (Anhang E).

5 Interaktive Karten

Im Rahmen der Entwicklung der Web-Mapping-Anwendung sind drei Probleme zu lösen. Zunächst einmal muss der Mapserver als Datenquelle in OpenLayers eingebunden werden. Die Benutzeranfragen, die daraufhin an den Mapserver geschickt werden, müssen schnellstmöglich verarbeitet werden. Denn grundsätzlich gilt, je länger die Ladezeiten des Inhaltes einer Web-Seite sind, desto ungehaltener wird der Nutzer. Dies führt im schlimmsten Fall dazu, dass er die Seite nicht mehr besucht, oder in diesem Fall die Web-Mapping-Anwendung nicht mehr nutzt. Drittens muss die grafische Benutzeroberfläche der Anwendung benutzerfreundlich sein. D. h. sie muss leicht zu bedienen sein, über ein ansprechendes Äußeres verfügen und Inhalte klar erkennen lassen.

5.1 WMS-Dienst

OpenLayers bietet das WMS-Layer an, welches die Darstellung von Daten aus WMS-Diensten ermöglicht. Soll der UMN Mapserver als WMS-Dienst betrieben werden, müssen einige zusätzliche Angaben zu den Namen, Projektionen und Metadaten der bereitgestellten Layer gemacht werden. Im folgenden ist ein Minimalbeispiel einer Konfigurationsdatei des UMN Mapserver abgebildet.

```
MAP
        NAME "klima"
        PROJECTION
                "init=epsg:900913"
        END


        METADATA
                'wms_title' 'klima'
                'wms_onlineresource' 'http://localhost/cgi-bin/mapserv.exe?map=C:\Dokumente
                                      und Einstellungen\sabine\Desktop\testMap\testMap.map'
                'wms_srs' 'EPSG:900913'
        END

        LAYER
                NAME "temp_vp_1_a1b"
                STATUS ON
                METADATA
                        'wms_title' 'Tagesmitteltemperatur in der Vegetationszeit (Klimaperiode 1)'
                        'wms_srs' 'EPSG:900913'
                END
                PROJECTION
                        "init=epsg:900913"
                END

        END
END
```

5.2 Benutzeranfragen

Die Geschwindigkeit, mit der Nutzeranfragen beantwortet werden, hängt unter anderem davon ab, wie schnell auf Serverseite das angeforderte Ergebnis erzeugt werden kann. In der Datenbank liegen Punktgeometrien im WGS 84 vor. Darzustellen sind jedoch Rasterzellen in der Google-Mercator-Projektion. Die dafür notwendige Verarbeitung der Daten im Rahmen einer Nutzeranfrage jedes Mal aufs Neue durchzuführen, hieße dem Nutzer eine unzumutbare Wartezeit aufzubürden. Deshalb werden die Daten weiter vorprozessiert. Es werden GeoTIFFs erzeugt, die die gewünschten Rasterzellen in der Google-Mercator-Projektion besitzen. Dies geschieht in zwei Schritten. Zunächst werden die GeoTIFFs mit dem Befehl gdal_grid erstellt:

gdal_grid **–ot** *Float32* **–of** *GTiff* **–txe** *4.9 15.9* **-tye** *46.9 56.1* **–outsize** *55 46* **–a** *nearest* **–a_srs** *EPSG:4326* **–zfield** *avg* **–l** *a1b_temp_vp_per1* **"PG:dbname**=’climate’ **host**=’localhost’ **po**rt=’5432’ **user**=’postgres’ **password**=’klima’**"** temp_vp_per1.tiff

Die Punktgeometrien in der Datenbank besitzen bereits die Form eines regelmäßigen Gitters und sollen die Mittelpunkte der Rasterzellen darstellen. Deshalb muss die Anzahl der Rasterzellen in x- und y-Richtung derjenigen der Punktgeometrien entsprechen und die minimalen und maximalen Ausdehnungen in x- und y-Richtung müssen um je 0.1 erweitert werden. So wird gewährleistet, dass anstelle jeder Punktgeometrie eine Rasterzelle mit 0.2° entsteht. Der Interpolationsalgorithmus „Nearest Neighbour“ sorgt dann dafür, dass jede Rasterzelle den Wert der ihr entsprechenden Punktgeometrie erhält.

Zum Umgehen einer „on-the-fly-projection“ wird das GeoTIFF, welches noch im WGS 84 vorliegt, mit gdalwarp in die Google-Mercator-Projektion umprojiziert. Hierbei ändert sich das Raster sowie die Anzahl der Zeilen und Spalten.

Gdalwarp **–a_srs** *EPSG:4326* **–t_srs** *EPSG 900913* temp_vp_per1.tiff temp_vp_per1_google.tiff

5.3 Benutzerfreundliche Oberfläche

Die Benutzeroberfläche (Abbildung 4, Auszüge aus dem JavaScript Code in Anhang H) besteht aus drei Komponenten: Kartenfenster, Legende und Auswahlbaum für die Layer. Im Mittelpunkt steht das Kartenfenster. Der Auswahlbaum und die Legende werden jeweils links sowie rechts angeordnet, um beide Komponenten nahe an der Karte zu haben. Dies ermöglicht einen komfortablen Abgleich der Karte mit der Legende sowie der Karte mit dem Auswahlbaum.

Das Kartenfenster ist ein MapPanel-Objekt. Darüber werden Größe, Position und Inhalt des Displays geregelt. Den Inhalt stellt ein OpenLayers Map-Objekt dar.

Das Map-Objekt beinhaltet ein Google-Layer als Base Layer und zahlreiche WMS-Layer als Non Base Layer. Das Base Layer bestimmt das Koordinatensystem und die Zoomstufen. Damit OpenLayers mit einem Layer aus der Google API interagieren kann, muss die Option sphericalMercator des Google-Layers auf „true“ gesetzt werden.

```
googlayer = new OpenLayers.Layer.Google(
            "Google",
            {sphericalMercator:true}
            );
```

Wie das Map-Objekt aufgrund dessen konfiguriert werden sollte, ist bei OSGEO (2010b) vorgeschlagen.:

```
maxResolution: 156543.0339,
maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
                                 20037508.34, 20037508.34),
```

Dass als Projektion das EPSG:900913 verwendet werden muss, versteht sich von selbst.

Die Beschränkung der Ausdehnung des Map-Objektes und der Zoomstufen

```
restrictedExtent:new OpenLayers.Bounds    (545000, 5914600,
            1778900, 7578400)

googlayer.MIN_ZOOM_LEVEL = 6 ;
googlayer.MAX_ZOOM_LEVEL = 12 ;
```

führt dazu, dass der Kartenausschnitt auf Deutschland beschränkt ist und bis auf die Ebene von ca. einer Rasterzelle hineingezoomt werden kann. Neben dieser für den Nutzer komfortablen Voreinstellung ist auch das Verschieben der Karte sowie das Zoomen mit der Maus ermöglicht.

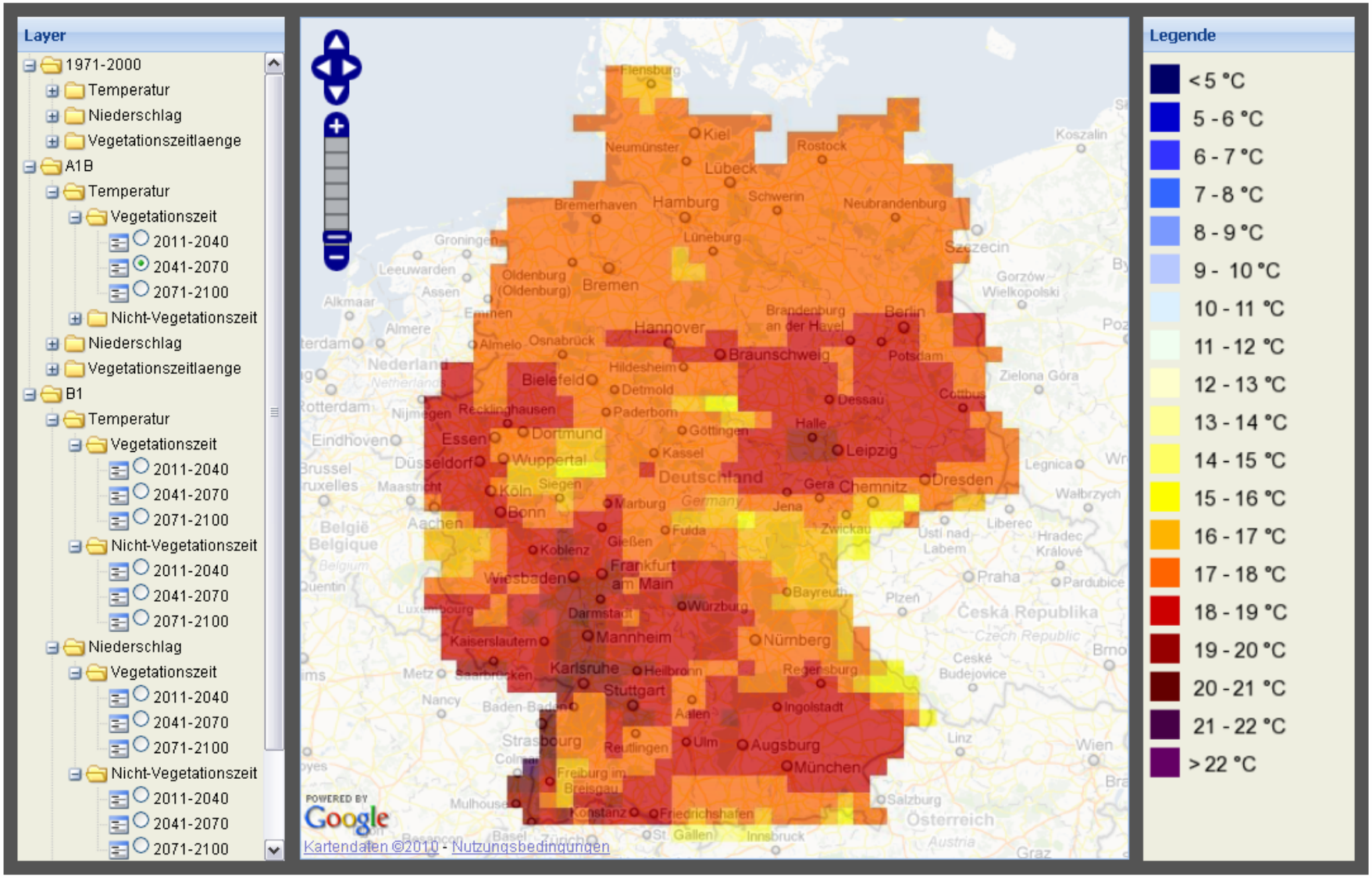


Abbildung 4: Oberfläche der Web-Mapping Anwendung

Wählt der Nutzer im Auswahlbaum ein Layer aus, bewirkt dies, dass das entsprechende Layer im Kartenfenster gezeigt, das bisher sichtbare Layer entfernt und für das nun aktuelle Layer eine Legende erzeugt wird. Diese drei Schritte können von GeoExt automatisch durchgeführt werden. Voraussetzung dafür ist, dass die Layerknoten im Auswahlbaum nicht die voreingestellten Checkboxes sondern Radiobuttons sind und dass ein LegendPanel existiert.
Ein Layerknoten lässt sich bei seiner Erzeugung als Radiobutton initialisieren, wenn die Option checkedGroup mit einem Gruppierungsnamen angegeben wird. Alle in der Anwendung enthaltenen Layerknoten verfügen über denselben Gruppierungsnamen, da niemals zwei Layer gleichzeitig gezeigt werden sollen. Jeder Layerknoten wird außerdem mit einem Layer verknüpft.

```
node_temp_vz_1_a1b = new GeoExt.tree.LayerNode({
                        layer:temp_vz_1_a1b,
                        text:'1971-2000',
                        checkedGroup:'checker'
                        });
```

Durch das Selektieren eines Layerknotens wird die Art des zugehörigen Layers bestimmt und abhängig davon entsprechend auf das Layer zugegriffen. Da hier alle Layer WMS-Layer sind, wird eine Karte als Bild vom UMN Mapserver abgefragt.
Das Layer des ausgewählten Knotens befindet sich im Layer Store. Existiert ein LegendPanel wird für alle Layer die im Layer Store enthalten sind, eine Legende erzeugt. Die Art der Erzeugung hängt ebenfalls wieder von der Art des Layers ab. In diesem Fall wird also mit einem GetLegendGraphic-Request vom UMN Mapserver eine Legende angefragt und das Legendenbild zusammen mit dem Layernamen in das LegendPanel eingefügt (Der Layername wurde entfernt, s. Anhang G).

Die Art eines Layers wird durch das OpenLayers Layer-Objekt vorgegeben. Dieses beinhaltet die in diesem Fall notwendigen Parameter zur Bildung der GetMap- und GetLegendGraphic-Requests: die URL für den Aufruf des Mapfiles und ein im Mapfile auftauchendes Layer.

```
temp_vz_1_a1b = new OpenLayers.Layer.WMS( "temp_vz_1_a1b",
                    "http://localhost/cgi-bin/mapserv.exe?map=C:/Dokumente
                    und Einstellungen/sabine/Desktop/visualization/sabine_klimaszenarien.map",
                    {layers: 'temp_vp_1_a1b'},
                    });
```

Ein weiteres Feature, das die Nutzung der Anwendung erleichtert, ist der Scroll-Balken im Auswahlbaum. Dieser wird automatisch erzeugt, wenn der Inhalt des Auswahlbaumes über den durch das TreePanel gebotenen Platz hinausreicht. Über die Eigenschaft

```
autoScroll:true
```

wird dem TreePanel diese Fähigkeit gegeben. Ein ständiges Öffnen und Schließen der Knoten kann dadurch umgangen werden.

Neben der einfachen Bedienbarkeit müssen die dargestellten Inhalte leicht zu erfassen sein. Die Positionierung des Auswahlbaumes, des Kartenfensters und der Legende wurden unter diesem Gesichtspunkt bereits angesprochen.
Im Kartenfenster sollen zum einen das Rasterlayer mit entsprechender Farbgebung für die Klimavariablen sichtbar sein und zum anderen zwecks räumlicher Orientierung das Google-Layer. Da das Rasterbild das Google-Layer überlagert, muss dessen Deckkraft verringert werden, um Städtenamen, Straßen etc. sichtbar zu machen. Im OpenLayers Layer-Objekt wird dafür die Eigenschaft

```
opacity:0.7
```

genutzt. Bei 70% Deckkraft sind die Farben des Rasterbildes noch gut zu erkennen, aber gleichzeitig bietet die Hintergrundkarte eine ausreichende Orientierung.
Die Farbgebung der einzelnen Rasterzellen muss so gewählt werden, dass sie sich sichtlich voneinander abgrenzen und zwischen den verschiedenen Klimaperioden Trends zu erkennen sind. Dazu werden zunächst alle in den GeoTIFFs vorhandenen Werte klassifiziert. Für die Temperatur wird eine Klassenbreite von 1°C, für den Niederschlag von 100mm und für die Länge der Vegetationszeit von 10 Tagen gewählt. Die Klassifizierung und Farbgebung der Klassen findet in den Layer-Objekten im Mapfile statt (Anhang F); eine grafische Darstellung der daraus resultierenden Legenden liefert Abbildung 5. Bei der Klassifizierung wurde allen Zellen mit dem Wert 0 die Farbe weiß (255 255 255) zugewiesen. Diese konnte dann auf transparent gesetzt werden, was dazu führt, dass nur die Zellen innerhalb Deutschlands sichtbar sind.
Die Temperaturen außerhalb der Vegetationsperiode liegen bei einer Betrachtung über alle Klimaperioden immer unterhalb denen der Vegetationsperiode. Es wäre demnach möglich ge-

wesen, für Sommer- und Wintertemperaturen dieselbe Anzahl an Klassen zu wählen und diesen Klassen dieselbe Farbgebung zuzuweisen. Stattdessen wurde jedoch entschieden, Wintertemperaturen in Blautönen und Sommertemperaturen in Rottönen darzustellen und beide in der Legende sichtbar zu machen. Das soll eine bessere Vergleichsmöglichkeit des Anstieges der Wintertemperaturen, der bis zur 4. Klimaperiode zu erwarten ist, mit den in der 1. Klimaperiode vorherrschenden Sommertemperaturen ermöglichen.

Über ganz Deutschland verteilt können innerhalb sowie außerhalb der Vegetationsperiode dieselben Niederschläge anfallen. Deshalb wurde eine allgemeine Klassifizierung des Niederschlags vorgenommen, welche auf beide Fälle angewendet wird. Da für die Länge der Vegetationszeit dieselbe Anzahl an Klassen anfiel wie bei der Klassifizierung des Niederschlags, wurde die Farbgebung beibehalten.

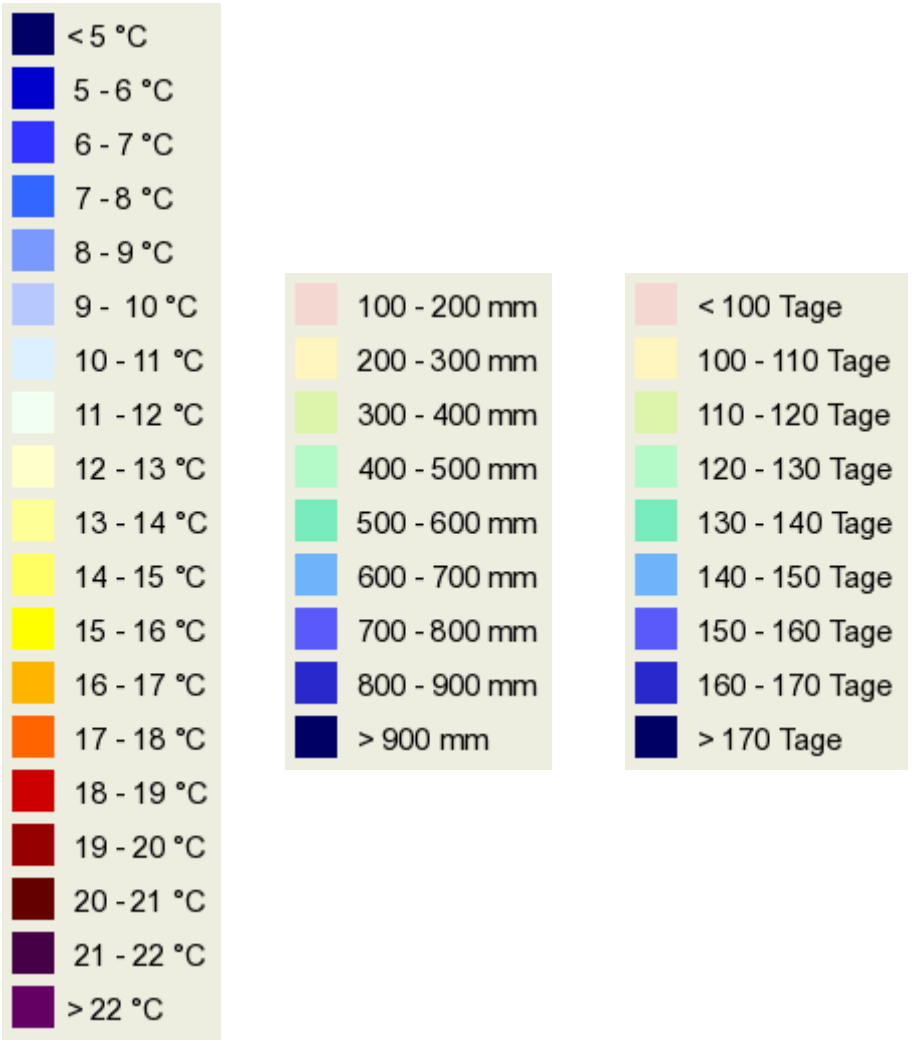


Abbildung 5: Klassifizierung und Farbgebung der Klimavariablen.

6 Dynamische Grafiken

Zur Anforderung der Liniendiagramme soll ein Formular bereitgestellt werden, in das der Nutzer von Hand Koordinaten eingeben kann oder die gewünschten Koordinaten ganz bequem mit der Maus aus einer Karte auswählen kann. Weiterhin soll er die Möglichkeit haben, für die entsprechenden Koordinaten Diagramme zu erhalten aus denen sich für die Klimavariablen entweder Unterschiede zwischen den Baumarten oder zwischen den Szenarien entdecken lassen. Zum Zeichnen dieser Diagramme soll eine Javascript-Funktion genutzt werden. Außerdem muss die Möglichkeit bestehen, die Linien zu glätten, um Unterschiede und Trends besser wahrzunehmen.

6.1 Formular

Auf der Benutzeroberfläche (Abbildung 6, Code in Anhang I) existiert je ein Eingabefeld für den Längengrad und für den Breitengrad. Zur Auswahl der Koordinaten aus der Karte und der Übertragung in die Eingabefelder wurde eine JavaScript-Funktion geschrieben: Für das OpenLayers Map-Objekt, in welchem sich die Google-Karte befindet, wird ein Klick-Event registriert. Die Pixel-Koordinaten, an denen das Klick-Event auftritt, werden in die Google Mercator Projektion übersetzt und von der Google Mercator Projektion ins WGS 84 transformiert.

```
map.events.register('click', map, function(e) {

  point = map.getLonLatFromViewPortPx(e.xy);
  point.transform(new OpenLayers.Projection("EPSG:900913"),
                  new OpenLayers.Projection("EPSG:4326"));
  x=point.lon;
  y=point.lat;
  document.coordform.lon.value = x;
  document.coordform.lat.value = y;

        });
```

Es existiert außerdem ein Eingabefeld für die Fenstergröße des gleitenden Mittels, welches standardmäßig mit dem Wert 5 initialisiert wird.

Es stehen zwei Buttons zum Absenden des Formulars bereit, mit denen sich entweder die Liniendiagramme zum Vergleich von Szenarien (Szenarienvergleich) oder Baumarten (Baumartenvergleich) anfordern lassen. Beim Absenden der Buttons wird ein View aufgerufen, der den Code zum Erzeugen der Daten für beide Arten von Liniendiagrammen enthält (vgl. Kapitel 6.3).

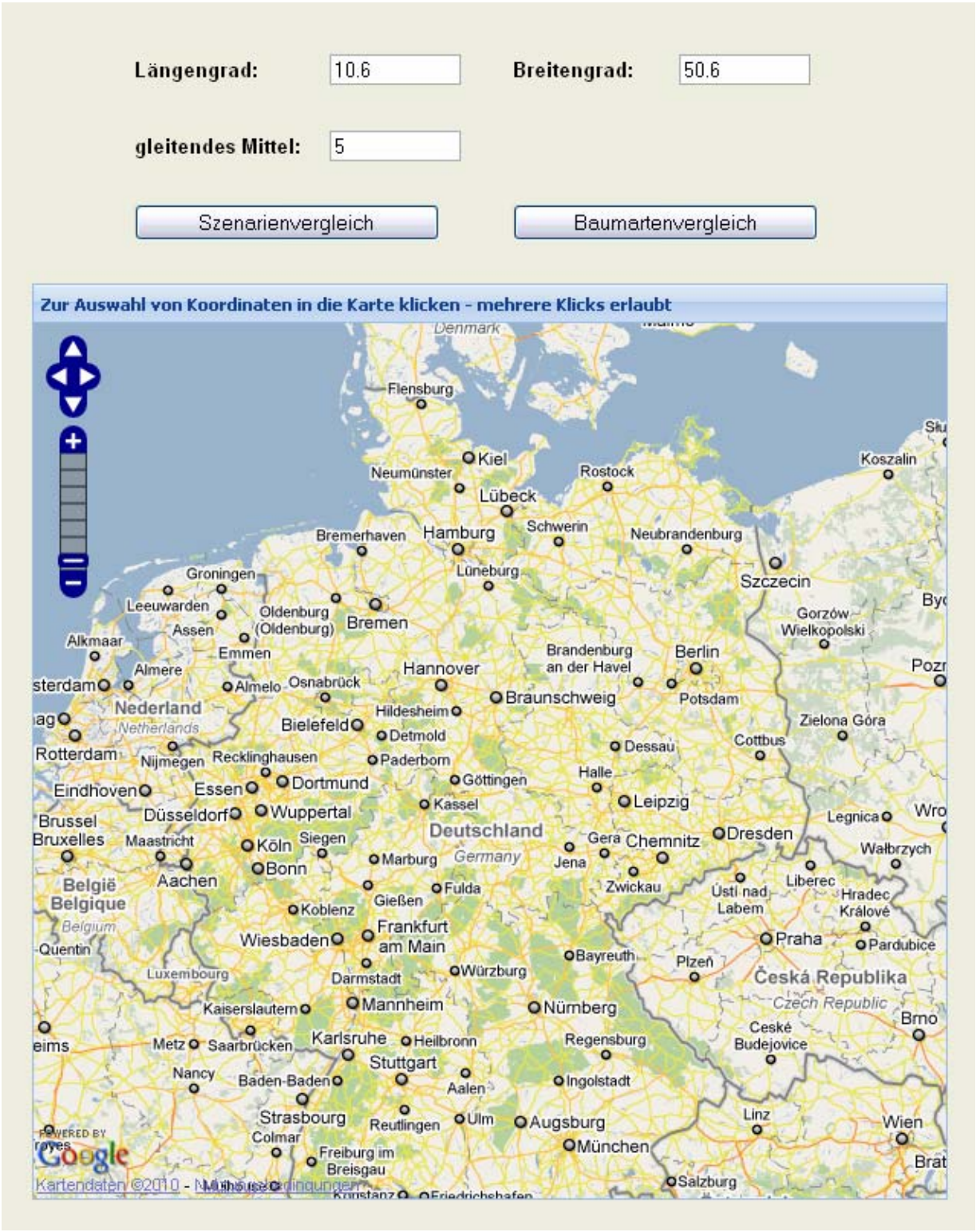


Abbildung 6: Benutzeroberfläche für die Anforderung der Liniendiagramme

6.2 JavaScript-Funktion

Nach einer erfolgreichen Nutzeranfrage (vgl. Kapitel 6.3) wird das Template gerendert und die Javascript-Funktion drawChart2(), welche auf der Google Visualization API basiert, kann ausgeführt werden. Sie erzeugt ein Liniendiagramm. Um verschiedene Liniendiagramme zu erhalten, kann drawChart2() beliebig oft aufgerufen und mit jeweils anderen Daten parametrisiert werden. In der Funktion drawChart2() wird eine Tabelle mit 5 Spalten erzeugt, in die die Klimadaten eingefügt werden müssen. Mit der Methode draw() wird ein auf dieser Tabelle basierendes Diagramm erstellt.

```
function drawChart2(werte, miny, maxy, div, title) {

        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Year');
        data.addColumn('number', '2');
        data.addColumn('number', '3');
        data.addColumn('number', '4');
        data.addColumn('number', '5');
        data.addRows(werte);

        var chart = new google.visualization.ImageLineChart(document.getElementById(div));
        chart.draw(data, {width: 300, height: 240, min: miny, max:maxy, title:title,
                colors:['CC0000','FF9900','0033FF','3399FF'], legend:'none'});

}
```

Die erste Spalte beinhaltet die Werte, die auf der x-Achse dargestellt werden sollen. Alle zu diesem Wert gehörigen Daten werden in derselben Zeile, aber in weiteren Spalten, eingefügt. Jede Spalte entspricht einem Graphen. Da in jedem Diagramm jeweils 4 Graphen abgebildet werden sollen, existieren 4 weitere Spalten. Als Argumente werden neben den Klimavariablen auch Minimal- und Maximalwerte für den y-Achsenabschnitt übergeben, wodurch eine individuelle Skalierung der Achse möglich ist.
Diese Funktion lässt sich für den Vergleich von Klimavariablen zwischen den Baumarten sowie den Szenarien nutzen. Tabelle 2 gibt Aufschluss darüber, in welcher Spalte jeweils die Klimavariablen einer Baumart oder eines Szenarienlaufes stehen werden. An diese Reihenfolge wird sich gehalten, während die Nutzeranfragen verarbeitet und die Daten entsprechend formatiert werden (vgl. Kapitel 6.3)

Tabelle 2: Spalteninhalte der Funktion drawChart2()

Spalte	Szenario	Baumart
2	A1B Lauf 1	Eiche
3	A1B Lauf 2	Buche
4	B1 Lauf 1	Fichte/Douglasie
5	B1 Lauf 2	Kiefer

Um in die Funktion eingefügt und richtig dargestellt zu werden, müssen die Klimadaten in geschachtelten Listen vorliegen:

[['1971',*temperatur_eiche_a1b_1*, *temperatur_eiche_a1b_2*, *temperatur_eiche_b1_1*, *temperatur_eiche_b2_2*], ['1972', *temperatur_eiche_a1b_1*, *temperatur_eiche_a1b_2*, *temperatur_eiche_b1_1*, *temperatur_eiche_b2_2*], ['1973', …]]

oder

[['1971',*temperatur_eiche_a1b_1* ,*temperatur_buche_a1b_1*, *temperatur_fi/dou_a1b_1*, *temperatur_kiefer_a1b_1*], ['1972', *temperatur_eiche_a1b_1*, *temperatur_buche_a1b_1*, *temperatur_fi/dou_a1b_1*, *temperatur_kiefer_a1b_1*], ['1973', …]]

Alle Diagramme, die an einer bestimmten Koordinate mit dieser Funktion entstehen, sind in Abbildung 7 und Abbildung 8 sowie in Anhang J am Beispiel der Koordinate (10.6 , 50.6) und für ein gleitendes Mittel von 7 dargestellt.

Eiche

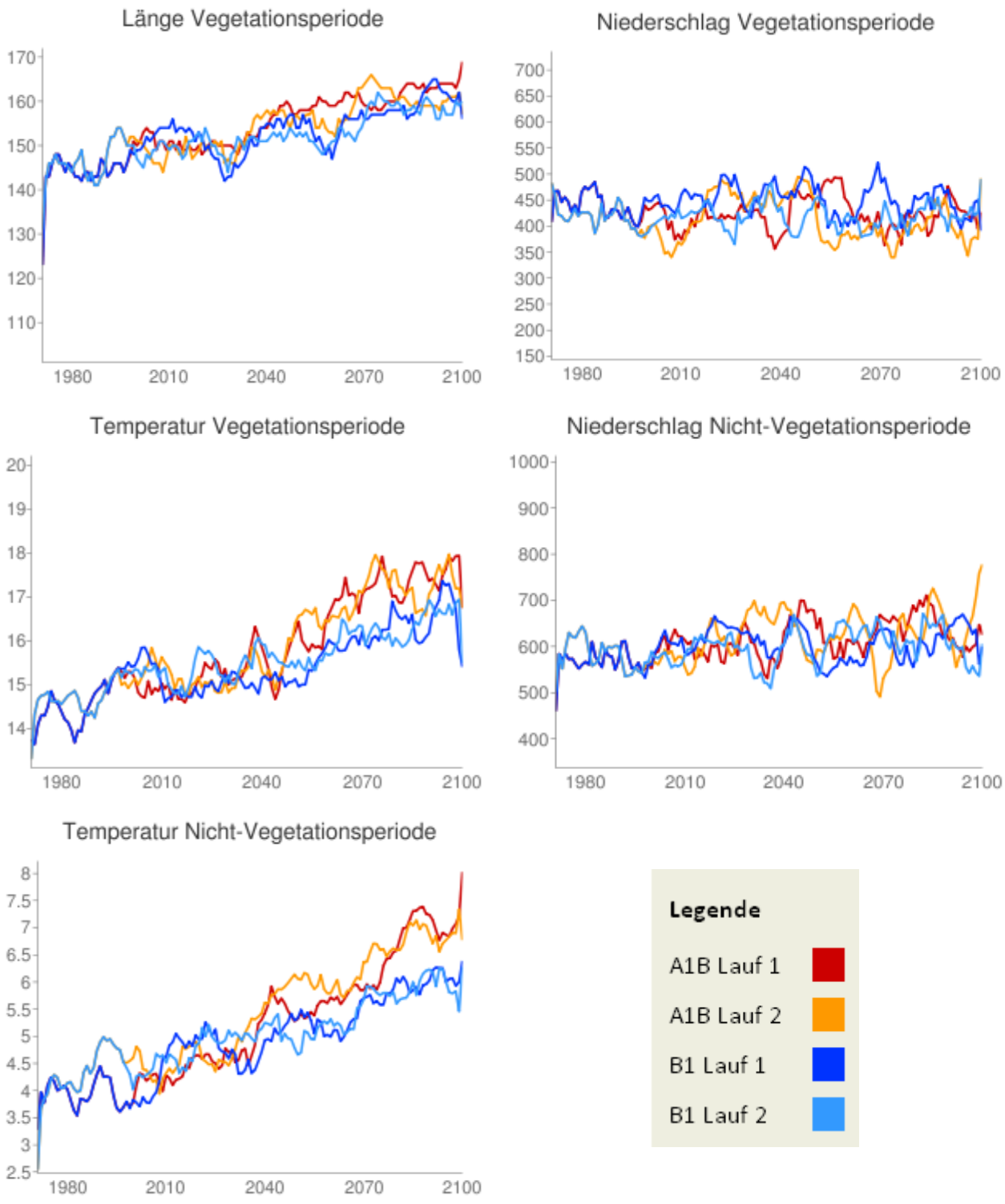


Abbildung 7: Liniendiagramm zum Szenarienvergleich der Eiche

A1B Lauf 1

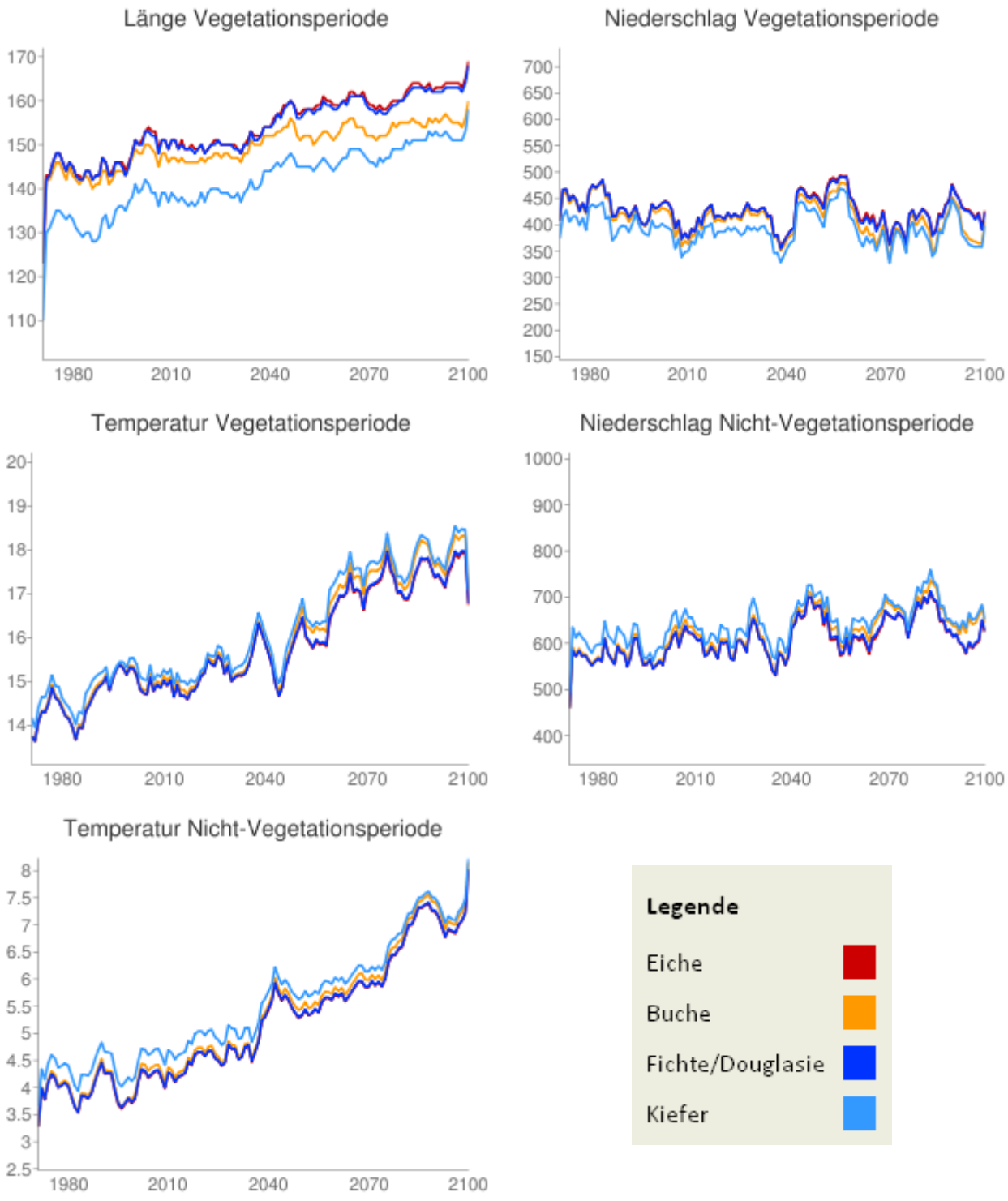


Abbildung 8: Liniendiagramm zum Vergleich der Baumarten in Szenario A1B

6.3 Daten für die Liniendiagramme

Der View, welcher die Nutzeranfrage verarbeitet, kann die Parameter des Requests aufgreifen und so über die Namen der Buttons prüfen, welcher Teil des Codes ausgeführt werden soll:

```
if 'scenario' in request.GET:
        <Code zur Validierung und –formatierung der Paramter>
        <Code zur Vorbereitung des Kontexts>
        <Code zum Rendrn des Templates >
if 'baumart' in request.GET:
        <Code zur Validierung und –formatierung der Paramter >
        <Code zur Vorbereitung des Kontexts>
        <Code zum Rendrn des Templates>
```

Der Code für Baumarten und Szenarien lässt sich in jeweils drei gleiche Blöcke gliedern: Die Validierung der Parameter, die Vorbereitung des Kontexts und das Rendern des Templates

Validierung

Der Code zur Validierung und Formatierung der Koordinaten ist für Baumarten und Szenarien gleich. Es wird geprüft, ob überhaupt Koordinaten eingegeben wurden

```
if not request.GET['lon'] or not request.GET['lat']:
        error.append('Bitte stellen Sie sicher, dass Sie Koordinaten angegeben haben')
```

und ob für die gewählten Koordinaten Daten in der Datenbank zur Verfügung stehen (Anhang K):

```
elif (float((request.GET['lon']).replace(',','.'))*10) < Diagramme.objects.aggregate(b=Min('lon'))['b']:
        error.append('Der Wert fuer Breitengrad war zu niedrig')
```

Dafür müssen die vom Nutzer übersandten Koordinaten in dieselbe Form gebracht werden, die auch in der Datenbank vorliegt. Gegebenenfalls wird eine Fehlermeldung an das Template übergeben und das Formular wird mit dieser Fehlermeldung erneut gerendert (Anhang L).

Die Fenstergröße des gleitenden Mittels muss zu dessen Berechnung eine ungerade Integer sein. Deshalb wird die durch den Nutzer gesendete Fenstergröße in eine Integer umgewandelt. Sollte sie dann gerade sein, wird sie um 1 erhöht:

```
w = int(round(float((request.GET['mean']).replace(',','.')),0)) #window-size, mustn't be even
        if w%2==0: #Wenn w gerade ist, dann wird w um 1 erhoeht
          w+=1
```

Vorbereitung des Kontexts

Als Kontext werden die Template-Variablen und ihre dazugehörigen Werte bezeichnet. Während des Renderns wird der Kontext genutzt, um die Inhalte des Templates zu füllen.

Es wird zum einen Kontext für die Skalierung der y-Achsen erzeugt (Anhang M), welcher für den Vergleich der Szenarien und der Baumarten jeweils gleich ist. Werden die Achsen automatisch skaliert, so erhalten Grafiken, die dieselbe Klimavariable darstellen, unterschiedliche Skalierungen, da sich Unterschiede aus den Baumarten und Szenarien ergeben. Um einheitlich skalierte Grafiken zu erzeugen, werden die Minimal- und Maximalwerte jeder Klimavariablen aus der Datenbank abgefragt und beim Rendern des Templates an den Funktionsaufruf übergeben.

Zum anderen müssen Template-Variablen mit den an die Javascript-Funktion zu übergebenden Daten bereitgestellt werden (vgl. Kapitel 6.2). Dies wird anhand der Daten, die für die Erzeugung der Diagramme für den Vergleich der Szenarien benötigt werden, erläutert. Der Zugriff auf die Datenbank erfolgt über das Model „Diagramme“ (Anhang N). Das Model „Diagramme“ regelt den Datenbankzugriff auf die Tabelle „diagramme“. Wichtig ist im Model die Angabe, dass die Datensätze bei einer Abfrage nach „years“ und „species“ sortiert sind. Es wird für jede Baumart und der durch den Nutzer angefragten Koordinate ein QuerySet erzeugt. Im folgenden Beispiel enthält das QuerySet „eiche“ alle in „diagramme“ vorliegenden Attribute (die Attribute beinhalten die Klimavariablen) an der angefragten Koordinate.

```
eiche = Diagramme.objects.filter(lon=lons, lat=lats, species=1)
```

In der Funktion processQuerys() (Anhang O) wird auf die Attribute (Klimavariablen) zugegriffen und deren Werte werden an NumPy-arrays (h1-h21) übergeben. Außerdem sind in ProcessQuerys() zwei weitere Funktionen, mov_avg() und liste(), eingebunden (Anhang O). mov_avg() wird benötigt, um die gleitenden Mittelwerte zu berechnen. Sie erhält als Argument unter anderem das Fenster (Zeitspanne) über welches das gleitende Mittel berechnet werden soll und ein durch processQuerys() bereits gefülltes NumPy-array. Mit mov_avg() wird ein beidseitiger gleitender Mittelwert berechnet. Soll ein gleitender Mittelwert über 5 Jahre gerechnet werden, hätte dies zur Folge, dass für die Jahre 1971, 1972, 2099 und 2100 kein gleitender Mittelwert berechnet werden könnte. Deshalb wird in diesen Randfällen das Fenster der Mittelwertbildung

verringert. Für 1972 und 2099 würde in diesem Fall nur über 3 Jahre gemittelt werden und für 1971 und 2100 würde keine Mittelwertbildung stattfinden. Diese Vorgehensweise hat den Vorteil, dass die Werte an den Randbereichen nicht einfach wegfallen oder ein einseitiger Mittelwert berechnet werden müsste. Die folgenden Formeln verdeutlichen das Vorgehen bei einem gleitenden Mittelwert über 5 Jahre (Der gleitende Mittelwert kann übrigens auch bei einer Fenstergröße von 1 berechnet werden). x_1 steht für Werte zum Zeitpunkt 1971 und x_{130} für Werte zum Zeitpunkt 2100.

$$x_1 = \frac{x_1}{1} \tag{3}$$

$$x_2 = \frac{x_1 + x_2 + x}{3} \tag{4}$$

$$x_3 = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{5} \tag{5}$$

$$x_4 = \frac{x_2 + x_3 + x_4 + x_5 + x_6}{5} \tag{6}$$

$$\vdots$$

$$x_{127} = \frac{x_{125} + x_{126} + x_{127} + x_{128} + x_{129}}{5} \tag{7}$$

$$x_{128} = \frac{x_{126} + x_{127} + x_{128} + x_{129} + x_{130}}{5} \tag{8}$$

$$x_{129} = \frac{x_{128} + x_{129} + x_{130}}{3} \tag{9}$$

$$x_{130} = \frac{x_{130}}{1} \tag{10}$$

Dementsprechend werden in mov_avg() fünf verschiedene Fälle zur Berechnung des gleitenden Mittels umgesetzt: 1) Zum Zeitpunkt 1971, 2) von 1972 bis zum Jahr, in dem mit der Berechnung des gewünschten gleitenden Mittels begonnen werden kann, 3) die reguläre Berechnung des gleitenden Mittels, 4) ab dem Jahr, in dem die Berechnung des gewünschten gleitenden Mittels nicht mehr möglich ist bis 2099 , 5) zum Zeitpunkt 2100. Dies gilt für alle möglichen Fenstergrößen.

Nachdem für jeweils 4 Attribute, die gemeinsam in einer Grafik dargestellt werden sollen, die gleitenden Mittelwerte berechnet sind, werden sie in eine geschachtelte Liste (vgl. Kaptel 6.2)

umgearbeitet. Dafür existiert die Funktion liste(). Alle Listen werden von processQuerys() zurückgegeben. Die Funktionsaufrufe für ProcessQuerys() werden im View verwendet und deren Rückgabewerte werden Template-Variablen zugeordnet. Das Template kann jetzt gerendert werden.

7 Diskussion

7.1 Die Anwendung

Die in dieser Arbeit entwickelte Internet-Anwendung bietet dem Nutzer eine klar und einfach strukturierte grafische Oberfläche. Deutliche Abgrenzungen einzelner Inhalte tragen zu einer guten Orientierung bei. Das Benutzen der Anwendung fällt somit auf Anhieb leicht. Mit einem Minimum an Aufwand (sehr wenige Mausklicks) erhält der Nutzer eine Vielzahl an Informationen, welche leicht erfassbar sind. Daher ist durchaus die Behauptung zulässig, dass eine funktionsfähige, attraktive Web-Anwendung entwickelt wurde, die auch inhaltlich überzeugt. Wie „gut“ oder „schlecht“ die Anwendung aber letztlich wirklich ist, wird sich erst herausstellen, wenn sie in Betrieb genommen wird und von den Nutzern positive oder negative Rückmeldungen kommen.

Vorab lässt sich die hier entwickelte Anwendung mit einer anderen Anwendung vergleichen, die im Rahmen des Projekts „Schutzgebiete Deutschlands im Klimawandel – Risiken und Handlungsempfehlungen“ vom Bundesamt für Naturschutzs entwickelt wurde. In deren Anwendung können über 4000 Schutzgebiete aus einer GoogleMaps-Karte ausgewählt werden. Für jedes dieser Schutzgebiete existieren Linien-, Balken-, und Walter-Lieth-Diagramme mit Informationen über Niederschlags- und Temperaturwerte, Kenntage und Klimatische Wasserbilanzen (nachzuvollziehen unter POTSDAM INSTITUE FOR CLIMATE IMPACT RESEARCH 2010). Die Jahresmitteltemperatur und Jahresniederschlagssumme werden für die Schutzgebiete in Form von Liniendiagrammen incl. einer Trendlinie angeboten. Zusätzlich werden auf Monatswerten basierende Differenzen zwischen dem vergangenen und zukünftigen Klima dargestellt. Referenzdaten und Projektionen werden in unterschiedlichen Diagrammen dargestellt. Übersichtskarten wurden in dieser Anwendung jedoch nicht bereitgestellt.

Mit der Schutzgebiet-Anwendung kann die im Rahmen dieser Arbeit entwickelte Anwendung durchaus mithalten. In den Liniendiagrammen sind keine fixen Trendlinien eingefügt, sondern der Nutzer erhält selbst die Möglichkeit, die Daten auf das gewünschte Maß zu glätten. Verschiedene Klimaprojektionen werden im selben Diagramm dargestellt und können dadurch gut miteinander verglichen werden. Die Klimavariablen, die sich auf die Vegetationsperiode und auf die Zeit außerhalb der Vegetationsperiode beziehen, spiegeln ähnliche Informationen wieder wie die in der Schutzgebiet-Anwendung vorhanden Jahres- und Monatswerte.

7.2 Klimadaten

Die Einschätzung der zukünftigen klimatischen Bedingungen an spezifischen Standorten zwecks passender Baumartenwahl hängt stark von den Klimaprojektionen ab. Wünschenswert sind Projektionen, die diese Bedingungen mit einiger Verlässlichkeit abbilden können. Dreh- und Angelpunkt der Projektionen sind die anthropogen verursachten Treibhausgasemissionen. Wie sich Emissionen oder auch deren Ursachen künftig entwickeln, ist ungewiss. Schon die 40 vom IPCC entwickelten Emissionsszenarien drücken es aus: Die Zukunft ist nicht vorhersagbar und es bleibt Ansichtssache, welches Szenario als das Wahrscheinlichste erachtet werden kann (IPCC 2000).

Die Szenarien A1B und B1 dienen als Grundlage für das gekoppelte Modell ECHAM5/MPI-OM. Generell lässt sich über gekoppelte Modelle sagen, dass sie klimatische Jahresmittelwerte glaubhaft simulieren, jedoch bestehen bei der Simulation von Wolken und Luftfeuchte noch große Unsicherheiten (IPCC 2001).

Die mit CLM herunterskalierten Daten können demnach erstens nur eine grobe Spannweite des zukünftigen Klimas abstecken. Die in den Karten und Diagrammen visualisierten Klimavariablen sollten nur als Annäherungen an zukünftig mögliche Bedingungen gesehen werden.

Zweitens muss die Darstellung von Klimavariablen genügen, die über längere Zeitabschnitte gemittelt werden (jährlich oder halbjährlich). So interessant z.B. die Betrachtung des Niederschlagsmusters innerhalb der Vegetationszeit auch wäre, sollten darüber jedoch keine Aussagen gemacht werden. Die fünf für die Visualisierung gewählten Klimavariablen sind demnach gut geeignet, um auf Grundlage von Klimaprojektionen auf die langfristige Entwicklung der klimatischen Gegebenheiten hinzudeuten.

Allerdings gestaltet sich ein direkter Vergleich der Klimavariablen zwischen den Szenarien als schwierig, da in verschiedenen Szenarien die Länge der Vegetationsperiode variiert. Um die Trends der einzelnen Szenarien besser zu erkennen, könnte man evtl. Kimavariablen darstellen, die sich auf den kalendarischen Sommer und Winter beziehen oder auf eine fixe Vegetationsperiode (z.B. 1. Mai-30. September).

7.3 Optimierung

Im Rahmen der in dieser Arbeit entwickelten Anwendung sind einige Punkte anzusprechen, die künftig noch optimiert werden sollten

Im Bereich der Datenhaltung muss versucht werden, die Performance zu erhöhen, denn momentan umfasst die Laufzeit des Aufbereitungsprozesses mehrere Tage. Möglicherweise kann über die weitere Nutzung von Vacuum Analyze (entfernt leere Zeilen aus der Tabelle und erstellt Statistiken über die Tabelle, welche zur Ermittlung des optimalen Queryplans beitragen) oder Explain (zeigt auf, welcher der Teil einer Anfrage wie lange dauert) eine Erhöhung der Performance erzielt werden. Auch die Erzeugung der Tabellen temperature1 und temperature2 kostet Zeit sowie Speicherplatz. Es sollte deshalb nach einer Lösung gesucht werden, die diesen Zwischenschritt nicht mehr benötigt.

Ebenfalls als Zwischenschritt bezeichnet werden kann die Erstellung der Tabellen diagramm_nveg_prec1, diagramm_nveg_prec2 etc. aus denen die Tabellen diagramm_a1b oder diagramm_b1 erzeugt werden. Es ist möglich, dies alles in einer einzigen SQL-Anweisung abzuarbeiten. Ein Punkt, der jedoch für den Zwischenschritt spricht, ist die Möglichkeit einen Primärschlüssel und damit auch einen Index auf diese Tabellen zu erzeugen, welcher in der SQL-Anweisung für die Erstellung der Tabellen diagramme_a1b und diagramme_b1 genutzt werden könnte. Durch die Untersuchung dieser beiden Möglichkeiten mit Explain sollte der effizientere Weg herausgefunden werden.

Die Daten aus dem 20. Jahrhundert (c_20) werden jeweils zeitlich vor die beiden Szenarien A1B und B1 gehängt. Dadurch werden sie sozusagen verdoppelt. Dies führt erstens zu unnötigem Speicherplatzverbrauch und erhöht zweitens auch den Rechenaufwand immens. Deshalb sollte auch hier eine andere Lösung gefunden werden.

In einer PostGIS-Datenbank ist die Reprojektion vom Koordinatensystem, in dem die Geometrien vorliegen in ein anderes Koordinatesystem möglich, sofern dieses Koordinatensystem in der Tabelle spatial_ref_sys eingetragen ist. Das Referenzsystem EPSG:900913 liegt in dieser Tabelle vor. Dies sollte also ausgenutzt werden, anstatt die GeoTIFFs mit gdalwarp zu reprojizieren, da anzunehmen ist, dass die Reprojektion in der Datenbank schneller und mit weniger Aufwand durchzuführen ist.

In der Web-Mapping-Anwendung wird bei Erzeugung der Legende standardmäßig der Titel des entsprechenden Layers mitangezeigt. Um das zu verhindern wurde kurzerhand der Quellcode

von GeoExt verändert. Obwohl aufgrund der BSD-Lizenz eine Veränderung am Quellcode vorgenommen werden darf, ist das prinzipiell keine gute Taktik, da sich die Anwendung dadurch nicht problemlos weitergeben lässt. Eine bessere Möglichkeit wäre, eine erweiterte Klasse des LegendPanel zu erzeugen, die keine Titelanzeige beinhaltet.
Die Deckkraft aller Rasterlayer wurde auf 70% gesetzt und lässt sich vom Nutzer nicht verändern. Da Menschen jedoch Farben unterschiedlich wahrnehmen, ist diese voreingestellte Deckkraft nicht für jeden Nutzer ideal. Um jedem Nutzer die Möglichkeit zu geben, selbst zu entscheiden, wie stark die Rasterlayer sichtbar sein sollen, böte sich die Einbindung eines Tranzparenzreglers an.

Im Bereich der dynamischen Liniendiagramme ist zunächst anzumerken, dass die Validierung der von den Nutzern abgeschickten Werte für die Längen- und Breitengrade nicht abschließend durchgeführt wurde. Eine clientseitige JavaScript basierte Validierung wurde im DSS-WuK bereits implementiert, wenn es darum geht, die Koordinaten eines Bestandes für die Baumartenbeurteilung auszuwählen und abzusenden. Dieses Script sollte sich problemlos für das Formular zum Erhalt der Liniendiagramme adaptieren lassen. Serverseitig wurde lediglich darauf geachtet, dass die Nutzer ihre Koordinaten mit einem Komma oder Punkt als Dezimaltrennzeichen eingeben können, dass die Zeichenlänge der Koordinaten keine Rolle spielt und dass die vom Nutzer angefragten Koordinaten auch in der Datenbank vorliegen. Der Wert für das gleitende Mittel darf vom Nutzer in Form von geraden, ungeraden oder Fließkommazahlen mit Punkt oder Komma als Dezimaltrennzeichen angegeben werden. Für das ganze Formular ist jedoch noch nicht geregelt, dass bei der Eingabe von Buchstaben oder Sonderzeichen serverseitig eine entsprechende Fehlermeldung gerendert werden soll.
In den Diagrammen ist eine farbliche Trennung der Szenarien A1B und B1 sowie ihren beiden Läufen vorgenommen. Die Klimavariablen des 20. Jahrhunderts werden momentan noch mit denselben Farben wie die Läufe des Szenarios B1 dargestellt. Möglicherweise sollte hier eine farbliche Abtrennung vorgenommen werden.
Die y-Achsenskalierung wurde mit den jeweils auftretenden Minimal- und Maximalwerten einer Klimavariablen durchgeführt. Bei einem gleitenden Mittel von 1 (d.h. kein gleitendes Mittel wird berechnet), füllen die angezeigten Daten nahezu die ganze Spannweite der y-Achse aus. Sobald ein gleitendes Mittel berechnet wird, erhöhen bzw. veringern sich die Minimal- und Maximalwerte, die in den Grafiken zu sehen sind. Entsprechend sollte sich dann auch die Achsen-

skalierung verändern, damit die Graphen über den gesamten Bereich der y-Achse gezeichnet werden können. Dies sollte noch umgesetzt werden.

Die Grafiken, welche die Länge der Vegetationszeit für alle Baumarten, jedoch getrennt nach Szenarienläufen darstellen, lassen deutliche Unterschiede zwischen den Baumarten erkennen (Abbildung 8). Mit dieser Erkenntnis besteht auch Interesse daran, die anderen Klimavariablen auf dieselbe Weise zu betrachten. Bei zufallsmäßiger Auswahl einiger Koordinaten hat sich jedoch gezeigt, dass für alle weiteren vier Klimavariablen, ebenso wie in Abbildung 8 zu sehen, kaum Unterschiede zwischen den Baumarten auftreten. Es könnte also auf diese Art der Darstellung verzichtet werden.

Ein generelles Problem bei der Entwicklung der dynamischen Liniendiagramme stellte der Internet Explorer dar. HTML5 ist der neueste HTML-Standard, der jedoch noch nicht von allen Browsern vollständig unterstützt wird. Mit diesem Standard wurde auch das <canvas> Element eingeführt (W3C 2010), welches hauptsächlich für die Erstellung von Grafiken genutzt wird. Leider wird es, im Gegensatz zu Open-Source Browsern wie Mozilla Firefox oder Google Chrome, vom Internet Explorer 8 noch nicht richtig unterstützt. Ältere Versionen des Internet Explorers (6 und 7), die noch weit verbreitet sind, unterstützen dieses Element überhaupt nicht. Bei der Suche nach Möglichkeiten, wie die Liniendiagramme in einer interaktiven Form erstellt werden könnten, erwies sich die dygraph Visualization Library als ideale Lösung (Beispiel unter VANDERKAM 2008). Sie nutzt jedoch das <canvas> Element und lässt sich dadurch nicht mit dem Internet Explorer verwenden. Theoretisch existiert ein workaround, so dass dygraphs auch im Internet Explorer problemlos verwendet werden können (VANDERKAM 2008), aber leider erwies sich dieser als nicht praktikabel aufgrund extrem langer Ladezeiten und teilweise fehlenden Inhalten in den Grafiken. Im Internet Explorer 9 soll das <canvas> Element jedoch unterstützt werden, so dass zu überlegen wäre, die Erstellung der Liniendiagramme in Zukunft auf eine JavaScript-Bibliothek umzustellen, die das <canvas> Element ausnutzt.

8 Schlussfolgerungen

Die Entwicklung einer Web-Anwendung zur Visualisierung forstlich relevanter Klimavariablen als deutschlandweite Übersichtskarten und als zeitliche Verläufe in Liniendiagrammen wurde erfolgreich unter Ausnutzung etablierter (PostgreSQL) oder noch sehr junger (z.B. GeoExt, Django) Software durchgeführt. Leider konnte die Anwendung nicht unter der Ausnutzung des HTML5-Standards entwickelt werden. Dementsprechend sollt sie nach einiger Zeit überarbeitet werden, um den Nutzern größtmögliche Funktionalität zu bieten.

Die dargestellten Klimavariablen werden den Nutzern Aufschluss über zukünftige Standortsbedingungen geben. Wenn im Laufe der Zeit die Klimaprojektionen immer sicherer werden, sollten weitere interessante Variablen, wie die bereits angesprochene Niederschlagsverteilung in der Vegetationsperiode mit in die Anwendung aufgenommen werden.

Auch die Modelle, welche versuchen, die Länge der Vegetationsperiode zu quantifizieren, werden sich in Zukunft noch verbessern. Dementsprechend wird die entwickelte Web-Anwendung auch in diesem Bereich angepasst werden müssen. Zum jetzigen Zeitpunkt erfüllt sie jedoch voll und ganz ihren Zweck und wird allen interessierten Forstpraktikern eine Hilfestellung bei der Abschätzung möglicher Risiken für die Baumarten Eiche, Buche, Fichte, Douglasie und Kiefer im künftig zu erwartenden Klima sein.

9 Literatur

AMERELLER K., KÖLLING C., BOLTE A., EISENHAUER D.-R., GRO? J.,HANEWINKEL M., PROFFT I., RÖHE P. 2009. Die „20 Freisinger Punkte“ – Gemeinsame Basis der deutschsprachigen forstlichen Ressortforschung. AFZ-Der Wald 17, 916-918

BLÜMEL W. D. 2002: 20.000 Jahre Klimawandel und Kulturgeschichte – von der Eiszeit in die Ge-genwart. In: Wechselwirkungen, Jahrbuch aus Lehre und Forschung der Universität Stuttgart, Referat für Presse- und Öffentlichkeitsarbeit der Universität, 2002, S. 2-19.

BOLTE A., IBISCH P. L. 2007. Neun Thesen zu Klimawandel, Waldbau und Waldnaturschutz. AFZ-Der Wald (11), 572-576

BOLTE, A, IBISCH P., MENZEL A., ROTHE A. 2008. Was Klimahüllen uns verschweigen. AFZ-Der Wald 15, S. 800-803

BUTLER H., SCHMIDT C., SPRINGMEYER D., LIVNI J. 2010. SR-ORG:6.
http://spatialreference.org/ref/sr-org/6/
letzter Zugriff: 12.07.2010

CLIMATE RESEARCH UNIT 2008. Data - Temperature.
http://www.cru.uea.ac.uk/cru/data/temperature/
letzter Zugriff:03.05.2010

DITTMAR D., FRICKE W., ELLING W. 2006. Impact of late frost events on radial growth of common beech (*Fagus sylvatica* L.) in Southern Germany. Eur J Forest Res 125: 249-259

DJANGO SOFTWARE FOUNDATION 2010. GeoDjango Tutorial.
http://docs.djangoproject.com/en/dev/ref/contrib/gis/tutorial/
letzter Zugriff: 15.07.2010

ELMASRI R. A., NAVATHE S. B. 2002. Grundlagen von Datenbanksystemen. Pearson Education Deutschland GmbH.

FANG J., LECHOWICZ M. J. 2006. Climatic limits for the present distribution of beech (*Fagus L.*) species in the world. J. Biogeogr. (33), 1804-1819

GERSTENGARBE F.-W., WERNER P. C., HAUF Y. 2004. Bericht zum Werkvertrag Erstellung regionaler Klimaszenarien für Nordrhein-Westfalen Nr 2-53710-2233 zwischen der Landesanstalt für Ökologie, Bodenordnung und Forsten, Nordrhein-Westfalen und der Firma BRUECKE-Potsdam GbR

GEOEXT COMMUNITY 2010. GeoExt – API Reference.
http://www.geoext.org/lib/index.html
letzter Zugriff: 13.07.2010

GOOGLE 2010. Getting started with charts.
http://code.google.com/intl/de-DE/apis/chart/docs/making_charts.html
letzter Zugriff: 15.07.2010

HAHN, S. 2009. Dem Klimawandel in Bayerns Wäldern mit Forschung begegnen. AFZ-Der Wald 17, S. 914-916

HARTWIG, J. 2001. PostgreSQl – Professionell und praxisnah. Addison-Wesley Verlag.

HEINO R., BRAZDIL R., FØRLAND E., TUOMENVIRTA H., ALEXANDERSSON H., BENISTON M., PFISTER C., REBETEZ M., ROSENHAGEN G., RÖSNER S., WIBIG J. 1999. Progress in the study of climatic extremes in northern and central Europe. Climatic Change 42: 151-181

HOLLWEG H-D., BÖHM U., FAST I., HENNEMUTH B., KEULER K., KEUP-THIEL E., LAUTENSCHLAGER M., LEGUTKE S., RADTKE K., ROCKEL B., SCHUBERT M., WILL A., WOLDT M., WUNRAM C. 2008. Technical Report - Ensemble Simulations over Europe with the Regional Climate Model CLM forced with IPCC AR4 Global Scenarios. Max-Planck-Institut für Meteorologie, Hamburg.

HOLOVATY A., KAPLAN-MOSS J. 2009. The Definitive Guide to Django – Web Development Done Right, Second Edition. Springer-Verlag New York.

INSTITUT DER DEUTSCHEN WIRTSCHAFT KÖLN 2010. Die Fördermaßnahme „klimazwei - Forschung für den Klimaschutz und Schutz vor Klimawirkungen".
http://www.klimazwei.de/F%C3%B6rderma%C3%9Fnahme/tabid/64/Default.aspx
letzter Zugriff:01.05.2010

IPCC 2000. Emissions Scenarios, Special Report, Cambridge University Press, UK. pp 570

IPCC 2001. Climate Change 2001. The Scientific Basis, Cambridge University Press

IPCC 2004. 16 Years of scientific assessment in support of the climate convention. Anniversary-brochure, World Meteorological Organization

JANSEN M., DÖRING C., AHRENDS B., BOLTE A., CZAIJKOWSKI T., PANFEROV O., ALBERT M., SPELLMANN H., NAGEL J., LEMME H., HABERMANN M., STAUPENDAHL K., MÖHRING B., BÖCHER M., STORCH S., KROTT M., NUSKE R., THIELE J. C., NIESCHULZE J., SABOROWSKI J., BEESE F. Anpassungsstrategien für eine nachhaltige Waldbewirtschaftung unter sich wandelnden Klimabedingungen – Entwicklung eines Entscheidungsunterstützungssystems „Wald und Klimawandel" (DSS-WuK). Forstarchiv 79, 121-142

JUMP A.S., HUNT J. M., PENUELAS. J. 2006. Rapid climate change-related growth decline at the southern range edge of *Fagus sylvatica*. Global Change Biology (12), 2163-2174

KÖLLING C. 2007. Klimahüllen für 27 Baumarten. AFZ-Der Wald 23, S. 1242-1245

KÖLLING C., KONNERT M., SCHMIDT O. 2008. Wald und Forstwirtschaft im Klimawandel – Antworten auf 20 häufig gestellte Fragen. AFZ-Der Wald 15, 804-807

KROPP J., HOLSTEN A., LISSNER T., ROITHMEIER O., HATTERMANN F., HUANG S., ROCK J., WECHSUNG F., LÜTTGER A., POMPE S., KÜHN I., COSTA L.., STEINHÄUSER M., WALTHER C., KLAUS M., RITCHIE S., METZGER M. 2009. Klimawandel in Nordrhein- Westfalen - Regionale Abschätzung der Anfälligkeit ausgewählter Sektoren. Abschlussbericht des Potsdam-Instituts für Klimafolgenforschung (PIK) für das Ministerium für Umwelt und Naturschutz, Landwirtschaft und Verbraucherschutz Nordrhein-Westfalen (MUNLV).

LE TREUTH H., SOMERVILLE R., CUBASCH U., DING Y., MAURITZEN C., MOKSSIT A., PETERSON T., PRATHER M. 2007. Historical Overview of Climate Change. In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M. Tignor and H.L. Miller (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

LINDNER M., MAROSCHEK M., NETHERER S., KREMER A., BARBATI A., GARCIA-GONZALO J., SEIDL R., DELZON S., CORONA P., KOLSTRÖM M., LEXER M. J., MARCHETTI M. 2010. Climate change impacts, adaptive capacity and vulnerability of European forest ecosystems. Forest Ecology and Management 259, 698-709

MAHAMMADZADEH M., BIEBELER H., BARDT H. 2009. Klimaschutz und Anpassung an die Klimafolgen-Strategien, Maßnahmen und Anwendungsbeispiele. Institut der deutschen Wirtschaft Köln

MATTHEW N., STONES R. 2005. Beginning Databases with PostgreSQL – From Novice to Professional, Second Edition. Springer-Verlag New York.

MENZEL A. 1997. Phänologie von Waldbäumen unter sich ändernden Klimabedingungen - Auswertung der Beobachtungen in den internationalen Phänologischen Gärten und Möglichkeiten der Modellierung von Phänodaten. Forstliche Forschungsberichte München 164

MITCHELL T. 2005. Web Mapping Illustrated. O'Reilly Media Sebastopol

MOMJIAN B. 2001. PostgreSQL – Einführung und Konzepte. Addison-Wesley Verlag

OSGEO 2010a. OpenLayers Info Sheet
http://www.osgeo.org/openlayers
letzter Zugriff: 07.07.2010

OSGEO 2010b. OpenLayers.
http://trac.openlayers.org/wiki/SphericalMercator
letzter Zugriff: 12.07.2010

OSGEO 2010c. OpenLayers – Class Documentation
http://dev.openlayers.org/releases/OpenLayers-2.9.1/doc/apidocs/files/OpenLayers-js.html
letzter Zugriff: 13.07.2010

PEARSON, R., DAWSON, T. 2003. Predicting the impacts of climate change on the distribution of species: are bioclimate envelope models useful? Global Ecology & Biogeography 12, 361-371

PETERCORD R., VEIT H., DELB H., SCHRÖTER H. 2008. Forstinsekten im Klimawandel – alte Bekannte mit neuem Potenzial. FVA-einblick^{+} 1, 36-39

POTSDAM INSTITUTE FOR CLIMATE IMPACT RESEARCH 2010. Klimawandel und Schutzgebiete.
http://www.pik-potsdam.de/infothek/klimawandel-und-schutzgebiete
letzter Zugriff: 27.07.2010

RASPE S., SCHULZ C., KROLL F. 2004. Wenn schon im Sommer tonnenweise Blätter fallen. LWF aktuell 43, 11-13

REICH P. B:, OLEKSYN J. 2008. Climate warming will reduce growth and survival of Scots Pine except in the far north. Ecology Letters 11: 588-597

REW R., DAVIS G., EMMERSON S., DAVIES H., HARTNETT E., HEIMGIGNER D. 2010. The NetCDF Users Guide. University Corporation for Atmospheric Research

RITTER N., RUTH M. 2000. GeoTIFF Format Specification.
http://www.remotesensing.org/geotiff/spec/geotiffhome.html
letzter Zugriff: 14.07.2010

ROECKNER E., BRASSEUR G. P., GIORGETTA M., JACOB D., JUNCLAUS J., REICK C., SILLMANN J. 2006. Klimaprojektionen für das 21. Jahrhundert. Max-Planck-Institut für Meteorologie, Hamburg

SCHLYTER P., STJERNQUIST I., BÄRRING L., JÖNSSON A. M., NILSSON C. 2006. Assessment of the impacts of climate change and weather extremes on boreal forests in northern Europe, focusing on Norway spruce. Clim Res 31: 75-84

SCHULZE E.-D., HESSENMÖLLER D., SEELE C., WÄLDCHEN J., VON LÜPKE N. 2010. Die Buche – eine Kultur- und Wirtschaftsgeschichte. Biol. Unserer Zeit Vol 40 (3), 171-183

SENCHA INC. 2010a. Ext JS – Ext JS Overview.
http://www.sencha.com/products/js/
letzter Zugriff: 08.07.2010

SENCHA INC. 2010b. Ext JS 3.2.1 API Documentation.
http://www.sencha.com/deploy/dev/docs/
letzter Zugriff: 13.07.2010

SPEKAT, J. et al. 2006. Fortschreibung der Klimaszenarien für Nordrhein-Westfalen. Landesanstalt für Ökologie, Bodenordnung und Forsten Nordrhein-Westfalen.
http://www.lanuv.nrw.de/klima/pdf/NRW_2006.pdf

SOLBERG S., DOBBERTIN M., REINDS G. J., LANGE H., ANDREASSEN K., FERNANDEZ P. G., HILDINGSSON A.,DE VRIES W. 2009. Analyses of the impact of changes in atmospheric deposition and climate on forest growth in European monitoring plots: A stand growth approach. Forest Ecology and Management 258, 1735-1750

TINNER W., LOTTER A. F. 2001. Central European vegetation response to abrupt climate change at 8.2 ka. Geology 2001, (29) 551-554

UNIVERSITY OF MINNESOTA 2010a. Mapserver.
http://mapserver.org/about.html
letzter Zugriff: 07.07.2010

UNIVERSITY OF MINNESOTA 2010b. An Introduction to Mapserver.
http://www.mapserver.org/introduction.html
letzter Zugriff: 11.07.2010

UNIVERSITY OF MINNESOTA 2010c. Mapserver 5.6.3 Documentation.
http://www.mapserver.org/documentation.html
letzter Zugriff:12.07.2010

VANDERKAM D. 2008. dygraphs JavaScript Visualization Library.
http://danvk.org/dygraphs/
letzter zugriff: 27.07.2010

VON WILPERT K. 1990. Die Jahrringstruktur von Fichten in Abhängigkeit vom Bodenwasserhaushalt auf Pseudogley und Parabraunerde – Ein Methodenkonzept zur Erfassung standortsspezifischer Wasserstressisposition. Dissertation, Institut für Bodenkunde und Waldernährungslehre der Albert-Ludwigs-Universität Freiburg i. Br.

W3C 2010. HTML5 differences from HTML4.
http://dev.w3.org/html5/html4-differences/
letzter Zugriff: 27.07.2010

WARMERDAM F. 2009a. Gdal_grid.
http://www.gdal.org/gdal_grid.html
letzter Zugriff: 12.07.2010

WARMERDAM F. 2009b. gdalwarp.
http://www.gdal.org/gdalwarp.html
letzter Zugriff: 12.07.2010

WARMERDAM F. 2010. FWTools: Open Source GIS Binary Kit for Windows and Linux.
http://fwtools.maptools.org/
http://www.gdal.org/gdal_utilities.html
http://www.gdal.org/formats_list.html
http://www.gdal.org/ogr/ogr_formats.html
Letzter Zugriff: 14.07.2010

WORLD METEOROLOGICAL ORGANIZATION 1997. WMO Statement on the status of the global climate in 1996.
http://www.wmo.int/pages/prog/wcp/wcdmp/statement/documents/wmo858.pdf

10 Anhang

A Tabellen der Rohdaten der Datenbank

- a1b_1_prec (tägliche Niederschlagssummen 01.01.2001 – 31.12.2100 A1B Lauf1)
- a1b_1_temp (Tagesmitteltemperaturen 01.01.2001 – 31.12.2100 A1B Lauf 1)
- a1b_2_prec (tägliche Niederschlagssummen 01.01.2001 – 31.12.2100 A1B Lauf 2)
- a1b_2_temp (Tagesmitteltemperaturen 01.01.2001 – 31.12.2100 A1B Lauf 2)
- b1_1_prec (tägliche Niederschlagssummen 01.01.2001 – 31.12.2100 B1 Lauf 1)
- b1_1_temp (Tagesmitteltemperaturen 01.01.2001 – 31.12.2100 B1 Lauf 1)
- b1_2_prec (tägliche Niederschlagssummen 01.01.2001 – 31.12.2100 B1 Lauf 2)
- b1_2_temp (Tagesmitteltemperaturen 01.01.2001 – 31.12.2100 B1 Lauf 2)
- c20_2_prec (tägliche Niederschlagssummen 01.01.1960 – 31.12.2000 Lauf 2)
- c20_2_temp (Tagesmitteltemperaturen 01.01.1960 – 31.12.2000 Lauf 2)
- c20_3_prec (tägliche Niederschlagssummen 01.01.1960 – 31.12.2000 Lauf 3)
- c20_3_temp (Tagesmitteltemperaturen 01.01.1960 – 31.12.2000 Lauf 3)

B Vorbereitung

```
-- Erzeugen einer Koordinatentabelle, welche als ID lat/lon-Werte erhält
CREATE TABLE coordinates(

--co_id     int4,
  lon       int4,
  lat       int4,
  PRIMARY KEY(lon,lat)

);
-- Anfuegen einer Geometriespalte
SELECT AddGeometryColumn('coordinates', 'coord', 4326, 'POINT', 2);

-- Extraktion der Punktgeometrien aus der Tabelle a1b_1_prec
INSERT INTO coordinates(/*co_id, */coord, lon, lat)
SELECT DISTINCT /*a.lat * 10000 + a.lon,*/ ST_SetSRID(ST_Point((a.lon::float / 10),(a.lat::float / 10)), 4326),
lon, lat
  FROM a1b_1_prec a;


-- Erzeugen einer Tabelle, die alle Temperaturdaten von Scenario 1 enthält
-- Für den Übergang von 2000 zu 2001 ist es im Schleifendurchlauf in Script 2 und 3
-- einfacher, auf eine Tabelle zuzugreifen, anstatt auf zwei
CREATE TABLE temperature1 AS

  SELECT "T_2M_AV", time, lon, lat FROM a1b_1_temp
  UNION
  SELECT "T_2M_AV", time, lon, lat FROM c20_2_temp WHERE time >= '1969-01-01';

-- Umrechnung von Kelvin in Celsius
ALTER TABLE temperature1
  ADD temp real;

UPDATE temperature1
  SET temp = "T_2M_AV"-271.15;

ALTER TABLE temperature1
  DROP "T_2M_AV";

-- Erweitern der Tabelle temperature um Tage, Monate und Jahre
-- Ermöglicht die Erstellung eines Indexes auf diese Attribute
-- sowie variablen Zugriff auf diese Attribute (s. Skript 2 und 3)
ALTER TABLE temperature1
  ADD column jahr int4, ADD column monat int4, Add column tage int4;

UPDATE temperature1
  SET jahr = EXTRACT(YEAR FROM time),
  monat = EXTRACT(Month fROM time),
  tage = EXTRACT(day FROM time);
```

```
-----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
-- Erzeugen einer Tabelle, die alle Temperaturdaten enthält von Scenario 2 enthält
CREATE TABLE temperature2 AS

  SELECT "T_2M_AV", time, lon, lat FROM a1b_2_temp
  UNION
  SELECT "T_2M_AV", time, lon, lat FROM c20_3_temp WHERE time >= '1969-01-01';

-- Umrechnung von Kelvin in Celsius
ALTER TABLE temperature2
  ADD temp real;

UPDATE temperature2
  SET temp = "T_2M_AV"-271.15;

ALTER TABLE temperature2
  DROP "T_2M_AV";

-- Erweitern der Tabelle temperature um Tage, Monate und Jahre
-- Ermöglicht die Erstellung eines Indexes auf diese Attribute
-- sowie variablen Zugriff auf diese Attribute (s. Skript 2 und 3)
ALTER TABLE temperature2
  ADD column jahr int4, ADD column monat int4, Add column tage int4;

UPDATE temperature2
  SET jahr = EXTRACT(YEAR FROM time),
  monat = EXTRACT(Month fROM time),
  tage = EXTRACT(day FROM time);

-----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
-- Erzeugen einer Tabelle, die Niederschlagsdaten enthält
-- Ein View reicht
-- Daten werden für die Visualisierung erst ab 1971 verwendet,
-- deshalb können hier alle Daten vor 1969 vernachlässigt werden
-- Im Rahmen späterer Aufbereitungen fallen 1969 und 1970 noch weg
CREATE VIEW precipitation1 AS

  SELECT "PRECIP_TOT", time, lon, lat FROM a1b_1_prec
  UNION
  SELECT "PRECIP_TOT", time, lon, lat FROM c20_2_prec WHERE time >= '1969-01-01';

CREATE VIEW precipitation2 AS

  SELECT "PRECIP_TOT", time, lon, lat FROM a1b_2_prec
  UNION
  SELECT "PRECIP_TOT", time, lon, lat FROM c20_3_prec WHERE time >= '1969-01-01';
```

```
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------
-- Erzeugen einer Tabelle, die alle Menzelparameter enthält (notwendig zur Berechnung der Vegetationspe-
riode)
CREATE TABLE menzel(

  menzel_id         serial    PRIMARY KEY,
  treespecies       int4,
  parms             float[]

);

INSERT INTO menzel(treespecies, parms)
  VALUES (110, '{9.0, 4.0, 1748, -298, 28, 279, 10, 56, 0.0}'),-- Quercus robur
         (211, '{9.0, 6.0, 1922, -348, 28, 279, 10, 56, 0.0}'),-- Fagus sylvatica
         (511, '{9.0, 4.0, 1848, -317, 28, 279, 10, 56, 0.8}'),-- Picea Abies und Douglasie
         (711, '{9.0, 5.0, 1395, -223, 28, 279, 10, 56, 0.6}');-- Pinus Sylvestris

-- Erzeugen eines Multipolygons, welches Deutschland umfasst
-- stammt ursprünglich aus einem Shapefile und die
-- folgende SQL Anweisung aus einem Backup
CREATE TABLE germany2 (
   gid integer NOT NULL,
   cat bigint
);

SELECT AddGeometryColumn('germany2', 'the_geom', 4326, 'MULTIPOLYGON', 2);
INSERT INTO germany2 (gid, cat, the_geom) VALUES (1, 2,
'0106000020E6100000010000000103000000010000000F01000000000040666621400000000000404B4000
0000406666214000000A099594B4000000040666621400000004033734B40000000C0CCCC214000000040
33734B40000000E0CCCC2140000000 4033734B4000000020333322400000004033734B4000000040333322
400000004033734B4000000080999922400000004033734B40000000C0999922400000004033734B400000
00E0FFFF22400000004033734B4000000020000023400000004033734B400000004066662340000000 4033
734B4000000080666623400000004033734B40000000E0CCCC2340000004033734B40000000E0CCCC2340
000000A099594B4000000040333324400000000A099594B400000004033332440000000000404B40000000
8099992440000000000404B40000000C0999924400000000000404B40000000E0FFFF2440000000000040
4B4000000020000025400000000000404B4000000040666625400000000000404B4000000080666625400
0000000404B40000000C0CCCC2540000000000404B40000000E0CCCC254000000000000404B400000002
033332640000000000404B4000000040333326400000000000404B40000000C0999926400000000000404
B40000000C099992640000000 6066264B4000000040333326400000006066264B400000004033332640000
000C0CC0C4B400000008099992640000000C0CC0C4B40000000C099992640000000C0CC0C4B40000000E0
FFFF2640000000C0CC0C4B400000002000002740000000C0CC0C4B400000004066662740000000C0CC0C4B
40000000406666274000000 6066264B40000000C0CCCC2740000006066264B40000000E0CCCC27400000
006066264B400000002033332840000006066264B400000004033332840000000 6066264B400000008099
99284000000 6066264B400000008099992840000000000404B40000000E0FFFF2840000000000404B40
000000200002940000000000404B400000004066662940000000000404B400000008066662940000000
0000404B40000000C0CCCC2940000000000404B40000000E0CCCC2940000000000404B4000000020333
32A400000000000404B400000002033332A40000000A099594B400000008099992A40000000A099594B40
000000C099992A40000000A099594B40000000E0FFFF2A40000000A099594B400000002000002B40000000
A099594B400000008066662B40000000A099594B400000008066662B400000000000404B4000000080666
62B400000006066264B400000002000002B400000006066264B400000002000002B40000000C0CC0C4B40
0000004066662B40000000C0CC0C4B400000008066662B40000000C0CC0C4B40000000C0CCCC2B4000000
0C0CC0C4B40000000E0CCCC2B40000000C0CC0C4B400000002033332C40000000C0CC0C4B400000004033
332C40000000C0CC0C4B40000000C099992C40000000C0CC0C4B40000000C099992C400000004033F34A4
0000000C099992C40000000A099D94A400000002000002D40000000A099D94A400000002000002D400000
000000C04A400000002000002D400000006066A64A400000002000002D40000000C0CC8C4A40000000200
0002D400000004033734A400000002000002D40000000A099594A400000008066662D40000000A099594A
400000008066662D400000000000404A400000008066662D400000006066264A40000000E0CCCC2D40000
```

0006066264A40000000E0CCCC2D40000000C0CC0C4A40000000E0CCCC2D400000004033F34940000000E0
CCCC2D40000000A099D94940000000E0CCCC2D400000000000C0494000000004033332E400000000000C04
9400000004033332E400000006066A6494000000004033332E40000000C0CC8C4940000004033332E40000
000403373494000000E0CCCC2D4000000040337349400000000E0CCCC2D40000000A09959494000000080
66662D40000000A0995949400000004066662D40000000A0995949400000002000002D40000000A099594
940000000E0FFFF2C40000000A099594940000000C099992C40000000A0995949400000008099992C40000
000A0995949400000004033332C40000000A0995949400000002033332C40000000A099594940000000E0C
CCC2B40000000A099594940000000C0CCCC2B40000000A0995949400000008066662B40000000A0995949
400000008066662B400000000000404940000000200002B400000000000404940000000E0FFFF2A400000
0000004049400000000C099992A400000000000040494000000080999 92A400000000000404940000000040333
332A400000000000404940000000 4033332A400000006066264940000000E0CCCC2940000006066264940
000000C0CCCC2940000000606626494000000080666629400000006066264940000000406666294000000 0
6066264940000000200002940000000606626494000000020000029400000 00C0CC0C4940000000200000
29400000004033F34840000000806666294000000 04033F34840000000806666294000000A099D9484000
0000806666294000000000000C04840000000C0CCCC2940000000000C04840000000E0CCCC294000000000
00C04840000004033332A400000000000C04840000004033332A400000006066A64840000000C099992
A400000006066A64840000000C099992A40000000C0CC8C4840000000E0FFFF2A40000000C0CC8C484000
00002000002B40000000C0CC8C484000000 08066662B40000000C0CC8C484000000 08066662B400000004
033734840000000E0CCCC2B400000004033734840000000E0CCCC2B40000000A099594840000000E0CCCC
2B400000000004048400000008066662B400000000004048400000004066662B4000000000040484000
00002000002B400000000004048400000002000002B400000006066264840000000C099992A4000000060
662648400000008099992A400000006066264840000004033332A400000006066264840000004033332A
40000000C0CC0C4840000004033332A400000004033F34740000004033332A40000000A099D94740000
0004033332A400000000000C04740000000E0CCCC2940000000000C04740000000C0CCCC294000000000
00C0474000000806666294000000000000C0474000000040666629400000000000C0474000000200000 29
400000000000C04740000000E0FFFF28400000000000C04740000000C0999928400000000000C047400000
00809999284000000000C0474000000403333284000000000C0474000000020333328400000000000
C04740000000E0CCCC27400000000000C04740000000C0CCCC27400000000000C0474000000 8066662740
000000000C0474000000406666274000000000C0474000000020000027400000000000C04740000000
20000027400000006066A64740000000C0999926400000006066A6474000000080999926400000006066A6
474000000040333326400000006066A6474000000020333326400000006066A64740000000E0CCCC254000
00006066A64740000000C0CCCC25400000006066A6474000000080666625400000006066A647400000004
066662540000000 6066A6474000000020000025400000006066A64740000000E0FFFF24400000006066A64
740000000C0999924400000006066A6474000000080999924400000006066A64740000000 4033332440000
0006066A6474000000020333324400000006066A64740000000C0CCCC23400000006066A64740000000C0
CCCC2340000000000C0474000000080666623400000000000C0474000000040666623400000000000C047
4000000020000023400000000000C04740000000E0FFFF224000000000000C04740000000C0999922400000
000000C0474000000080999922400000000000C0474000000040333322400000000000C047400000002033
33224000000000000C04740000000E0CCCC214000000000000C04740000000C0CCCC2140000000000C04740
000000806666214000000000000C0474000000040666621400000000000C0474000000020000021400000000
0000C04740000000E0FFFF204000000000000C04740000000C0999920400000000000C0474000000809999
204000000000000C0474000000040333320400000000000C0474000000020333320400000000000C0474000
0000A099991F400000000000C04740000000E0CCCC1E400000000000C04740000000C0CCCC1E4000000000
00C0474000000000000001E400000000000C0474000000000000001E40000000A099D947400000000000001E
400000004033F34740000000000000001E40000000C0CC0C484000000000000001E4000000060662648400000
00C0CCCC1E400000006066264840000000C0CCCC1E40000000000040484000000C0CCCC1E40000000A09
9594840000000A099991F40000000A099594840000000A099991F400000004033734840000000E0CCCC1E4
000000040337348400000000C0CCCC1E400000004033734840000000000000001E400000004033734840000000
00000001E40000000C0CC8C484000000004033331D40000000C0CC8C484000000002033331D40000000C0CC
8C484000000006066661C40000000C0CC8C4840000000A099991B40000000C0CC8C4840000000E0CCCC1A4
0000000C0CC8C4840000000C0CCCC1A40000000C0CC8C484000000000000001A40000000C0CC8C48400000
000000001A400000006066A6484000000020333319400000006066A648400000000203333194000000000000
C048400000002033331940000000A099D948400000006066661840000000A099D9484000000606666184
00000004033F34840000000606666184000000C0CC0C494000000060666618400000006066264940000000
060666618400000000000404940000000606666184000000A099594940000000A099991740000000A0995
94940000000A099991740000000403373494000000A099991740000000C0CC8C4940000000A099991740
0000006066A64940000000A099991740000000000C04940000000A099991740000000A099D9494000000
0A099991740000004033F34940000000606666184000000004033F34940000000203333194000000004033F

```
34940000000403333194000000004033F34940000000000000001A400000004033F34940000000000000001A400
00000C0CC0C4A40000000C0CCCC1A40000000C0CC0C4A40000000C0CCCC1A400000006066264A4000000
0C0CCCC1A400000000000404A40000000C0CCCC1A40000000A099594A40000000A099991B40000000A099
594A40000000A099991B400000004033734A400000006066661C400000004033734A400000006066661C4
0000000C0CC8C4A400000006066661C400000006066A64A40000000A099991B400000006066A64A400000
00A099991B400000000000C04A400000006066661C400000000000C04A400000006066661C40000000A09
9D94A400000002033331D40000000A099D94A400000004033331D40000000A099D94A400000000000001
E40000000A099D94A40000000C0CCCC1E40000000A099D94A40000000E0CCCC1E40000000A099D94A400
00000A099991F40000000A099D94A4000000020333320400000000A099D94A400000004033332040000000
A099D94A400000008099992040000000A099D94A40000000C099992040000000A099D94A40000000E0FFF
F2040000000A099D94A40000000E0FFFF20400000004033F34A400000004066662140000004033F34A400
000008066662140000004033F34A40000000C0CCCC21400000004033F34A40000000C0CCCC2140000000
C0CC0C4B400000004066662140000000C0CC0C4B400000004066662140000006066264B40000000E0FFFF
20400000006066264B40000000E0FFFF204000000000000404B400000004066662140000000000404B40');
ALTER TABLE ONLY germany2
    ADD CONSTRAINT germany2_pkey PRIMARY KEY (gid);
```

C Berechnen der Vegetationszeit

```
CREATE TABLE veg_per1 (

  years     float,
  next_begins       date,
  this_begins       date,
  this_ends         date,
  length    int4,
  species int4,
  lon       int4,
  lat       int4,

  PRIMARY KEY (lon, lat, species, years)

);

-- Erzeugen von Funktionen, die die Vegetationszeit berechnen
-- Hauptfunktion ist dabei die Funktion veggi, welche keine Werte zurückgibt,
-- aber letztlich die Tabelle veg_per auffüllt

---------- verwendete funktionen----------
-- Zählen der höchsten und niedrigsten Koordinaten_ID / für schleife über koordinaten
CREATE OR REPLACE FUNCTION highest_lon() RETURNS int4 AS $$
  SELECT DISTINCT(MAX(lon)) FROM coordinates;
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION highest_lat() RETURNS int4 AS $$
  SELECT DISTINCT(MAX(lat)) FROM coordinates;
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION lowest_lon() RETURNS int4 AS $$
  SELECT DISTINCT(MIN(lon)) FROM coordinates;
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION lowest_lat() RETURNS int4 AS $$
  SELECT DISTINCT(MIN(lat)) FROM coordinates;
$$ LANGUAGE 'sql';

-- Zählen des niedrigsten und höchsten Jahres innerhalb eines Scenarios / innerhalb erster WHILE NOT
schleife und loop über die Jahre
CREATE OR REPLACE FUNCTION lowest_year() RETURNS float AS $$
  SELECT DISTINCT(MIN(EXTRACT(YEAR FROM time))) FROM temperature1;
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION highest_year() RETURNS float AS $$
  SELECT DISTINCT(MAX(EXTRACT(YEAR FROM time))) FROM temperature1;
$$ LANGUAGE 'sql';

-- Zählen der niedrigsten und höchsten Menzel_ID / für Schleife über Baumarten
CREATE OR REPLACE FUNCTION lowest_menzel() RETURNS int4 AS $$
  SELECT MIN(menzel_id) FROM menzel;
$$ LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION highest_menzel() RETURNS int4 AS $$
  SELECT MAX(menzel_id) FROM menzel;
$$ LANGUAGE 'sql';
```

```
-- DOY des 05.10. jeden Jahres für jede Koordinate
CREATE OR REPLACE FUNCTION DayOfYear_oct_five(int4, int4, int4) RETURNS float AS $$

  SELECT DISTINCT(EXTRACT (DOY FROM time))  FROM temperature1
            WHERE lon = $1
            AND lat = $2
            AND jahr = $3
            AND monat = 10
            AND tage = 05;

$$ LANGUAGE 'sql';

-- DOY des 01.08. jeden Jahres für jede Koordinate
CREATE OR REPLACE FUNCTION DayOfYear_aug_first(int4, int4, int4) RETURNS float AS $$

  SELECT DISTINCT(EXTRACT (DOY FROM time)) FROM temperature1
            WHERE lon = $1
            AND lat = $2
            AND jahr = $3
            AND monat = 08
            AND tage = 01;

$$ LANGUAGE 'sql';

-- Ausgeben des Datums
CREATE OR REPLACE FUNCTION time_for_tag(int4, int4, int4, int4) RETURNS date AS $$

  SELECT time+$3 FROM temperature1
            WHERE lon = $1
                  AND lat = $2
                  AND jahr = $4+1
                  AND monat = 02
                  AND tage = 01;

$$ LANGUAGE 'sql';

-- Auswahl der Tagesmitteltemperaturen nach 31.10
CREATE OR REPLACE FUNCTION daily_mean(int4, int4, date) RETURNS real AS $$
  SELECT temp FROM temperature1
            WHERE lon = $1
            AND lat = $2
            AND time = $3;
$$ LANGUAGE 'sql';

-- daily mean summe für gleitendes mittel
CREATE OR REPLACE FUNCTION daily_mean_sum(int4, int4, date) RETURNS real AS $$
  SELECT sum(temp) FROM temperature1
            WHERE lon = $1
            AND lat = $2
            AND time between $3 AND $3+6;
$$ LANGUAGE 'sql';

-- funktion, um auf array menzel zuzugreifen
CREATE OR REPLACE FUNCTION menzelpara(int4, int4) RETURNS float AS $$
  SELECT parms[$1] FROM menzel WHERE menzel_id = $2;
$$ LANGUAGE 'sql';
```

```
-- Auswahl des Datums am 5. 10. jeden Jahres
CREATE OR REPLACE FUNCTION time_for_five_oct(int4, int4, int4) RETURNS date AS $$

  SELECT time FROM temperature1
        WHERE lon = $1
              AND lat = $2
              AND jahr = $3
              AND monat = 10
              AND tage = 05;

$$ LANGUAGE 'sql';

-- Auswahl des Datums an jedem Tag, benötigt zum Abgleich mit time_for_five_oct
CREATE OR REPLACE FUNCTION date_oct(date) RETURNS date AS $$
  SELECT time FROM temperature1 WHERE time = $1;
$$ LANGUAGE 'sql';

-- Summierung der Kältetage im November, Dezember und Januar
CREATE OR REPLACE FUNCTION count_colddays(int4, int4, int4, float4) RETURNS int8 AS $$

  SELECT count(time) FROM temperature1
        WHERE lon = $1
        AND lat = $2
        AND ((jahr = $3 AND (monat = 11 OR monat = 12)) OR (jahr = $3+1 AND monat = 1))
        AND temp <= $4;

$$ LANGUAGE 'sql';

-- Erzeugen von Indices zur Leistungsverbesserung
CREATE INDEX co_jahr_monat_tage_index1 ON temperature1(lon, lat, jahr, monat, tage);
CREATE INDEX co_time_index1 ON temperature1(lon, lat, time);

-- Funktion zur Berechnung der Vegetationsperiode
CREATE OR REPLACE FUNCTION veggi() RETURNS void AS $$
  DECLARE
  hsum real;                    Temperatursumme ab dem 01. Februar      tscrit real;
Kritische Temperatursumme ab dem 01. Februar
  tag int4;                     durchzählen der Tage ab dem 01. Februar
  colddays int8;                Anzahl der Kältetage 01.11 – Beginn Vegetationsperiode
  tag_veg_begin int4;           Tag des Jahres, an dem die Vegetationsperiode beginnt
  cgm7 real;                    Zähler für das Auftreten eines gleitenden Mittels unter-
                                    halb des Schwellenwertes
  ts real;                      Temperatursumme einer 7-Tagesperiode (zur Berech-
                                    nung des gleitenden Mittels)
  gmt7 real;                    Temperaturmittelwert einer 7-Tagesperiode (ts/7)
  tag_veg_ende int4;            Tag des Jahres, an dem die Vegetationsperiode endet
  length_veg_per int4;          Länge der Vegetationsperiode
  datum date;                   Datum von Variable tag
  five_oct date;                Datum des 05. Oktobers
  date_veg_begin date;          Datum, an dem die Vegetationsperiode beginnt
  date_veg_ende date;           Datum, an dem die Vegetationsperiode endet
  menzel_temp_coldday float4;   Temperaturschwelle für die Kältetage
```

```
BEGIN
FOR lons IN lowest_lon()..highest_lon() BY 2 LOOP

        BEGIN
        FOR lats IN lowest_lat()..highest_lat() BY 2 LOOP

                BEGIN
                FOR menzel_id IN lowest_menzel()..highest_menzel() LOOP

                        BEGIN
                        FOR year IN 1969..2099 /*lowest_year()..(highest_year()-1)*/ LOOP

                                menzel_temp_coldday := menzelpara(1, menzel_id);
                                colddays := count_colddays(lons, lats, year,
                                                           menzel_temp_coldday);
                                hsum := 0.0;
                                tscrit := 1.0;
                                tag := 0;

                                BEGIN
                                WHILE NOT hsum >= tscrit LOOP

                                        datum:=time_for_tag(lons, lats, tag, year);

                                        IF daily_mean(lons, lats, datum) <=
                                            menzelpara(1,menzel_id)

                                                THEN colddays := colddays+1;

                                        END IF;

                                        tscrit := menzelpara(3, menzel_id) +
                                                  menzelpara(4, menzel_id)*ln(colddays);

                                        IF daily_mean(lons, lats, datum) >
                                                        menzelpara(2, menzel_id)


                                                THEN hsum := hsum +
                                                             daily_mean(lons, lats, datum)-
                                                             menzelpara(2, menzel_id);

                                        END IF;


                                        tag := tag+1;
                                        datum:=time_for_tag(lons, lats, tag, year);

                                END LOOP;
-- Baum weiß erst im Laufe des Tages, ob die kritische Temperatursumme erreicht wird. D.h. "heute" zählt trotz erreichen der
--kritischen Temperatursumme noch nicht zur Vegetationszeit. Am ende des Schleifendurchlaufs wird tag auf nächsten Tag erhöht, wovon dann
-- auch date_veg_beginn berechnet wird. Date_veg_begin zählt zur Vegetationszeit dazu
--Fehler: bei mir kanns nie am 1. Feb losgehen

                                tag_veg_begin := tag +31;-- hier reicht 31, weil in der schleife
                                                            der Tag ja noch erhöht wird
```

```
                                        date_veg_begin := time_for_tag(lons, lats, tag, year);

                                        UPDATE veg_per1
                                                SET next_begins = date_veg_begin where years =
                                                                year AND lon = lons AND lat = lats
                                                                AND species= menzel_id;

                                        tag := DayOfYear_aug_first(lons, lats, year+1)-32;
-- DayOfYear_aug_first berechnet den DOY. übersetzt in Tag muss natürlich Januar und erster Feb abgezo-
gen werden
                                        cgm7 := 0.0;
                                        gmt7 := 0.0;
                                        datum:=time_for_tag(lons, lats, tag, year);

                                        WHILE NOT cgm7 > 4.0 LOOP

                                                ts := daily_mean_sum(lons, lats, datum);

                                                gmt7 := ts / 7.0;

                                                IF gmt7 < menzelpara(7, menzel_id)

                                                        THEN cgm7 := cgm7 + 1.0;

                                                END IF;

                                                tag := tag + 1;
                                                datum:=time_for_tag(lons, lats, tag, year);

                                                IF EXTRACT(MONTH FROM datum) = 09 AND
                                                     EXTRACT(DAY FROM datum) = 30

                                                        THEN cgm7 := 5.0;

                                                END IF;
                                        END LOOP;

                                        tag := tag+5; -- + 5 um Ende des 7-tägigen Mittels zu erreichen
                                        five_oct:=time_for_five_oct(lons, lats, year+1);

--                                      IF datum+5 >= date_oct(five_oct)

--                                              THEN     tag := DayOfYear_oct_five(lons,
                                                                                lats, year + 1)-32;
                                                  -- 32 weil 1.Februar 0 ist und in der Zählung fehlt

--                                      END IF;
-- Date_veg_ende gibt das letzte zur Vegetationsperiode zugehörige Datum an

                                        tag_veg_ende := tag + 32;
                                        date_veg_ende := time_for_tag(lons, lats, tag, year);
                                        length_veg_per := tag_veg_ende - tag_veg_begin; -- rechnet
richtig bsp:beginn 11.8. ende 12.08. dann länge = 2

                                        INSERT INTO veg_per1(years, this_begins, this_ends, length,
                                                        species, lon, lat)
                                                VALUES(year+1, date_veg_begin, date_veg_ende,
                                                        length_veg_per, menzel_id, lons, lats);
```

```
                                        END;
                                    END LOOP;
                                    END;
                                END LOOP;
                                END;
                            END LOOP;
                            END;
                        END LOOP;
                        END;

$$ LANGUAGE 'plpgsql';

SELECT veggi();
```

D Aggregierung zu Klimavariablen

```
-- Erstellen einer diagramm_tabelle für die VP
-- Temperatur
CREATE TABLE diagramm_veg_temp1 AS
SELECT avg(temp) AS mean_temp_vp, b.lon, b.lat, b.species, b.years::int4
  FROM temperature1 a, veg_per1 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND a.jahr = b.years
  AND time between this_begins AND this_ends
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_veg_temp1
  ADD PRIMARY KEY(lon, lat, species, years);

CREATE TABLE diagramm_veg_temp2 AS
SELECT avg(temp) AS mean_temp_vp, b.lon, b.lat, b.species, b.years::int4
  FROM temperature2 a, veg_per2 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND a.jahr = b.years
  AND time between this_begins AND this_ends
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_veg_temp2
  ADD PRIMARY KEY(lon, lat, species, years);

-- Niederschlag
CREATE TABLE diagramm_veg_prec1 AS
SELECT sum("PRECIP_TOT") AS sum_prec_vp, b.lon, b.lat, b.species, b.years::int4
  FROM precipitation1 a, veg_per1 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND EXTRACT(YEAR FROM time) = b.years
  AND time between this_begins AND this_ends
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_veg_prec1
  ADD PRIMARY KEY(lon, lat, species, years);

CREATE TABLE diagramm_veg_prec2 AS
SELECT sum("PRECIP_TOT") AS sum_prec_vp, b.lon, b.lat, b.species, b.years::int4
  FROM precipitation2 a, veg_per2 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND EXTRACT(YEAR FROM time) = b.years
  AND time between this_begins AND this_ends
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_veg_prec2
  ADD PRIMARY KEY(lon, lat, species, years);
```

```
-- Erstellen der diagramm_tabellen außerhalb der VP
-- Temperatur
CREATE TABLE diagramm_nveg_temp1  AS
SELECT avg(temp) AS mean_temp_nvp, b.lon, b.lat, b.species, (b.years+1)::int4 AS years
  FROM temperature1 a, veg_per1 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND (a.jahr = b.years OR a.jahr-1 = b.years)
  AND time > this_ends AND time < next_begins
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_nveg_temp1
  ADD PRIMARY KEY(lon, lat, species, years);

CREATE TABLE diagramm_nveg_temp2 AS
SELECT avg(temp) AS mean_temp_nvp, b.lon, b.lat, b.species, (b.years+1)::int4 AS years
  FROM temperature2 a, veg_per2 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND (a.jahr = b.years OR a.jahr-1 = b.years)
  AND time > this_ends AND time < next_begins
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_nveg_temp2
  ADD PRIMARY KEY(lon, lat, species, years);


-- Niederschlag ------------------------------------------------------------------------

CREATE TABLE diagramm_nveg_prec1 AS
SELECT sum("PRECIP_TOT") AS sum_prec_nvp, b.lon, b.lat, b.species, (b.years+1)::int4 AS years
  FROM precipitation1 a, veg_per1 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND (EXTRACT(YEAR FROM time) = b.years OR EXTRACT(YEAR FROM time)-1 = b.years)
  AND time > this_ends AND time < next_begins
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_nveg_prec1
  ADD PRIMARY KEY(lon, lat, species, years);


CREATE TABLE diagramm_nveg_prec2 AS
SELECT sum("PRECIP_TOT") AS sum_prec_nvp, b.lon, b.lat, b.species, (b.years+1)::int4 AS years
  FROM precipitation2 a, veg_per2 b
  WHERE a.lon = b.lon
  AND a.lat = b.lat
  AND (EXTRACT(YEAR FROM time) = b.years OR EXTRACT(YEAR FROM time)-1 = b.years)
  AND time > this_ends AND time < next_begins
  GROUP BY b.lon, b.lat, b.species, b.years;

ALTER TABLE diagramm_nveg_prec2
  ADD PRIMARY KEY(lon, lat, species, years);
```

```
-- Zusammenfassen der einzelnen diagram-Tabellen zu einer einzigen
CREATE TABLE diagramme_a1b AS

SELECT a.mean_temp_vp AS temp_vp_run1, b.mean_temp_vp AS temp_vp_run2, c.sum_prec_vp AS prec_vp_run1, d.sum_prec_vp AS prec_vp_run2,
  e.mean_temp_nvp AS temp_nvp_run1, f.mean_temp_nvp AS temp_nvp_run2, g.sum_prec_nvp AS prec_nvp_run1, h.sum_prec_nvp AS prec_nvp_run2,
  i.length AS length_run1, j.length AS length_run2,
  a.lon, a.lat, a.species, a.years
FROM diagramm_veg_temp1 a, diagramm_veg_temp2 b, diagramm_veg_prec1 c, diagramm_veg_prec2 d,
  diagramm_nveg_temp1 e, diagramm_nveg_temp2 f, diagramm_nveg_prec1 g, diagramm_nveg_prec2 h,
  veg_per1 i, veg_per2 j
WHERE    (a.lon = b.lon
  AND a.lat = b.lat
  AND a.species = b.species
  AND a.years = b.years)
AND        (a.lon = c.lon
  AND a.lat = c.lat
  AND a.species = c.species
  AND a.years = c.years)
AND        (a.lon = d.lon
  AND a.lat = d.lat
  AND a.species = d.species
  AND a.years = d.years)
AND        (a.lon = e.lon
  AND a.lat = e.lat
  AND a.species = e.species
  AND a.years = e.years)
AND        (a.lon = f.lon
  AND a.lat = f.lat
  AND a.species = f.species
  AND a.years = f.years)
AND        (a.lon = g.lon
  AND a.lat = g.lat
  AND a.species = g.species
  AND a.years = g.years)
AND        (a.lon = h.lon
  AND a.lat = h.lat
  AND a.species = h.species
  AND a.years = h.years)
AND        (a.lon = i.lon
  AND a.lat = i.lat
  AND a.species = i.species
  AND a.years = i.years)
AND        (a.lon = j.lon
  AND a.lat = j.lat
  AND a.species = j.species
  AND a.years = j.years);

ALTER TABLE diagramme_a1b
  ADD PRIMARY KEY(lon, lat, species, years);
```

```
-----------------------------------------------------------------------------------------------------
-- Erzeugen einer Tabelle, die für jedes Jahr die Klimaperiode angibt

CREATE TABLE period AS SELECT DISTINCT(years) FROM diagramme;

ALTER TABLE period
  ADD COLUMN period int2,
  ADD PRIMARY KEY (years);

UPDATE period
  SET       period = 1 WHERE years BETWEEN 1971 AND 2000;

UPDATE period
  SET       period = 2 WHERE years BETWEEN 2011 AND 2040;

UPDATE period
  SET       period = 3 WHERE years BETWEEN 2041 AND 2070;

UPDATE period
  SET       period = 4 WHERE years BETWEEN 2071 AND 2100;

-----------------------------------------------------------------------------------------------------
-- Aggregieren der Jahreswerte zu Klimaperioden

CREATE TABLE klimaperioden_a1b AS

SELECT avg(temp_vp_run1) AS temp_vp_run1, avg(temp_vp_run2) AS temp_vp_run2, avg(prec_vp_run1)
AS prec_vp_run1, avg(prec_vp_run2) AS prec_vp_run2,
   avg(temp_nvp_run1) AS temp_nvp_run1, avg(temp_nvp_run2) AS temp_nvp_run2, avg(prec_nvp_run1)
AS prec_nvp_run1, avg(prec_nvp_run2) AS prec_nvp_run2,
  avg(length_run1) AS length_run1, avg(length_run2) AS length_run2,
  lon, lat, species, period
FROM diagramme_a1b, period
WHERE     diagramme_a1b.years = period.years
  AND period IS NOT NULL
GROUP BY lon, lat, species, period;

ALTER TABLE klimaperioden_a1b
ADD PRIMARY KEY(lon, lat, species, period);


-- notwendig beim setzen der Nullwerte
SELECT AddGeometryColumn('klimaperioden_a1b', 'coords', 4326, 'POINT', 2);

UPDATE klimaperioden_a1b
  SET coords = (SELECT coord FROM coordinates WHERE coordinates.lat = klimaperioden_a1b.lat AND co-
ordinates.lon = klimaperioden_a1b.lon);
```

```
-- Setzen von Nullwerten
UPDATE klimaperioden_a1b
  SET prec_vp_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);-- muss 0 sein und nicht NULL, sonst funktioniert es mit gdal_grid nicht
UPDATE klimaperioden_a1b
  SET prec_vp_run2 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET prec_nvp_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET prec_nvp_run2 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET temp_vp_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET temp_vp_run2 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET temp_nvp_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET temp_nvp_run2 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET length_run1 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);
UPDATE klimaperioden_a1b
  SET length_run2 = 0.0 FROM  germany2 WHERE ST_DISJOINT(coords, the_geom);


--Loeschen aller Zwischentabellen, die in B1 wiederverwendet werden
DROP TABLE temperature1;
DROP TABLE temperature2;
DROP VIEW precipitation1;
DROP VIEW precipitation2;
DROP TABLE veg_per1;
DROP TABLE veg_per2;
DROP TABLE diagramm_veg_temp1;
DROP TABLE diagramm_veg_temp2;
DROP TABLE diagramm_nveg_temp1;
DROP TABLE diagramm_nveg_temp2;
DROP TABLE diagramm_veg_prec1;
DROP TABLE diagramm_veg_prec2;
DROP TABLE diagramm_nveg_prec1;
DROP TABLE diagramm_nveg_prec2;
```

E Umstrukturierung der Ergebnisse

```
-- sum_prec_vp_run1
CREATE VIEW a1b_prec_vp_per1 AS
  SELECT avg((prec_vp_run1+prec_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 1
  GROUP BY coords;

CREATE VIEW a1b_prec_vp_per2 AS
  SELECT avg((prec_vp_run1+prec_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 2
  GROUP BY coords;

CREATE VIEW a1b_prec_vp_per3 AS
  SELECT avg((prec_vp_run1+prec_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 3
  GROUP BY coords;

CREATE VIEW a1b_prec_vp_per4 AS
  SELECT avg((prec_vp_run1+prec_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 4
  GROUP BY coords;

-- sum_prec_nvp_run1
CREATE VIEW a1b_prec_nvp_per1 AS
  SELECT avg((prec_nvp_run1+prec_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 1
  GROUP BY coords;

CREATE VIEW a1b_prec_nvp_per2 AS
  SELECT avg((prec_nvp_run1+prec_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 2
  GROUP BY coords;

CREATE VIEW a1b_prec_nvp_per3 AS
  SELECT avg((prec_nvp_run1+prec_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 3
  GROUP BY coords;

CREATE VIEW a1b_prec_nvp_per4 AS
  SELECT avg((prec_nvp_run1+prec_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 4
  GROUP BY coords;
```

```
-- mean_temp_vp_run1
CREATE VIEW a1b_temp_vp_per1 AS
  SELECT avg((temp_vp_run1+temp_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 1
  GROUP BY coords;

CREATE VIEW a1b_temp_vp_per2 AS
  SELECT avg((temp_vp_run1+temp_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 2
  GROUP BY coords;

CREATE VIEW a1b_temp_vp_per3 AS
  SELECT avg((temp_vp_run1+temp_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 3
  GROUP BY coords;

CREATE VIEW a1b_temp_vp_per4 AS
  SELECT avg((temp_vp_run1+temp_vp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 4
  GROUP BY coords;

-- mean_temp_nvp_run1
CREATE VIEW a1b_temp_nvp_per1 AS
  SELECT avg((temp_nvp_run1+temp_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 1
  GROUP BY coords;

CREATE VIEW a1b_temp_nvp_per2 AS
  SELECT avg((temp_nvp_run1+temp_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 2
  GROUP BY coords;

CREATE VIEW a1b_temp_nvp_per3 AS
  SELECT avg((temp_nvp_run1+temp_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 3
  GROUP BY coords;

CREATE VIEW a1b_temp_nvp_per4 AS
  SELECT avg((temp_nvp_run1+temp_nvp_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 4
  GROUP BY coords;

-- laenge vegetationsperiode
CREATE VIEW a1b_length_vp_per1 AS
  SELECT avg((length_run1+length_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 1
  GROUP BY coords;
```

```
CREATE VIEW a1b_length_vp_per2 AS
  SELECT avg((length_run1+length_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 2
  GROUP BY coords;

CREATE VIEW a1b_length_vp_per3 AS
  SELECT avg((length_run1+length_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 3
  GROUP BY coords;

CREATE VIEW a1b_length_vp_per4 AS
  SELECT avg((length_run1+length_run2)/2), coords
  FROM klimaperioden_a1b
  WHERE period = 4
  GROUP BY coords;

CREATE TABLE diagramme AS
SELECT a.temp_vp_run1 AS temp_vp_run1_a1b, a.temp_vp_run2 AS temp_vp_run2_a1b,
a.temp_nvp_run1 AS temp_nvp_run1_a1b, a.temp_nvp_run2 AS temp_nvp_run2_a1b,
  b.temp_vp_run1 AS temp_vp_run1_b1, b.temp_vp_run2 AS temp_vp_run2_b1, b.temp_nvp_run1 AS
temp_nvp_run1_b1, b.temp_nvp_run2 AS temp_nvp_run2_b1,
  a.prec_vp_run1 AS prec_vp_run1_a1b, a.prec_vp_run2 AS prec_vp_run2_a1b, a.prec_nvp_run1 AS
prec_nvp_run1_a1b, a.prec_nvp_run2 AS prec_nvp_run2_a1b,
  b.prec_vp_run1 AS prec_vp_run1_b1, b.prec_vp_run2 AS prec_vp_run2_b1, b.prec_nvp_run1 AS
prec_nvp_run1_b1, b.prec_nvp_run2 AS prec_nvp_run2_b1,
  a.length_run1 AS length_run1_a1b, a.length_run2 AS length_run2_a1b, b.length_run1 AS
length_run1_b1, b.length_run2 AS length_run2_b1,
  a.lon, a.lat, a.species, a.years

FROM diagramme_a1b a, diagramme_b1 b
WHERE a.lon=b.lon
AND a.lat=b.lat
AND a.species=b.species
AND a.years=b.years;
```

F Auszug aus dem Mapfile

```
MAP

        NAME "klima"
        IMAGETYPE gif
        EXTENT 1045000 5914600 1778900 7578400
        SIZE 500 500
        SHAPEPATH "testMap"
        IMAGECOLOR 255 255 255
        PROJECTION
               "init=epsg:900913"
        END
        STATUS ON
        TRANSPARENT ON
        FONTSET '.\fonts\fonts.list'


        LEGEND
               STATUS ON
               KEYSIZE 22 22
               IMAGECOLOR 238 238 224

               LABEL
                      TYPE truetype
                      SIZE 11
                      COLOR 1 1 1
                      FONT arial
                      OFFSET 0 -5
               END
        END

         WEB
               IMAGEPATH 'C:\ms4w\tmp\ms_tmp\'
               IMAGEURL '.\'

               METADATA
                'wms_title' 'klima'
                'wms_onlineresource' 'http://localhost/cgi-bin/mapserv.exe?map=C:\Dokumente
                                      und Einstellungen\sabine\Desktop\testMap\testMap.map'
                'wms_srs' 'EPSG:900913'
              END
         END

         LAYER
              NAME "temp_vp_1_a1b"
              DATA '..\geotiffs\a1b_google\temp_vp_per1_google.tiff'
              TYPE RASTER
               STATUS ON
              METADATA
                'wms_title' 'Tagesmitteltemperatur in der Vegetationszeit (Klimaperiode 1)'
                'wms_srs' 'EPSG:900913'
              END
              PROJECTION
                 "init=epsg:900913"
              END

              CLASS
#               NAME "0"
                EXPRESSION ([pixel] < 0.001)
                STYLE
                    COLOR 255 255 255
                END
              END

              CLASS
                NAME "< 5 °C"
                EXPRESSION ([pixel] >=0.001 AND [pixel] < 5 )
                STYLE
                    COLOR 0 0 102
                END
              END

              CLASS
                NAME " 5 - 6 °C"
                EXPRESSION ([pixel]>=5 AND [pixel] < 6)
                STYLE
```

```
      COLOR 0 0 204
  END
 END

CLASS
  NAME " 6 - 7 °C"
  EXPRESSION ([pixel]>=6 AND [pixel] < 7)
  STYLE
      COLOR 51 51 255
  END
 END

CLASS
  NAME " 7 - 8 °C"
  EXPRESSION ([pixel]>=7 AND [pixel] < 8)
  STYLE
      COLOR 51 102 255
  END
 END

CLASS
  NAME " 8 - 9 °C"
  EXPRESSION ([pixel]>=8 AND [pixel] < 9)
  STYLE
      COLOR 122 153 255
  END
 END

CLASS
  NAME " 9 -  10 °C"
  EXPRESSION ([pixel]>=9 AND [pixel] < 10)
  STYLE
      COLOR 183 200 255
  END
 END

CLASS
  NAME " 10 - 11 °C"
  EXPRESSION ([pixel]>=10 AND [pixel] < 11)
  STYLE
      COLOR 220 240 255
  END
 END

CLASS
  NAME " 11 - 12 °C"
  EXPRESSION ([pixel]>=11 AND [pixel] < 12)
  STYLE
      COLOR 240 255 240
  END
 END

CLASS
  NAME " 12 - 13 °C"
  EXPRESSION ([pixel]>=12 AND [pixel] < 13)
  STYLE
      COLOR 255 255 204
  END
 END

CLASS
  NAME " 13 - 14 °C"
  EXPRESSION ([pixel]>=13 AND [pixel] < 14)
  STYLE
      COLOR 255 255 152
  END
 END

CLASS
  NAME " 14 - 15 °C"
  EXPRESSION ([pixel]>=14 AND [pixel] < 15)
  STYLE
      COLOR 255 255 100
  END
 END

CLASS
  NAME " 15 - 16 °C"
```

```
        EXPRESSION ([pixel]>=15 AND [pixel] < 16)
        STYLE
            COLOR 255 255 0
        END
      END

      CLASS
        NAME " 16 - 17 °C"
        EXPRESSION ([pixel]>=16 AND [pixel] < 17)
        STYLE
            COLOR 255 180 0
        END
      END

      CLASS
        NAME " 17 - 18 °C"
        EXPRESSION ([pixel]>=17 AND [pixel] < 18)
        STYLE
            COLOR 255 100 0
        END
      END

      CLASS
        NAME " 18 - 19 °C"
        EXPRESSION ([pixel]>=18 AND [pixel] < 19)
        STYLE
            COLOR 204 0 0
        END
      END

      CLASS
        NAME " 19 - 20 °C"
        EXPRESSION ([pixel]>=19 AND [pixel] < 20)
        STYLE
            COLOR 150 0 0
        END
      END

      CLASS
        NAME " 20 - 21 °C"
        EXPRESSION ([pixel]>=20 AND [pixel] < 21)
        STYLE
            COLOR 100 0 0
        END
      END

      CLASS
        NAME " 21 - 22 °C"
        EXPRESSION ([pixel]>=21 AND [pixel] < 22)
        STYLE
            COLOR 70 0 70
        END
      END

      CLASS
        NAME "> 22 °C"
        EXPRESSION ([pixel]>=22)
        STYLE
            COLOR 100 0 100
        END
      END
END

LAYER
      NAME "prec_vp_1_a1b"
      DATA '..\geotiffs\a1b_google\prec_vp_per1_google.tiff'
      TYPE RASTER
       STATUS ON
      METADATA
        'wms_title' 'Niederschlagssumme in der Vegetationszeit (Klimaperiode 1),
                     Szenario A1B'
        'wms_srs' 'EPSG:900913'
      END
      PROJECTION
         "init=epsg:900913"
      END

      CLASS
```

```
#               NAME "< 100 mm"
                EXPRESSION ([pixel] <100)
                STYLE
                    COLOR 255 255 255
                END
              END

              CLASS
                NAME " 100 - 200 mm"
                EXPRESSION ([pixel]>=100 AND [pixel] <200)
                STYLE
                    COLOR 245 215 210
                END
              END

              CLASS
                NAME " 200 - 300 mm"
                EXPRESSION ([pixel]>=200 AND [pixel] <300)
                STYLE
                    COLOR 255 245 190
                END
              END

              CLASS
                NAME " 300 - 400 mm"
                EXPRESSION ([pixel]>=300 AND [pixel] <400)
                STYLE
                    COLOR 220 245 170
                END
              END

              CLASS
                NAME " 400 - 500 mm"
                EXPRESSION ([pixel]>=400 AND [pixel] <500)
                STYLE
                    COLOR 180 250 200
                END
              END

              CLASS
                NAME " 500 - 600 mm"
                EXPRESSION ([pixel]>=500 AND [pixel] <600)
                STYLE
                    COLOR 120 235 190
                END
              END

              CLASS
                NAME " 600 - 700 mm"
                EXPRESSION ([pixel]>=600 AND [pixel] <700)
                STYLE
                    COLOR  110 180 250
                END
              END

              CLASS
                NAME " 700 - 800 mm"
                EXPRESSION ([pixel]>=700 AND [pixel] <800)
                STYLE
                    COLOR 90 90 250
                END
              END

              CLASS
                NAME " 800 - 900 mm"
                EXPRESSION ([pixel]>=800 AND [pixel] <900)
                STYLE
                    COLOR 40 40 205
                END
              END


              CLASS
                NAME " > 900 mm"
                EXPRESSION ([pixel]>=900)
                STYLE
                    COLOR 0 0 100
                END
```

```
      END
END

LAYER
      NAME "length_vp_1_a1b"
      DATA '..\geotiffs\a1b_google\length_vp_per1_google.tiff'
      TYPE RASTER
       STATUS ON
      METADATA
        'wms_title' 'Länge der Vegetationszeit (Klimaperiode 1), Szenario A1B'
        'wms_srs' 'EPSG:900913'
      END
      PROJECTION
         "init=epsg:900913"
      END

      CLASS
#       NAME "0 "
        EXPRESSION ([pixel] <1)
        STYLE
            COLOR 255 255 255
        END
      END

      CLASS
        NAME " < 100 Tage"
        EXPRESSION ([pixel] <100)
        STYLE
            COLOR 245 215 210
        END
      END

      CLASS
        NAME " 100 - 110 Tage"
        EXPRESSION ([pixel]>=100 AND [pixel] <110)
        STYLE
            COLOR 255 245 190
        END
      END

      CLASS
        NAME " 110 - 120 Tage"
        EXPRESSION ([pixel]>=110 AND [pixel] <120)
        STYLE
            COLOR 220 245 170
        END
      END

      CLASS
        NAME " 120 - 130 Tage"
        EXPRESSION ([pixel]>=120 AND [pixel] <130)
        STYLE
            COLOR 180 250 200
        END
      END

      CLASS
        NAME " 130 - 140 Tage"
        EXPRESSION ([pixel]>=130 AND [pixel] <140)
        STYLE
            COLOR 120 235 190
        END
      END

      CLASS
        NAME " 140 - 150 Tage"
        EXPRESSION ([pixel]>=140 AND [pixel] <150)
        STYLE
            COLOR  110 180 250
        END
      END

      CLASS
        NAME " 150 - 160 Tage"
        EXPRESSION ([pixel]>=150 AND [pixel] <160)
        STYLE
            COLOR 90 90 250
        END
```

```
            END

            CLASS
              NAME " 160 - 170 Tage"
              EXPRESSION ([pixel]>=160 AND [pixel] <170)
              STYLE
                  COLOR 40 40 205
              END
            END


            CLASS
              NAME " > 170 Tage"
              EXPRESSION ([pixel]>=900)
              STYLE
                  COLOR 0 0 100
              END
             END
        END


END
```

G Entfernung des Layernamens

Die Entfernung des Layernamens beim Erzeugen der Legende wurde im Source Code vorgenommen. In der Datei „C:\ms4w\tmp\GeoExt\lib\GeoExt\widgets\LegendPanel.js“ wurde in Zeile 112 folgendes eingefügt: *title=""";* Dadurch wird der Titel durch einen leeren Wert überschrieben

H JavaScript-Code für die Web-Mapping Anwendung

```
    Ext.onReady(function() {

 var map,
     temp_vz_1_c20, temp_vz_2_a1b, temp_vz_3_a1b, temp_vz_4_a1b,
     temp_nvz_1_c20, temp_nvz_2_a1b, temp_nvz_3_a1b, temp_nvz_4_a1b,
     prec_vz_1_c20, prec_vz_2_a1b, prec_vz_3_a1b, prec_vz_4_a1b,
     prec_nvz_1_c20, prec_nvz_2_a1b, prec_nvz_3_a1b, prec_nvz_4_a1b,
     length_vz_1_c20, length_vz_2_a1b, length_vz_3_a1b, length_vz_4_a1b,
     temp_vz_2_b1, temp_vz_3_b1, temp_vz_4_b1,
     temp_nvz_2_b1, temp_nvz_3_b1, temp_nvz_4_b1,
     prec_vz_2_b1, prec_vz_3_b1, prec_vz_4_b1,
     prec_nvz_2_b1, prec_nvz_3_b1, prec_nvz_4_b1,
     length_vz_2_b1, length_vz_3_b1, length_vz_4_b1,
     googlayer;

            map = new OpenLayers.Map( {
                projection: "EPSG:900913",//new OpenLayers.Projection("EPSG:900913"),
                units: "m",
                maxResolution:            156543.0339,            //vorgeschlagen            unter:
http://trac.openlayers.org/wiki/SphericalMercator
                maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
                                                 20037508.34, 20037508.34),
                controls: [ new OpenLayers.Control.PanZoomBar(),
                            new OpenLayers.Control.Navigation()],
                restrictedExtent:new OpenLayers.Bounds(545000, 5914600, 1778900, 7578400)

                 } );
            // Zoom ab Deutschland bis zu einer Zelle möglich


////////////////////////////////////////////////////////////////////////////////
// layer fuer A1B

            temp_vz_1_c20 = new OpenLayers.Layer.WMS( "temp_vz_1_c20",
                    "http://localhost/cgi-bin/mapserv.exe?map=C:/Dokumente und Einstellungen/
                      sabine/Desktop/visualization/sabine_klimaszenarien.map",
                    {layers: 'temp_vp_1_c20'},
                    {opacity:0.7,
                    visibility:false,
                    singleTile:true
                    });

            temp_vz_2_a1b = new OpenLayers.Layer.WMS( "temp_vz_2_a1b",
                    "http://localhost/cgi-bin/mapserv.exe?map=C:/Dokumente und Einstellungen/
                      sabine/Desktop/visualization/sabine_klimaszenarien.map",
                    {layers: 'temp_vp_2_a1b'},
                    {transparent:true,
                    opacity:0.7,
                    visibility:false,
                    singleTile:true,

                    });

//Weitere WMS-Layer

            googlayer = new OpenLayers.Layer.Google(
                 "Google",
                 {sphericalMercator:true}
                 );

            googlayer.MIN_ZOOM_LEVEL = 6 ;
            googlayer.MAX_ZOOM_LEVEL = 12 ;

           map.addLayers( [googlayer,
                        temp_vz_1_c20, temp_vz_2_a1b, temp_vz_3_a1b, temp_vz_4_a1b,
                        temp_nvz_1_c20, temp_nvz_2_a1b, temp_nvz_3_a1b, temp_nvz_4_a1b,
                        prec_vz_1_c20, prec_vz_2_a1b, prec_vz_3_a1b, prec_vz_4_a1b,
                        prec_nvz_1_c20, prec_nvz_2_a1b, prec_nvz_3_a1b, prec_nvz_4_a1b,
                        length_vz_1_c20, length_vz_2_a1b, length_vz_3_a1b, length_vz_4_a1b,
                        temp_vz_2_b1, temp_vz_3_b1, temp_vz_4_b1,
                        temp_nvz_2_b1, temp_nvz_3_b1, temp_nvz_4_b1,
                        prec_vz_2_b1, prec_vz_3_b1, prec_vz_4_b1,
                        prec_nvz_2_b1, prec_nvz_3_b1, prec_nvz_4_b1,
                        length_vz_2_b1, length_vz_3_b1, length_vz_4_b1]);
```

```
// Erstellen des MapPanels
        var panel = new GeoExt.MapPanel({
            height: 600,
            width: 590,
            map: map,
            x:210,
            y:10
        });

// Variable für die Root des TreePanel
         var rooter = new Ext.tree.TreeNode({
             text:'test',
             leaf:false
         });
// Unterteilung in A1, B1und C20

        var c20 = new Ext.tree.TreeNode({
            text:'1971-2000',
            leaf:false
            });

        var a1b = new Ext.tree.TreeNode({
            text:'A1B',
            leaf:false
            });

        var b1 = new Ext.tree.TreeNode({
            text:'B1',
            leaf:false
            });

// erste Instanz fuer c20
// erste Instanz fuer A1B
// erste Instanz fuer B1

// zweite Instanz nach c20


        var node_temp_vz_1_c20 = new GeoExt.tree.LayerNode({
            layer:temp_vz_1_c20,
            text:'Vegetationszeit',
            checkedGroup:'checker'
            });

         var node_temp_nvz_1_c20 = new GeoExt.tree.LayerNode({
            layer:temp_nvz_1_c20,
            text:'Nicht-Vegetationszeit',
            checkedGroup:'checker'
            });

// Weitere Knoten

// zweite Instanz nach A1B
// zweite Instanz nach B1

// Layer-Knoten fuer A1B


         var node_temp_vz_2_a1b = new GeoExt.tree.LayerNode({
            layer:temp_vz_2_a1b,
            text:'2011-2040',
            checkedGroup:'checker'
            });
         var node_temp_vz_3_a1b = new GeoExt.tree.LayerNode({
            layer:temp_vz_3_a1b,
            text:'2041-2070',
            checkedGroup:'checker'
            });

// weitere Knoten fuer A1B
// Layer-Knoten fuer B1


        rooter.appendChild([c20, a1b, b1]);
        c20.appendChild([temperature_c20, precipitation_c20, vegetation_c20]);
        a1b.appendChild([temperature_a1b, precipitation_a1b, vegetation_a1b]);
        b1.appendChild([temperature_b1, precipitation_b1, vegetation_b1]);
```

```
        temperature_c20.appendChild([node_temp_vz_1_c20, node_temp_nvz_1_c20]);
        temperature_a1b.appendChild([veg_a1b, nveg_a1b]);
        temperature_b1.appendChild([veg_b1, nveg_b1]);
        veg_a1b.appendChild([node_temp_vz_2_a1b, node_temp_vz_3_a1b, node_temp_vz_4_a1b]);
        nveg_a1b.appendChild([node_temp_nvz_2_a1b, node_temp_nvz_3_a1b, node_temp_nvz_4_a1b]);
        veg_b1.appendChild([node_temp_vz_2_b1, node_temp_vz_3_b1, node_temp_vz_4_b1]);
        nveg_b1.appendChild([node_temp_nvz_2_b1, node_temp_nvz_3_b1, node_temp_nvz_4_b1]);
        precipitation_c20.appendChild([node_prec_vz_1_c20, node_prec_nvz_1_c20]);
        precipitation_a1b.appendChild([veg2_a1b,nveg2_a1b]);
        precipitation_b1.appendChild([veg2_b1,nveg2_b1]);
        veg2_a1b.appendChild([node_prec_vz_2_a1b, node_prec_vz_3_a1b, node_prec_vz_4_a1b]);
        nveg2_a1b.appendChild([node_prec_nvz_2_a1b, node_prec_nvz_3_a1b, node_prec_nvz_4_a1b]);
        veg2_b1.appendChild([node_prec_vz_2_b1, node_prec_vz_3_b1, node_prec_vz_4_b1]);
        nveg2_b1.appendChild([node_prec_nvz_2_b1, node_prec_nvz_3_b1, node_prec_nvz_4_b1]);
        vegetation_c20.appendChild([node_length_vz_1_c20]);
        vegetation_a1b.appendChild([node_length_vz_2_a1b, node_length_vz_3_a1b,
                                    node_length_vz_4_a1b]);
        vegetation_b1.appendChild([node_length_vz_2_b1, node_length_vz_3_b1,
                                    node_length_vz_4_b1]);


        var LayerTree = new Ext.tree.TreePanel({
            title:'Layer',
            width:190,
            height:600,
            root:rooter,
            rootVisible:false,
            autoScroll:true,
            x:10,
            y:10,
            border:false,
            bodyStyle:{"backgroundColor":"#EEEEE0"}
        });

 var legend = new GeoExt.LegendPanel({
          x:810,
          y:10,
          width:150,
          height:600,
          bodyStyle:{"backgroundColor":"#EEEEE0"},
          border:false,
          title:'Legende'
          });

new Ext.Panel({
          items: [LayerTree, panel, legend],
          layout:'absolute',
          renderTo:leg,
          height:620,
          width:970,
          border:false,
          bodyStyle:{"backgroundColor":"#545454"},
          autoScroll:true
          });



});
```

I Code für die Benutzeroberfläche der Liniendiagramme

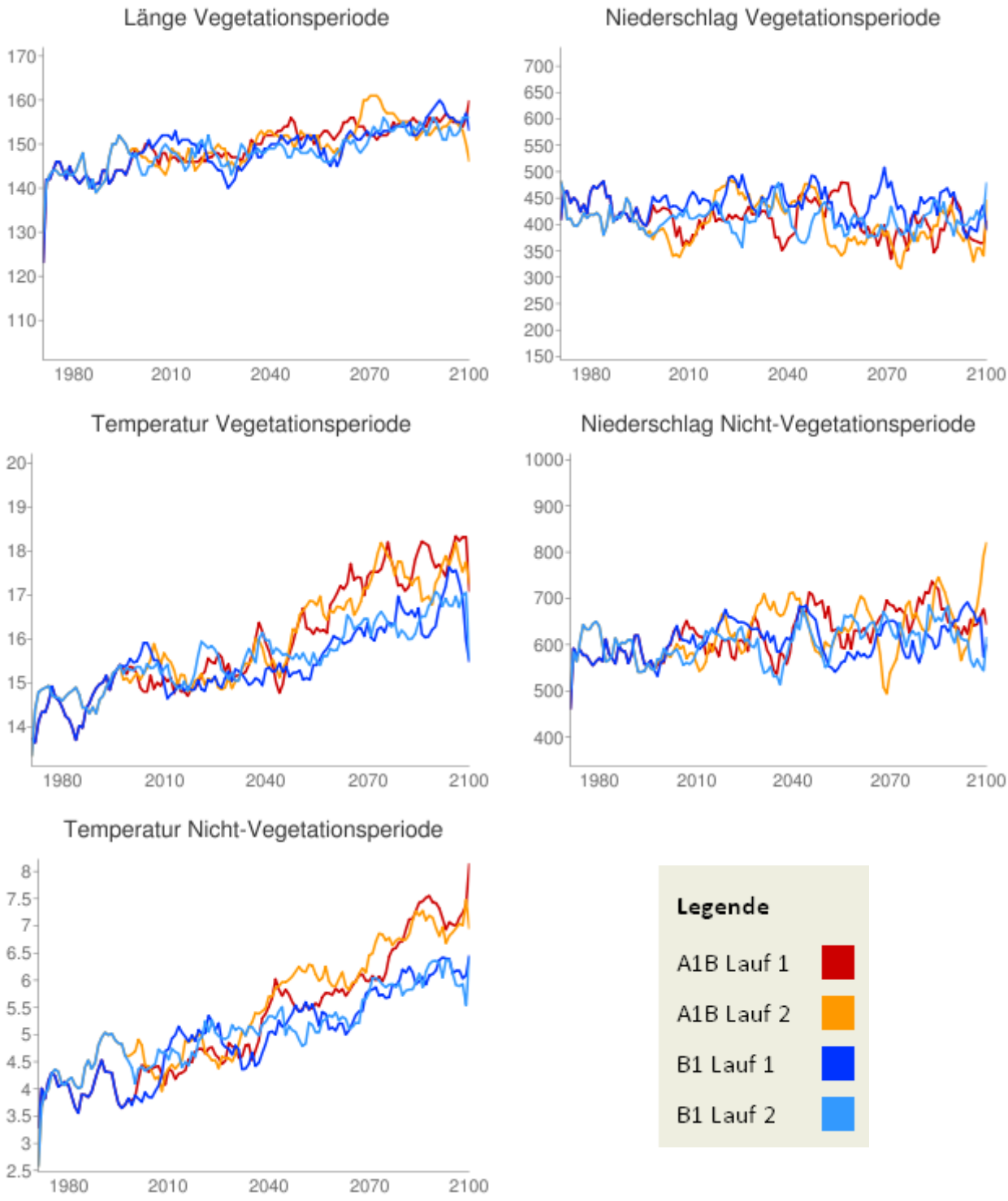
```
<html>
<head>

<title> Koordinatenwahl </title>

<link rel="stylesheet" type="text/css"
        href="http://extjs.cachefly.net/ext-2.2.1/resources/css/ext-all.css" />

<link rel="stylesheet" type="text/css"
        href="http://extjs.cachefly.net/ext-2.2.1/examples/shared/examples.css" />

<script src="http://extjs.cachefly.net/ext-2.2.1/adapter/ext/ext-base.js"
        type="text/javascript"></script>

<script src="http://extjs.cachefly.net/ext-2.2.1/ext-all.js"
        type="text/javascript"></script>

<script src="http://openlayers.org/dev/lib/OpenLayers.js" type="text/javascript"></script>
<script src="http://localhost/GeoExt/lib/GeoExt.js"  type="text/javascript"></script>

 <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;key=ABQIAAAAzr2EBOXU
Knm_jVnk0OJI7xSosDVG8KKPE1-m51RBrvYughuyMxQ-i1QfUnH94QxWIa6N4U6MouMmBA"
           type="text/javascript"></script>

<script type="text/javascript">

 var map, googlayer;

    Ext.onReady(function() {



            map = new OpenLayers.Map( 'map',{
                projection: "EPSG:900913",
                displayProjection: new OpenLayers.Projection("EPSG:4326"),
                units: "m",
                maxResolution: 156543.0339,
                //vorgeschlagen unter: http://trac.openlayers.org/wiki/SphericalMercator

                maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
                                                 20037508.34, 20037508.34),
                controls: [ new OpenLayers.Control.PanZoomBar(),
                            new OpenLayers.Control.Navigation(),
                            new OpenLayers.Control.MousePosition({numDigits:2})],
                restrictedExtent:new OpenLayers.Bounds(545000, 5914600, 1778900, 7578400)

                 } );
            // Zoom ab Deutschland bis zu einer Zelle m&ouml;glich
            OpenLayers.Layer.Google.prototype.MIN_ZOOM_LEVEL = 6 ;
            OpenLayers.Layer.Google.prototype.MAX_ZOOM_LEVEL = 12 ;

            googlayer = new OpenLayers.Layer.Google(
                 "Google",
                 {sphericalMercator:true}
                 );

           map.addLayer( googlayer
                         );

// Erstellen des MapPanels
           var panel = new GeoExt.MapPanel({
              height: 600,
              width: 600,
              map: map,
              renderTo:leg,
              title: 'Zur Auswahl von Koordinaten in die Karte klicken - mehrere Klicks er-
laubt'

           });
```

```
                map.events.register('click', map, function(e) {

                     //uebersetzen von xy-Pixelkoordinaten in LonLat-Werte des BaseLayers

                              point = map.getLonLatFromViewPortPx(e.xy);

                     // wird ueberschrieben, kein neues Objekt wird angelegt

                              point.transform(new OpenLayers.Projection("EPSG:900913"),
                                             new OpenLayers.Projection("EPSG:4326"));
                              x=point.lon;
                              y=point.lat;
                              //name, keine Id verwenden im entsprechenden Tag
                              document.coordform.lon.value = x;
                              document.coordform.lat.value = y;
                });

 });

</script>


</head>

<body >

<form action="/detailinfo/" target="_blank" method="get" name="coordform">

                  <table border="0" cellspacing="10" style="float:top; font:'arial';
                        font-size:13px; background-color:#eeeee0; empty-cells:show">

                           <colgroup>
                                   <col width="70">
                                   <col width="120">
                                   <col width="110">
                                   <col width="100">
                                   <col width="190">
                           </colgroup>

                           <tr height="5px">
                                  <td></td>
                                  <td colspan="4"><span style="color:#E50000;">{% if error %}
                                                        <ul> {% for err in error %}
                                                                <li > {{err}} </li>
                                                               {% endfor %}
                                                        </ul>
                                                        {% endif %} </span></td>
                           </tr>

                           <tr height="40px">
                                    <td></td>
                                    <td><b>L&auml;ngengrad:</b></td>
                                    <td><input name="lon" type="text"
                                               size="10" maxlength="6"></td>

                                    <td><b>Breitengrad:</b></td>
                                    <td><input name="lat" type="text"
                                                size="10" maxlength="6"></td>
                           </tr>

                            <tr height="40px">
                                    <td></td>
                                    <td><b>gleitendes Mittel:</b></td>
                                    <td><input name="mean" type="text" size="10"
                                                       maxlength="3" value="1" ></td>

                            </tr>

                            <tr height="40px">
                                    <td></td>
                                    <td colspan="2"><input type="submit"
                                               value=" Szenarienvergleich " name="scenario"
                                               style="width:200px;"></td>

                                    <td colspan="2"><input type="submit"
                                               value=" Baumartenvergleich " name="baumart"
                                               style="width:200px;"></td>
                            </tr>
```

```
                        <tr height="20px">
                                <td colspan="5"><div id="leg"
                             style="height:600px; width:600px; margin:10px;" ></div></td>

                        </tr>
                </table>

        </form>

</body>
</html>
```

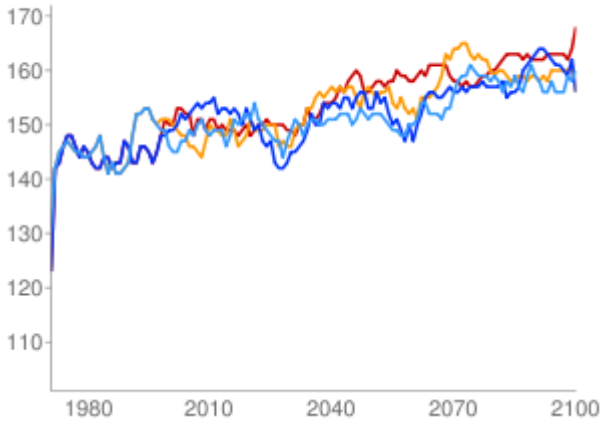
J Liniendiagramme für eine Koordinate

Buche

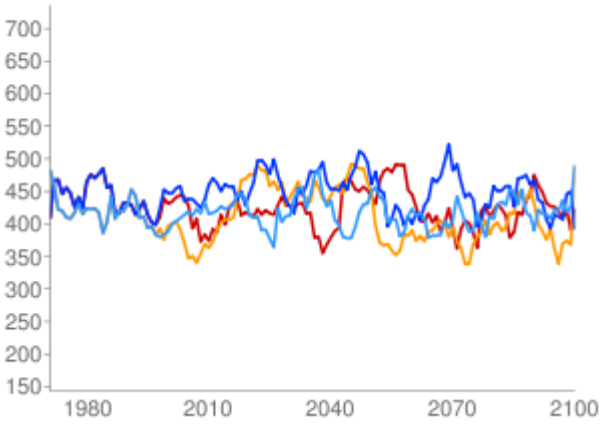


Fichte / Douglasie

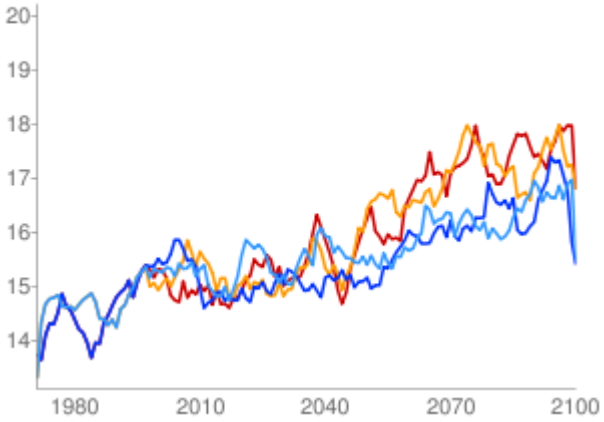
Länge Vegetationsperiode



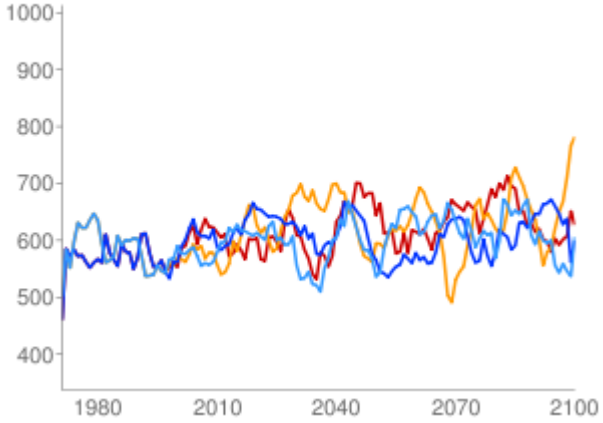
Niederschlag Vegetationsperiode



Temperatur Vegetationsperiode



Niederschlag Nicht-Vegetationsperiode

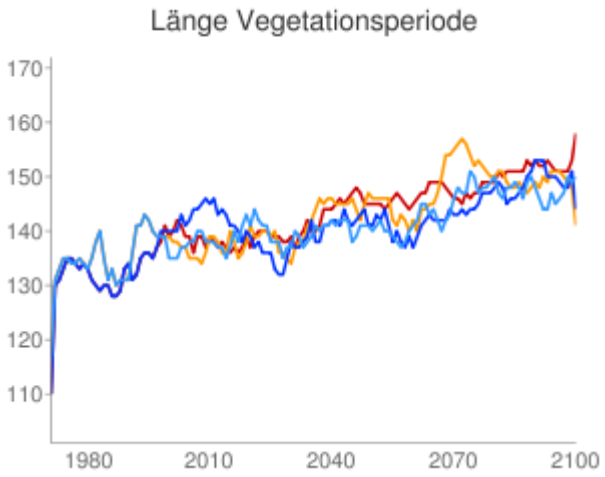


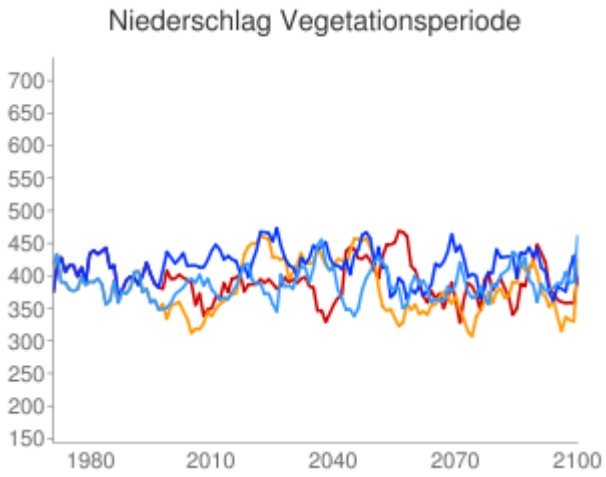
Temperatur Nicht-Vegetationsperiode

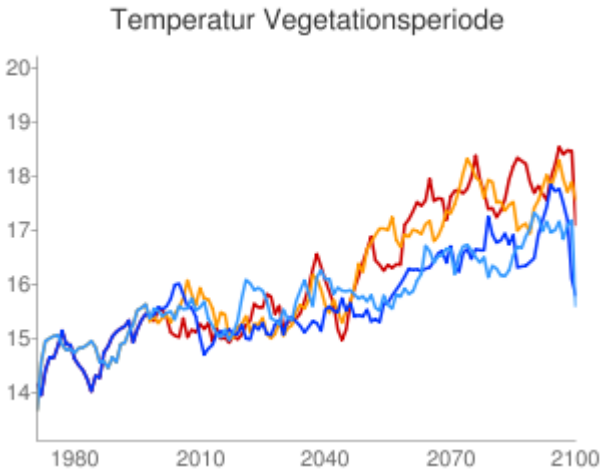
Legende

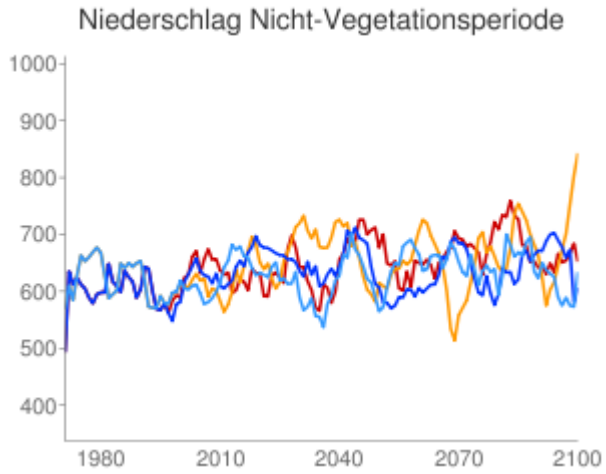
- A1B Lauf 1
- A1B Lauf 2
- B1 Lauf 1
- B1 Lauf 2

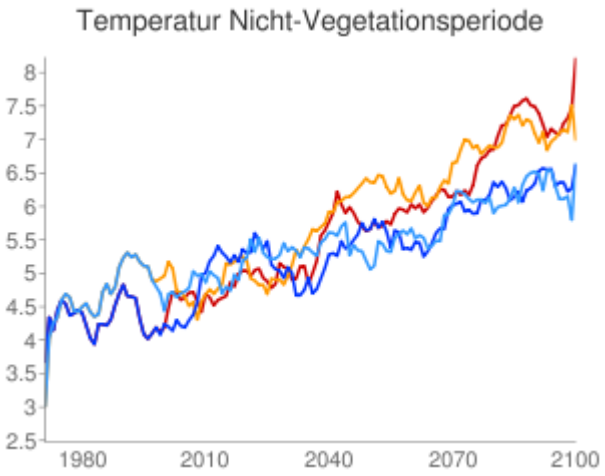
Kiefer

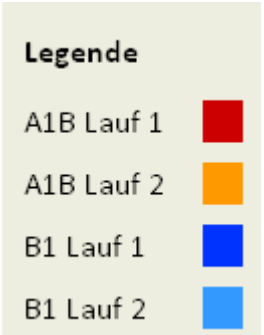

Länge Vegetationsperiode
170
160
150
140
130
120
110
1980
2010
2040
2070
2100


Niederschlag Vegetationsperiode
700
650
600
550
500
450
400
350
300
250
200
150
1980
2010
2040
2070
2100


Temperatur Vegetationsperiode
20
19
18
17
16
15
14
1980
2010
2040
2070
2100


Niederschlag Nicht-Vegetationsperiode
1000
900
800
700
600
500
400
1980
2010
2040
2070
2100


Temperatur Nicht-Vegetationsperiode
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
1980
2010
2040
2070
2100


Legende
A1B Lauf 1
A1B Lauf 2
B1 Lauf 1
B1 Lauf 2

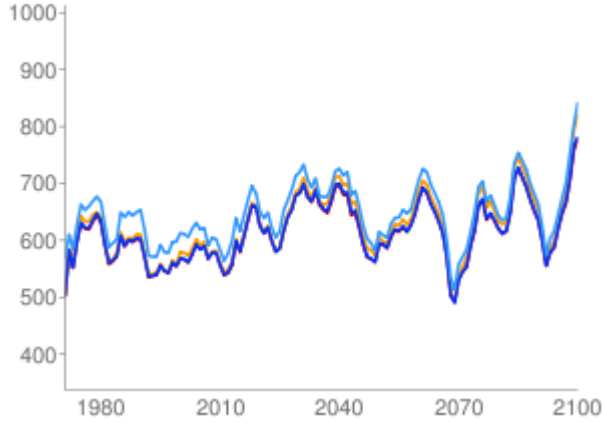
A1B Lauf 2

Länge Vegetationsperiode

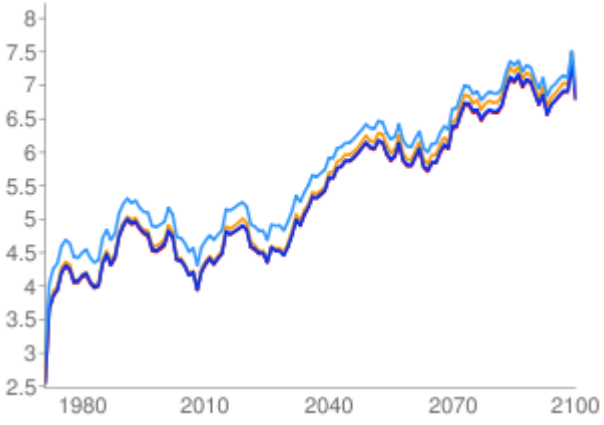
Niederschlag Vegetationsperiode

Temperatur Vegetationsperiode

Niederschlag Nicht-Vegetationsperiode



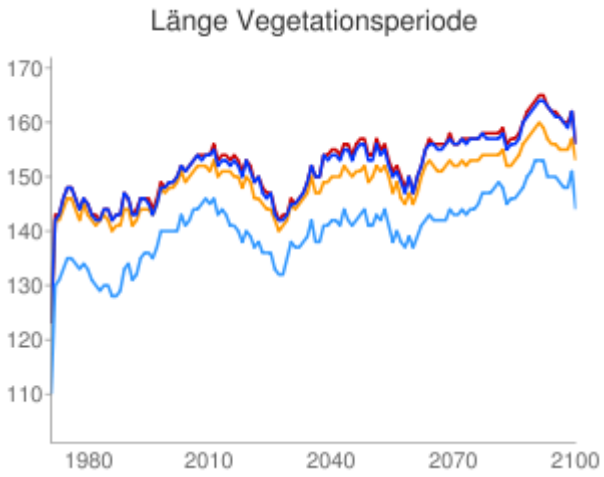
Temperatur Nicht-Vegetationsperiode

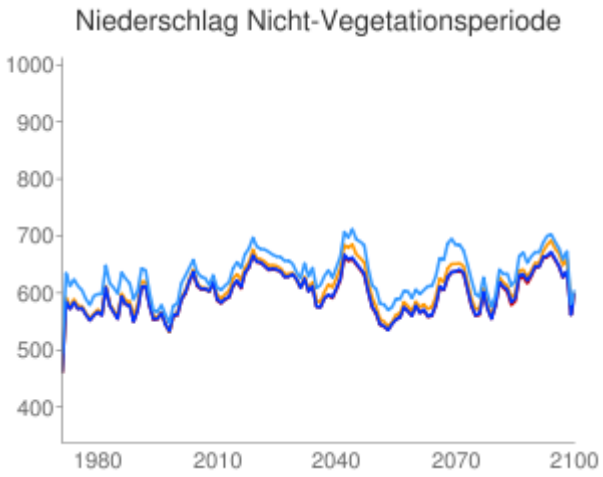


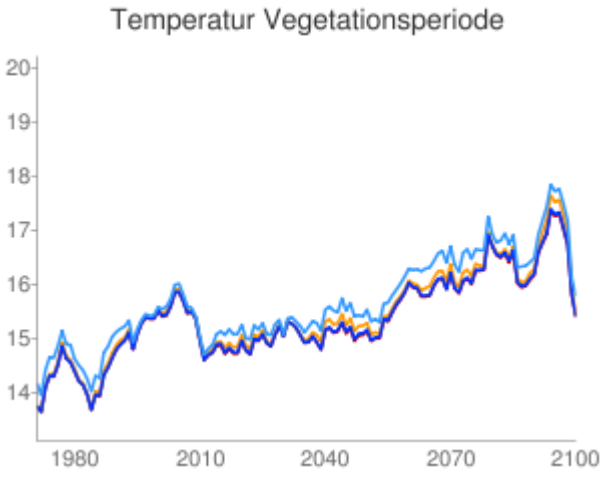
Legende

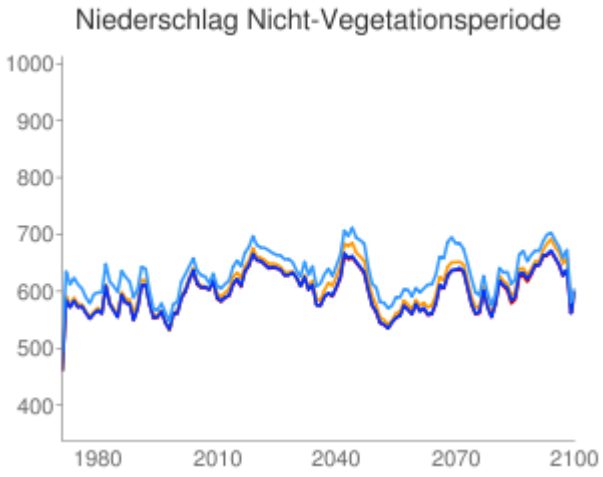
- Eiche
- Buche
- Fichte/Douglasie
- Kiefer

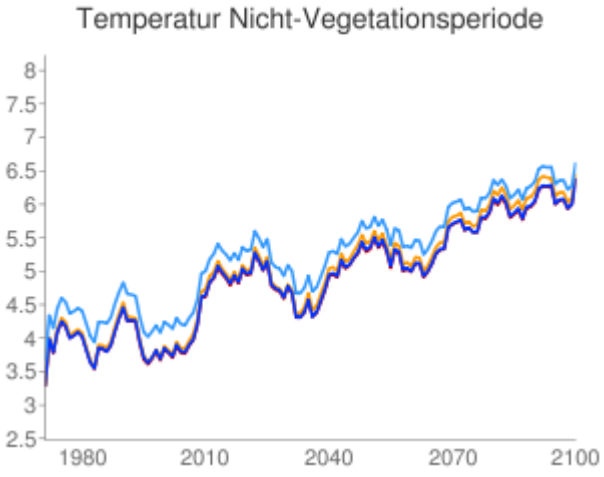
B1 Lauf 1


Länge Vegetationsperiode
170
160
150
140
130
120
110
1980
2010
2040
2070
2100


Niederschlag Nicht-Vegetationsperiode
1000
900
800
700
600
500
400
1980
2010
2040
2070
2100


Temperatur Vegetationsperiode
20
19
18
17
16
15
14
1980
2010
2040
2070
2100

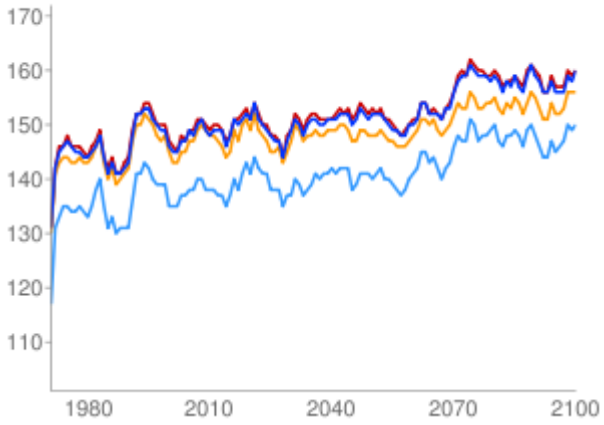

Niederschlag Nicht-Vegetationsperiode
1000
900
800
700
600
500
400
1980
2010
2040
2070
2100


Temperatur Nicht-Vegetationsperiode
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
1980
2010
2040
2070
2100

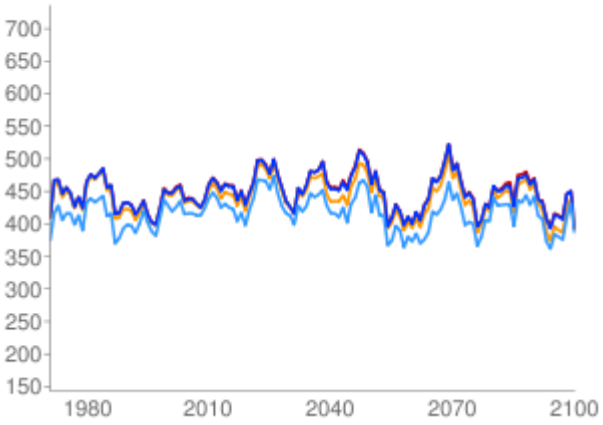

Legende
Eiche
Buche
Fichte/Douglasie
Kiefer

B1 Lauf 2

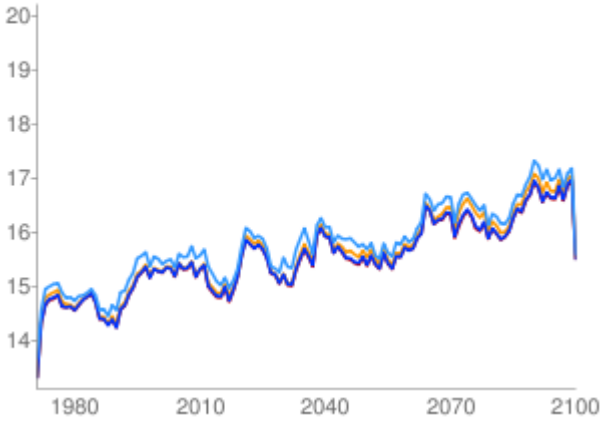
Länge Vegetationsperiode



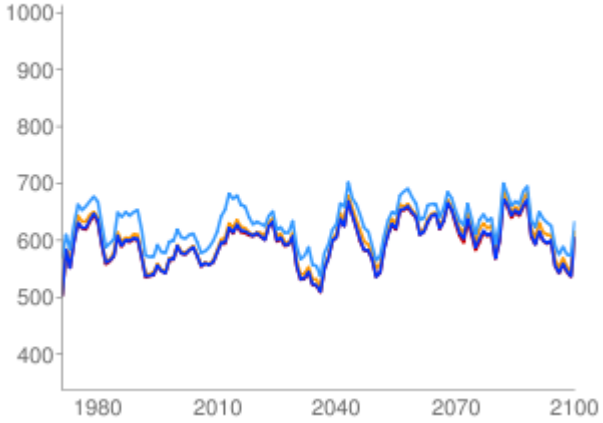
Niederschlag Vegetationsperiode



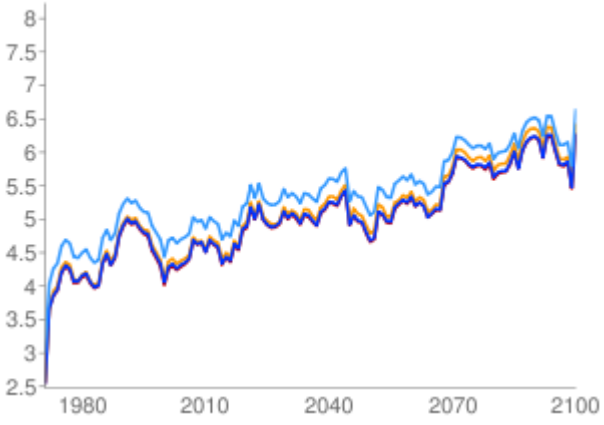
Temperatur Vegetationsperiode



Niederschlag Nicht-Vegetationsperiode



Temperatur Nicht-Vegetationsperiode



Legende

- Eiche
- Buche
- Fichte/Douglasie
- Kiefer

K Code für die Validierung und Formatierung der Koordinaten

```
def detailinfo(request):
    error=[]
    if 'scenario' in request.GET:

        if 'lon' in request.GET and 'lat' in request.GET: #sind die beiden Parameter ue-
berhaupt im Get-Request
                                                vorhanden

            if not request.GET['lon'] or not request.GET['lat']:
                error.append('Bitte stellen Sie sicher, dass Sie Koordinaten angegeben ha-
ben')

            elif       (float((request.GET['lon']).replace(',','.'))*10)       <       Dia-
gramme.objects.aggregate(b=Min('lon'))['b']:
                error.append('Der Wert fuer Breitengrad war zu niedrig')

            elif        float((request.GET['lon']).replace(',','.'))*10        >        Dia-
gramme.objects.aggregate(b=Max('lon'))['b']:
                error.append('Der Wert fuer Laengengrad war zu hoch')

            elif        float((request.GET['lat']).replace(',','.'))*10        <        Dia-
gramme.objects.aggregate(b=Min('lat'))['b']:
                error.append('Der Wert fuer Breitengrad war zu niedrig')

            elif        float((request.GET['lat']).replace(',','.'))*10        >        Dia-
gramme.objects.aggregate(b=Max('lat'))['b']:
                error.append('Der Wert fuer Laengengrad war zu hoch')

            else:
                <Code zur Vorbereitung der Kontextvariablen>

        return render_to_response('koordinatenauswahl.html', {'error':error,
                                                               })
```

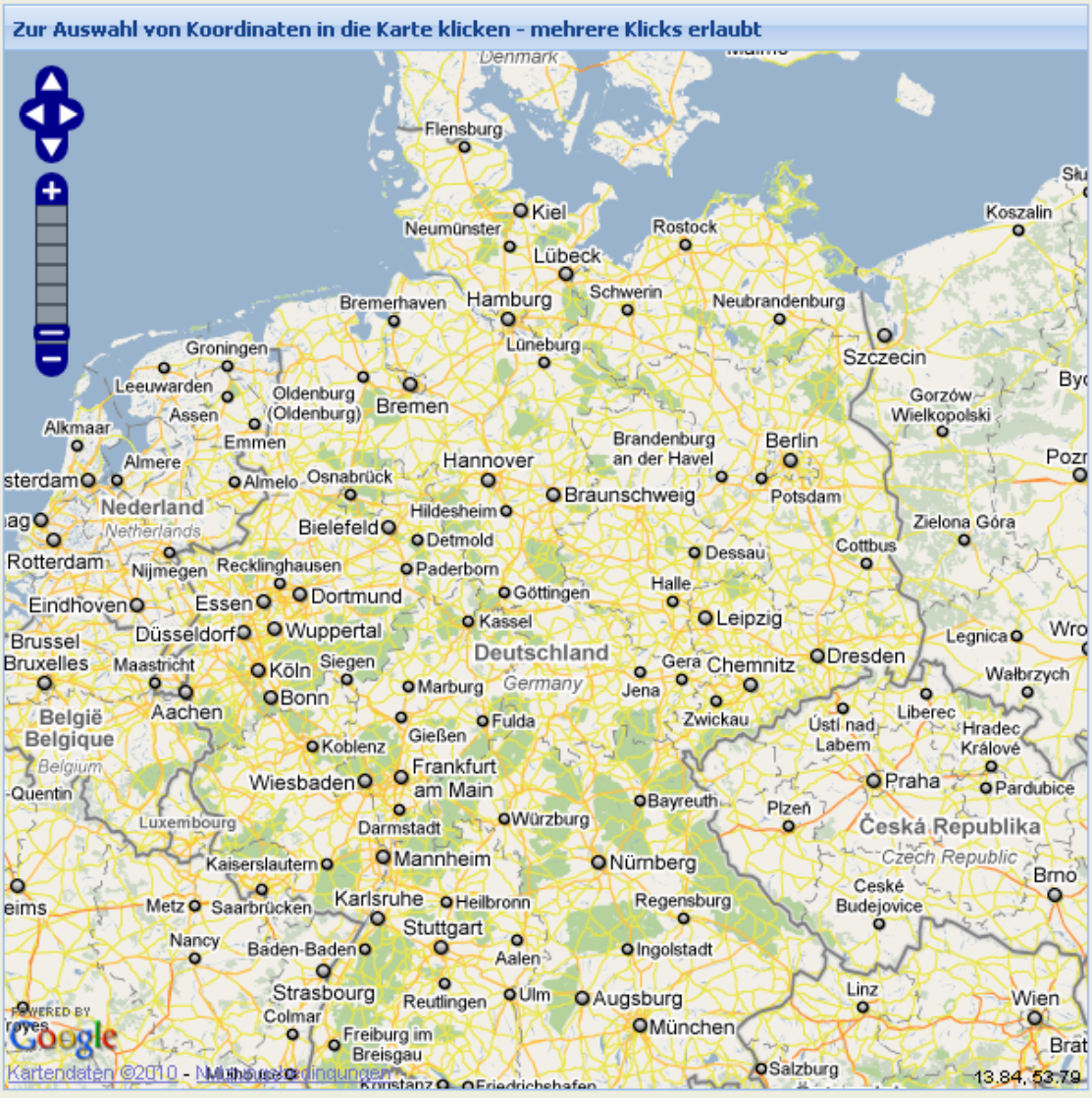
L Fehlermeldung

Der Wert fuer Breitengrad war zu niedrig

Längengrad: **Breitengrad:**

gleitendes Mittel: 1

Szenarienvergleich Baumartenvergleich



M Kontextvariablen zur Achsen-Skalierung

```
for i in range(5):
    if i == 0:
        a1='temp_vp_run1_a1b'
        b1='temp_vp_run2_a1b'
        c1='temp_vp_run1_b1'
        d1='temp_vp_run2_b1'
    elif i==1:
        a1='temp_nvp_run1_a1b'
        b1='temp_nvp_run2_a1b'
        c1='temp_nvp_run1_b1'
        d1='temp_nvp_run2_b1'
    elif i==2:
        a1='prec_vp_run1_a1b'
        b1='prec_vp_run2_a1b'
        c1='prec_vp_run1_b1'
        d1='prec_vp_run2_b1'
    elif i==3:
        a1='prec_nvp_run1_a1b'
        b1='prec_nvp_run2_a1b'
        c1='prec_nvp_run1_b1'
        d1='prec_nvp_run2_b1'
    elif i==4:
        a1='length_run1_a1b'
        b1='length_run2_a1b'
        c1='length_run1_b1'
        d1='length_run2_b1'
    a=Diagramme.objects.filter(lon=lons,
              lat=lats).aggregate(x=Min(a1),y=Max(a1))
    b=Diagramme.objects.filter(lon=lons,
              lat=lats).aggregate(x=Min(b1), y=Max(b1))
    c=Diagramme.objects.filter(lon=lons,
              lat=lats).aggregate(x=Min(c1), y=Max(c1))
    d=Diagramme.objects.filter(lon=lons,
              lat=lats).aggregate(x=Min(d1), y=Max(d1))

    if i == 0:
        skalTempMinVP=min((a['x'],b['x'],c['x'],d['x']))
        skalTempMaxVP=max((a['y'],b['y'],c['y'],d['y']))
    elif i == 1:
        skalTempMinNVP=min((a['x'],b['x'],c['x'],d['x']))
        skalTempMaxNVP=max((a['y'],b['y'],c['y'],d['y']))
    elif i == 2:
        skalPrecMinVP=min((a['x'],b['x'],c['x'],d['x']))
        skalPrecMaxVP=max((a['y'],b['y'],c['y'],d['y']))
    elif i == 3:
        skalPrecMinNVP=min((a['x'],b['x'],c['x'],d['x']))
        skalPrecMaxNVP=max((a['y'],b['y'],c['y'],d['y']))
    elif i == 4:
        skalLenMinVP=min((a['x'],b['x'],c['x'],d['x']))
        skalLenMaxVP=max((a['y'],b['y'],c['y'],d['y']))
```

N Model

```
class Diagramme(models.Model):
    temp_vp_run1_a1b = models.FloatField()
    temp_vp_run2_a1b = models.FloatField()
    prec_vp_run1_a1b = models.FloatField()
    prec_vp_run2_a1b = models.FloatField()
    temp_nvp_run1_a1b = models.FloatField()
    temp_nvp_run2_a1b = models.FloatField()
    prec_nvp_run1_a1b = models.FloatField()
    prec_nvp_run2_a1b = models.FloatField()
    length_run1_a1b = models.IntegerField()
    length_run2_a1b = models.IntegerField()
    temp_vp_run1_b1 = models.FloatField()
    temp_vp_run2_b1 = models.FloatField()
    prec_vp_run1_b1 = models.FloatField()
    prec_vp_run2_b1 = models.FloatField()
    temp_nvp_run1_b1 = models.FloatField()
    temp_nvp_run2_b1 = models.FloatField()
    prec_nvp_run1_b1 = models.FloatField()
    prec_nvp_run2_b1 = models.FloatField()
    length_run1_b1 = models.IntegerField()
    length_run2_b1 = models.IntegerField()
    lon = models.IntegerField(primary_key=True)
    lat = models.IntegerField( primary_key=True)
    species = models.IntegerField(primary_key=True)
    years = models.IntegerField(primary_key = True)
    class Meta:
        db_table = u'diagramme'
        ordering = ['years','species']
    def __unicode__(self):
       return u"%s, %s, %s, %s" %(self.lon, self.lat, self.species, self.years)
```

O Funktionen

```
def mov_avg(gesamtlaenge, helper, w):
    #Fasst die Daten, die in den Querysets vorliegen als moving averages
    #in arrays zusammen,
    #return wird weiterverwendet in processQuerys

    data=zeros(gesamtlaenge)

    for i in range(0,gesamtlaenge):

        #erste werte
        if i ==0:
            data[0]=helper[0:1]

        if i > 0 and i < int(math.floor(float(w)/2)):
            data[i]=helper[0:2*i+1].sum()/(2*i+1)

        if i < gesamtlaenge-(w-1):
            j=i+w
            t=math.ceil(float(w)/2.0)-1.0+float(i)
            data[t]=helper[i:j].sum()/w

        elif i > gesamtlaenge-(w-1) and i < gesamtlaenge-1:
            data[i+math.floor((gesamtlaenge-i)/2)]=helper[i:].sum()/(gesamtlaenge-i)


        elif i == gesamtlaenge-1:
            data[i]=helper[i]


    return data




def liste(j,data1,data2,data3,data4,laenge):
    #bildet aus den einzelnen moving averages arrays Listen, die dem
    #in den Grafiken benoetigten Datenformat entsprechen
    #return wird weiterverwendet in processQuerys
    Liste=[]
    Liste2=[]
    for i in range(0,laenge):  #laeuft von 0-129
        jahr=str(int(j[i]))
        if jahr != '1980' and jahr != '2010' and jahr != '2040' and jahr != '2070' and jahr !=
'2100':
            jahr = ''

        x1=data1[i]
        x2=data2[i]
        x3=data3[i]
        x4=data4[i]
        Liste2.extend([jahr,x1,x2,x3,x4])
        Liste.append(Liste2)
        Liste2=[]
    return Liste




def processQuerys(baumart, gesamtlaenge, w, h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12,
h13, h14, h15, h16, h17, h18, h19, h20, h21):

    #
    i=0
    for value in baumart:

        #Uebergeben der Werte im Queryset an ein Array
        h1[i]=value.years
        h2[i]=value.length_run1_a1b
        h3[i]=value.length_run2_a1b
        h4[i]=value.length_run1_b1
        h5[i]=value.length_run2_b1
```

```
    h6[i]=value.temp_vp_run1_a1b
    h7[i]=value.temp_vp_run2_a1b
    h8[i]=value.temp_vp_run1_b1
    h9[i]=value.temp_vp_run2_b1

    h10[i]=value.temp_nvp_run1_a1b
    h11[i]=value.temp_nvp_run2_a1b
    h12[i]=value.temp_nvp_run1_b1
    h13[i]=value.temp_nvp_run2_b1

    h14[i]=value.prec_vp_run1_a1b
    h15[i]=value.prec_vp_run2_a1b
    h16[i]=value.prec_vp_run1_b1
    h17[i]=value.prec_vp_run2_b1

    h18[i]=value.prec_nvp_run1_a1b
    h19[i]=value.prec_nvp_run2_a1b
    h20[i]=value.prec_nvp_run1_b1
    h21[i]=value.prec_nvp_run2_b1

    i += 1


jahr=(mov_avg(gesamtlaenge,h1, w)).round(0)
z = mov_avg(gesamtlaenge,h2, w).round(0)
z2 = mov_avg(gesamtlaenge,h3, w).round(0)
z3 = mov_avg(gesamtlaenge,h4, w).round(0)
z4 = mov_avg(gesamtlaenge,h5, w).round(0)
length = liste(jahr,z,z2,z3,z4,gesamtlaenge)

z = mov_avg(gesamtlaenge,h6, w).round(2)
z2 = mov_avg(gesamtlaenge,h7, w).round(2)
z3 = mov_avg(gesamtlaenge,h8, w).round(2)
z4 = mov_avg(gesamtlaenge,h9, w).round(2)
temp_vp = liste(jahr,z,z2,z3,z4,gesamtlaenge)

z = mov_avg(gesamtlaenge,h10, w).round(2)
z2 = mov_avg(gesamtlaenge,h11, w).round(2)
z3 = mov_avg(gesamtlaenge,h12, w).round(2)
z4 = mov_avg(gesamtlaenge,h13, w).round(2)
temp_nvp = liste(jahr,z,z2,z3,z4,gesamtlaenge)

z = mov_avg(gesamtlaenge,h14, w).round(0)
z2 = mov_avg(gesamtlaenge,h15, w).round(0)
z3 = mov_avg(gesamtlaenge,h16, w).round(0)
z4 = mov_avg(gesamtlaenge,h17, w).round(0)
prec_vp = liste(jahr,z,z2,z3,z4,gesamtlaenge)

z = mov_avg(gesamtlaenge,h18, w).round(0)
z2 = mov_avg(gesamtlaenge,h19, w).round(0)
z3 = mov_avg(gesamtlaenge,h20, w).round(0)
z4 = mov_avg(gesamtlaenge,h21, w).round(0)
prec_nvp = liste(jahr,z,z2,z3,z4,gesamtlaenge)


return length,temp_vp,temp_nvp,prec_vp,prec_nvp
```

Hiermit versichere ich gemäß § 9 Abs. 5 der Master-Prüfungsordnung vom 22.07.2005, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum:_____ Unterschrift:_____