Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„The performance assessment of vegetation models“

Demonstration of a benchmarking system application

vorgelegt von

B.Sc. Jan Kowalewski
103250, UNIGIS MSc Jahrgang 2013

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachterin:
Ass.-Prof. Dr. Gudrun Wallentin

Reykjavik, 25.01.2016

Dedication

I dedicate this thesis to my family, which has always supported and encouraged me in my intentions.

Declaration

I declare that I have developed and written the enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Reykjavik, 25.01.2016 Jan Kowalewski

Acknowledgements

First of all, I would like to express my deep gratitude to Dr. Susanne Rolinski for her guidance, advice, support and patience throughout the development of this thesis, and in previous years.

Next, I would like to thank the land use modelling group at the Potsdam-Institute for Climate Impact Research. Working there has broadened my horizon and motivated me to extend my skills beyond my university education.

My sincere thanks also go to Ass.-Prof. Dr. Gudrun Wallentin and the UNIGIS team at the University of Salzburg.

Finally, I would like to thank my brother, Eric Kowalewski, for his encouraging words and for proofreading parts of my thesis.

Abstract

In the course of creating this thesis, the author has designed and implemented a benchmarking system application for vegetation models, with due regard to user-requirements and recommendations given in relevant publications. This work thereby contributes to recent efforts in the vegetation modelling community, to establish a generally-accepted, standardised benchmarking system for effective and comprehensive model performance evaluation and improvement tracking.

Particular emphasis is laid on the discussion and demonstration of the scoring system of metrics, which has been constructed for the benchmarking system. It includes a set of complementary measures for value distribution, model error and similarity, and a concise visual method for relative performance assessment between different models and processes, as well as for model improvement tracking.

Another important system feature is a clear, user-friendly procedure for spatial data integration, combined in one function. This function facilitates standardised harmonisation of model results and reference datasets, and contains spatial extraction tools for the investigation of varying model performances at different spatial scales.

Implementation of the benchmarking system has been realised in the *R* software environment, using an object-oriented design and programming approach. The system comprises two customised spatial data classes, including a metadata data type, and several utility and analysis functions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In order to increase the understanding of natural and anthropogenic systems, and to forecast their possible future states, scientists build models that are based on the mathematical description of real-world processes, such as photosynthesis, or energy- and water cycles. By definition, a model can only examine certain aspects of reality, and is therefore never an exact or complete representation, but rather an approximation of reality. One group of models, called vegetation models, has been developed, amongst others, to simulate and increase the understanding of the global carbon and water cycle. In addition, these models are often used to analyse potential shifts in vegetation, and associated biogeochemical and hydrological cycles, in response to climate change. Therefore, simulations produced by these models can serve as a basis for estimating future changes in biome distribution, or analyse the effect of climate change on soil carbon content, and are thus important for the development of adaptation strategies in fields such as agriculture or conservation management. Making theses models more robust and less uncertain, is fundamental if future policies or adaptation strategies are to be based on their forecasts.

An increasingly important concept that is used for evaluating and improving the performance of a model is benchmarking. This procedure employs a scoring system of metrics that relate model results of different process variables to pre-defined performance levels (i.e. benchmarks), enabling model developers to systematically identify processes and variables that are subject to high errors or have been neglected so far, although essential for model performance. Subsequently, developers can perform targeted parameter adjustments in the model and track resulting improvements.

A standardised benchmarking system for vegetation models should offer great flexibility with regard to the integration of reference datasets of different sources and formats, and facilitate performance analysis and improvement tracking at different spatial scales. It therefore makes sense to incorporate specialised functions found in Geographical Information System (GIS), since both model results and reference datasets are of the spatial type. In addition, this benchmarking system should be readily accessible and user-friendly, to ensure a wide acceptance in the vegetation modelling community.

1.2 Research question and sub-goals

The central objective of the underlying thesis is to present and demonstrate the functionality of an open-source benchmarking system that can be used to evaluate the performance of fundamental processes in vegetation models, by identifying systematic errors and by subsequently tracking model improvements with the aid of a comprehensive scoring system of metrics.

The sub-goals to achieve this aim are to:

- Evaluate suitable methods for the integration and manipulation of model results and reference datasets from different sources.
- Determine a set of complementary metrics, feasible for a comprehensive model evaluation, and targeted identification of weaknesses and improvement potential in the model.
- Implement a customised, user-friendly open-source benchmarking system that includes functions for the integration and manipulation of model results and reference data, as well as for the targeted performance evaluation and improvement of vegetation models.
- Demonstrate the use of the benchmarking system with model results from LPJmL (Lund-Potsdam-Jena with managed Land), a dynamic global vegetation model that simulates physical, biological and biogeochemical processes, as well as vegetation dynamics in natural and agricultural ecosystems at a spatial resolution of 0.5° latitude and longitude.

1.3 Overview: thesis development

Figure 1.1 gives a schematic view of the main steps that have been taken in the development of the benchmarking system and thesis. The contents of individual steps are detailed in subsequent chapters, and are complemented by a system demonstration and conclusion at the end of this work.

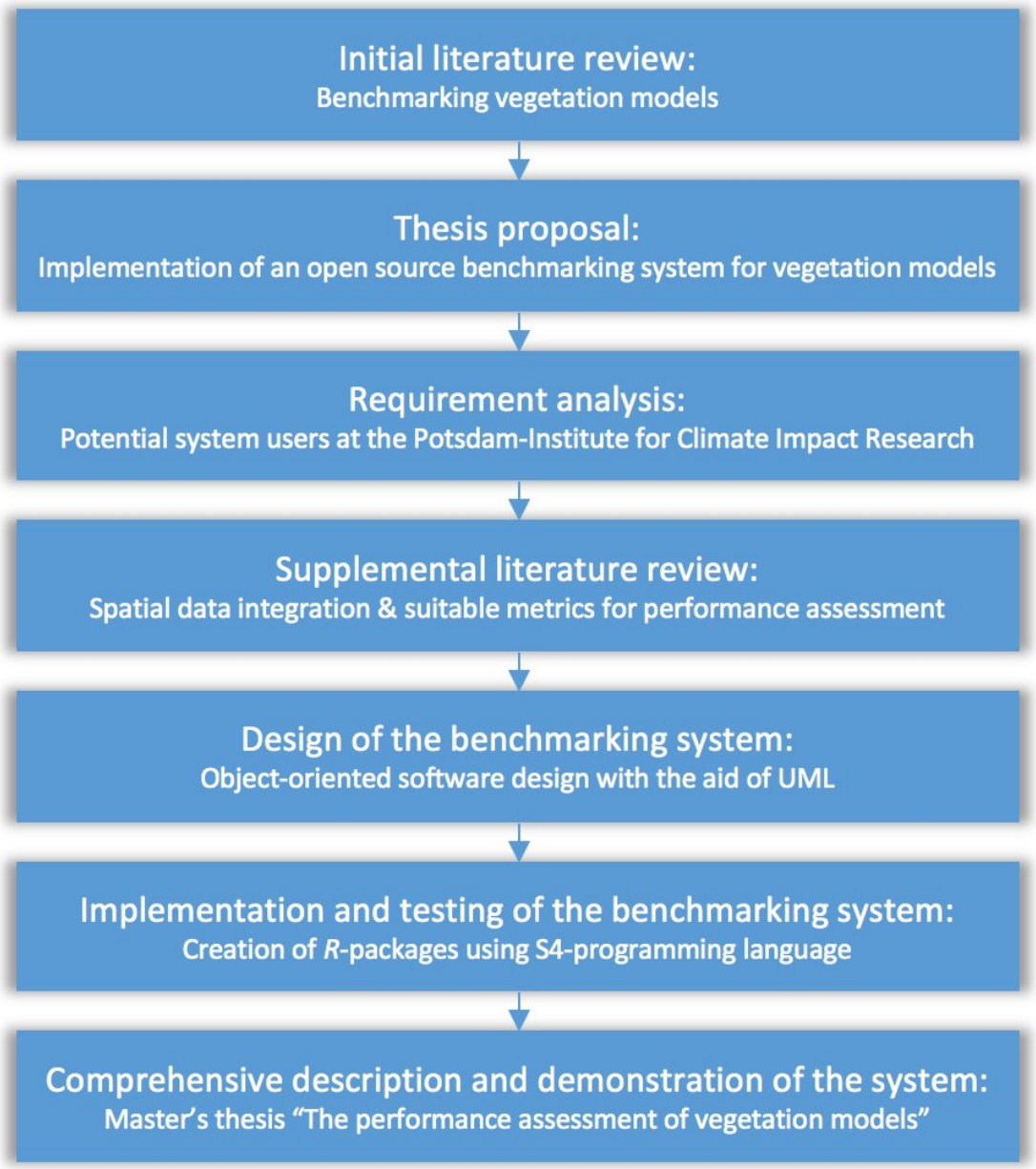


Figure 1.1: Steps taken in the development of the underlying work.

Chapter 2

Vegetation models: context and description

This chapter introduces the role and application of vegetation models in climate research, before treating important concepts and natural processes on which these models are based. Finally, the LPJmL model, which will be used for demonstration purposes in this thesis, is described in more detail.

2.1 Modelling the climate system

Scientists, policy-makers, and ultimately the global population, is interested in predictions about the future state of our climate system. Rising global temperatures, induced by human activity, are likely to have a significant effect on sea level, frequency of weather extremes, changing weather patterns, and terrestrial and aquatic ecosystems, and thus bear severe implications for human economies, health, safety, and food- and water security (e.g. Intergovernmental Panel on Climate Change, 2014a). In order to gain more knowledge about the climate system as a whole and about the feedbacks between its interrelated components, that is the atmosphere,

hydrosphere, land surface, biosphere and cryosphere, scientists create models to handle the system's complexity. Predictive models simplify reality by using theory-based mathematical descriptions or statistical relationships to represent real-world processes. Coupled global climate models (GCMs), often interchangeably called (coupled) general circulation models, or simply climate models, produce results on spatial grids for possible future states of the world (i.e. scenarios) that, for example, form the basis of the reports issued by the Intergovernmental Panel on Climate Change (Intergovernmental Panel on Climate Change, 2014b). They are comprised of several connected sub-models that have been developed to represent the physical processes of different components of the climate system. For example, the Community Climate System Model (CCSM), maintained by the U.S. National Center for Atmospheric Research (NCAR), employs models of the circulation in the atmosphere and ocean, a sea-ice model, and a land-surface model (LSM) (Gent et al., 2011). Latter group of models simulate the energy and water budget of the terrestrial surface. They have undergone significant changes since the first implementation by Manabe (1969), a bucket model with a very simple hydrology and energy balance equation. The second generation of LSMs honoured the role of vegetation in the exchange of energy and water between land and atmosphere, by including more detailed vegetation characteristics, notably an empirical model of canopy conductance for transpiration (Pitman, 2003). Ecosystem processes became the focus of attention in the third generation of LSMs, which include a representation of the carbon cycle - a major determinant of climate (see Pitman, 2003). This advancement highlighted the importance of the vegetation component in the climate system as an interface of the water, energy and carbon cycle, and has been termed as "the greening of LSMs" (Pitman, 2003). Recent changes in LSMs increasingly blurred boundaries to a related group of models, the so-called dynamic global vegetation models (DGVMs) (Prentice et al., 2015), which have been developed alongside LSMs. DGVMs simu-

late changes in the pattern and composition of potential vegetation, including related physical, chemical and biological processes, and are therefore particularly suitable for examining the feedbacks that occur between changes in atmospheric CO_2, climate, and the biosphere. Several LSMs have already been converted to or replaced by DGVMs (e.g. Krinner et al., 2005; Blyth et al., 2006). Climate models that deploy a third-generation LSM, or a DGVM, for the land component are generally referred to as Earth system models (ESMs), which implies a stronger consideration of complex feedback processes in the climate system (e.g. Hurrell et al., 2013). DGVMs are the focus of this work and are described in more detail in the next section.

2.2 Dynamic global vegetation models

The first DGVMs were developed in the second half of the 1990s (Foley et al., 1996; Brovkin et al., 1997; Friend et al., 1997), and continued to appear through the 2000s (e.g. Sitch et al., 2003). They generally simulate the terrestrial carbon and water cycle, and derive patterns of potential vegetation in response to climate, disturbances and competition on a spatial grid. Global potential vegetation is represented by plant functional types (PFTs), which can be understood as plant prototypes with similar traits, like leaf type (broad- or needle-leaved), phenology (evergreen, rain-green or summer-green), physiognomy (woody or grass), climate (tropical, temperate, or boreal) and photosynthetic pathway, i.e. the mechanism by which CO_2 is fixed by plants. Individual PFTs prescribe parameters and threshold values for plant and ecosystem functions that determine, for example, carbon allocation, hydrological properties, plant establishment, growth, competitiveness, and mortality. DGVMs can differ in several aspects, like the temporal and spatial resolution, the number and specification of PFTs, and the emphasis on and implementation of different processes, i.e. the number, detail and type (e.g. statistical vs. mechanistic) of

process representations. In the following two subsections, important processes that are modelled in DGVMs are discussed.

2.2.1 Process representation

Processes represented in DGVMs generally describe plant physiological functions, the fluxes of energy and water between ecosystems and the atmosphere, and in between ecosystem elements (biophysical processes), the cycling of nutrients (biogeochemical processes), and vegetation dynamics. Table 2.1 provides an overview of some important processes and concepts, with vegetation dynamics listed separately, to highlight the initial distinguishing feature of DGVMs compared to, for example, LSMs. These processes, which can occur on a multitude of scales, form a system of interrelated parts that interact with other components of the climate system, such as the atmosphere. Consider, for example, a simplified pathway of carbon in an ecosystem: Carbon, in the form of CO_2, and moisture enter and leave plants through stomata, at a rate determined by stomatal conductance. Photosynthesis is the conversion of intra-cellular CO_2, water and light-energy to carbohydrates, which store the sun's energy in chemical bonds, and play a central role in living organisms. Growth and maintenance respiration by plants again re-releases CO_2, water and energy in the form of heat. A frequently-used measure that quantifies the rate of photosynthesis, in terms of the amount of carbon fixed from the atmosphere per unit area and time (e.g. g C m^{-2} hr^{-1}), is called gross primary productivity (GPP). The difference between GPP and the amount of carbon used in plant metabolism, i.e. autotrophic respiration, is called net primary productivity (NPP), and represents the amount of carbon that can be allocated to the structural components of a plant, like leaves, stems and roots. NPP is therefore an approximate measure for plant growth in a given area and time interval.

Table 2.1: Description of some important processes in vegetation models.

	Process/Concept	Description	Implementation
Physiological - Biophysical - Biogeochemical	Photosynthesis	Process by which plants convert CO_2, water and sunlight to carbohydrates and chemical energy. This process removes CO_2 from the atmosphere.	Farquhar et al. (1980), Collatz et al. (1991), Collatz et al. (1992)
	Respiration	Metabolic process by which organisms turn chemical energy into useful energy, e.g. for growth and maintenance. This process adds CO_2 to the atmosphere.	Dependent on sapwood volume, temperature, evaporative demand, C:N ratio
	Phenology	Periodic life cycle events of an organism, often initiated by environmental factors: e.g. leaf start growing at a specific temperature threshold	PFT-specific thresholds for moisture, temperature, growth, productivity + daylength requirements
	Energy fluxes	Processes such as solar radiation absorption, reflected radiation, latent heat flux, sensible heat flux	Beer's law of light attenuation
	Water fluxes	Processes such as transpiration, evaporation, percolation, runoff	Darcy's Law; Monteith (1981), Monteith (1995); surface-runoff and drainage; snow hydrology
	Carbon allocation	The partitioning of carbon to different structural plant parts: leaves, stem, roots	Fixed amount; foliage- and demand-based; allometric relationships
Vegetation Dynamics	Competition	Concept describing the competition between organisms for territory or resources	Individual- or area-based competition for light and H_2O
	Establishment	Conditions that determine the establishment of different types of vegetation	PFTs establish uniformly or proportional to available area, depending on climatic thresholds
	Mortality	Conditions under which a plant is susceptible to death	Baseline carbon balance, extreme temperatures, wind throw, fire
	Disturbance	Temporary changes in the environment that usually cause long-lasting changes in an ecosystem	Deterministic fire occurrence and PFT-specific fire resistance

Dead and dropped leaves and roots undergo decomposition by chemical reactions or other organisms, which again respire CO_2 to the atmosphere, and release heat and water in the process. The remaining, more stable carbon-compounds are stored in different soil layers, where they continue to decompose at distinct rates. Decomposition rates themselves are primarily dependent on temperatures and moisture status. The described sequence of processes illustrates the close linkage of carbon, water and energy cycles, and adequate mathematical descriptions are required to achieve realistic results. Examples of some mathematical descriptions are given in Table 2.1.

2.2.2 Adding agriculture: LPJmL model description

Besides evaluating the role of natural terrestrial ecosystems in the global carbon and hydrological cycle under climate change, it is equally important to gain a deeper understanding of how the large-scale conversion of natural vegetation to agricultural land affects climate and human society. In order to do this, Bondeau et al. (2007) have developed an extended version of the LPJ-DGVM (Sitch et al., 2003), called LPJmL (mL stands for 'managed Land'), which applies the concept of PFTs to the most common cultivated plants (crops) and managed grasslands used for livestock feed. Crop functional types (CFTs) contain physiological descriptions adapted to the specific phenology and growth of plants that are bred for high yields and short life cycles (die-off or harvest after 1 year). Shorter life-cycles and different growth characteristics also require a modified treatment of carbon allocation and vegetation dynamics, which are computed daily for CFTs in LPJmL, in contrast to annual updating for the natural vegetation (PFTs). An additional factor that can influence phenology, growth, or the fluxes of carbon and water in agricultural systems is management.

Table 2.2: Important features of the LPJmL model.

Representation of natural vegetation, crops and bioenergy	9 PFTs	Tropical broad-leaved evergreen Tropical broad-leaved rain-green Temperate needle-leaved evergreen Temperate broad-leaved evergreen Temperate broad-leaved summer-green Boreal needle-leaved evergreen Boreal needle-leaved summer-green C3-Grasses C4-Grasses
	12+1 CFTs	Temperate cereals Tropical cereals Rice Maize Pulses Temperate roots Tropical roots Sunflower Soybean Groundnuts Rapeseed Sugar cane Managed grasslands (C3/C4)[1]
	Bioenergy	Tropical bioenergy tree Temperate bioenergy tree Bioenergy grasses
Configuration	Spatial	Reference system: WGS1984 Resolution: 0.5° x 0.5° No. Cells: 67420 land cells
	Temporal	Daily Monthly Annual
Main outputs	Daily	All variables, if simulation is done for one grid cell at a time
	Monthly	C-fluxes: GPP, NPP, respiration; Water fluxes: evapotranspiration, runoff, discharge
	Annually	Carbon stored in: vegetation, litter, soil; Carbon released by fire; Crop yield; Crop residue yield; Fire return interval; Foliage projected cover of natural vegetation

[1]Same as natural grasses. The fraction of managed grass is determined by landuse input data.

The model includes various practices, such as crop sowing dates (Waha et al., 2012), irrigation, connected with a river-routing scheme (Rost et al., 2008), harvesting methods (i.e. residues can decompose fast or slow) and different grassland management options (i.e. carbon is removed from grass leaves or incorporated into soil organic matter). Dynamic features, like fire-disturbance (Thonicke et al., 2010), and permafrost-soil melting and hydrology (Schaphoff et al., 2013), with their effects on carbon, water and energy fluxes, can also be studied. LPJmL is an offline-model, meaning it is not coupled to a climate model, and is run in a spatially-distributed mode on a 0.5° x 0.5° land-cell grid, with several input datasets, such as atmospheric CO_2, climate and land-use distribution. Historical land-use with distributions and areas of crop-, grass-, and irrigated land is derived from various map products (e.g. Ramankutty and Foley, 1999; Döll and Siebert, 1999). These are used to derive fractions, per year and 0.5° land grid-cell, for the different CFTs, which can be further partitioned into irrigated and non-irrigated fractions. The remaining cell area is attributed to natural vegetation. This approach allows for dynamic simulations of land-use change within cells, and therefore spatially explicit estimations of the effect of such changes on biophysical and biogeochemical processes, and also climate.

LPJmL produces output data for several variables, which constitute the information base for assessing the effects of climate, land-use, and other process scenarios on the global biosphere. Table 2.2 lists some of the main LPJmL outputs that are typically used to evaluate the outcome of different scenarios, either directly or in further analysis steps.

2.3 Note on terminology and procedure

The important concepts and processes relating to climate and vegetation modelling have been explained on the basis of LSMs and DGVMs. In order to keep the discussions in the remaining thesis chapters clear and simple, the more generic term 'vegetation models' is used to refer to LSMs, DGVMs and other models with similar purposes and structures.

Due to the author's affiliation with the Potsdam-Institute for Climate Impact Research, which develops and maintains the LPJmL model, the use-case scenarios will be performed using LPJmL model results. However, it should be noted that the system is independent of varying process implementations in different vegetation models, and has been designed to be universally applicable.

Chapter 3

Benchmarking

This chapter introduces the idea of benchmarking and discusses components of a benchmarking framework and system for the vegetation model community. Special attention is paid to the discussion of a comprehensive set of metrics for model performance evaluation.

3.1 Benchmarking - a definition

The idea of benchmarking has been ascribed to Robert C. Camp (1989), a former employee of Xerox, who defined it as 'the search for industry best practices that lead to superior performance.' This quote shows that the origin of benchmarking lies in the corporate world, focusing mainly on improving a company's strategy, processes, and consequently market position. Improved business performance should be achieved by systematically comparing a company's key figures and processes to those of the industry leader. Subsequently, strengths and weaknesses are identified and analysed, whereupon best-practices can be adopted (Fifer, 1988). Essentially, this suggests that in order to become and remain competitive, a company must not solely focus on performance differences between internal processes or compartments,

but needs to continuously observe and compare itself with its competitors (Venetucci, 1992).

Meanwhile, benchmarking has been applied in many different disciplines and there exists a multitude of procedures for achieving process or service performance improvement. However, there are two fundamental, application-independent questions, which form the basis of every benchmarking approach (Venetucci, 1992):

- 'What strategies or processes should be compared?'
- 'Against whom or what should they be compared?'

On the basis of the above-mentioned characteristics, following generic definition of benchmarking is proposed for the purpose of this thesis:

Benchmarking is a management tool that is aimed at evaluating and improving the strategy, the performance of processes, or other aspects within an entity by means of comparison with a purposefully defined standard, i.e. a benchmark.

Subsequently, the benchmarking framework and definition of the term benchmark will be further specified to fit the distinctive requirements of vegetation-model benchmarking.

3.2 Benchmarking vegetation model

Good scientific practice within and among modelling groups should comprise a standardised and widely accepted quality management system. Comprehensive benchmarking in the vegetation modelling community would require a perpetual performance comparison of models, but should also include evaluation and analysis of varying process performances within a model. Being aware of the strengths and weaknesses of vegetation models helps to establish performance rankings with regard to different applications, and fosters targeted improvement. In the last decade, several papers promoted a common effort to establish a standardised benchmarking framework and system for LSMs (Abramowitz, 2005; Abramowitz et al., 2008; Blyth et al., 2011; Abramowitz, 2012; Luo et al., 2012; Best et al., 2015) and DGVMs specifically (Kelley et al., 2013). Moreover, the International Land Model Benchmarking (ILAMB) project, initiated in 2009, hosted two international meetings, in 2009 and 2011, with the stated goals being to (Oak Ridge National Laboratory, 2010):

1. 'develop internationally accepted benchmarks for land model performance
2. promote the use of these benchmarks by the international community for model intercomparison
3. strengthen linkages between experimental, remote sensing, and climate modeling communities in the design of new model tests and new measurement programs, and
4. support the design and development of a new, open source, benchmarking software system for use by the international community'

Kelley et al. (2013) remark that there is no established standard practice in the DGVM community that systematically tracks the effect of model developments on performance. Furthermore, Prentice et al. (2015) argue that due to large differences between LSMs of the latest generation, especially with regard to projections of carbon- and water-cycle aspects, benchmarking should be a required component of model development. The work at hand picks up on this suggestion and uses the ILAMB statement of purpose as a general guideline for the development of a benchmarking system for vegetation models.

3.3 Benchmarking system framework

Luo et al. (2012) propose a framework of 4 sequential steps to assure comprehensive benchmarking of the land-component in ESMs (see section 2.1). Generally, the framework comprises 1) the identification of sensible model-aspects for evaluation, 2) the selection of suitable benchmarks as references for performance, 3) a scoring system, including a-priori thresholds, to measure relative model performance, and 4) the diagnostic assessment of model strengths and weaknesses for targeted model improvement. In the following sections, the four different steps of a comprehensive benchmarking process for vegetation models are explained in more detail, and the requirements for a benchmarking system are discussed. The emphasis is on suitable metrics for a scoring system, which have only been treated secondarily in relevant publications, in contrast to qualified datasets and benchmarks, which are discussed in depth by Luo et al. (2012), Kelley et al. (2013), (Abramowitz, 2012) or Best et al. (2015).

3.3.1 Model aspects for evaluation

Deciding which model aspects should be evaluated is always the first step in a benchmarking process for vegetation models. A straightforward guiding principle is to select properties that are common to all models, and to evaluate them on various spatial and temporal scales (Luo et al., 2012). As mentioned in the previous chapter, an important application of vegetation models is to assess the interplay of climate and vegetation dynamics, and the effects on connected biophysical and biogeochemical processes. Their common properties are, among others, a representation of photosynthesis, a description of carbon allocation to different carbon pools, decomposition rates for litter and soil organic matter, and energy and water fluxes (see Table 2.1). It is vital that such fundamental processes are represented adequately in vegetation models. However, whether a certain model aspect is represented well can be judged on the basis of different preconditions (Abramowitz, 2012; Best et al., 2015), for example: How does an aspect perform compared to the same aspect in another model? How does an aspect perform with respect to a particular application? How effective does a model use the information provided to it by the input data? Once the intention of a benchmarking application is defined, appropriate reference datasets and benchmarks for the performance assessment can be selected.

3.3.2 Reference data and benchmarks

The second step is to choose reference datasets and benchmarks that are suitable to evaluate the model aspects identified beforehand. Reference datasets are needed as the basis of comparison for model results, and include observations, like satellite images or data from measuring stations, and other data products, possibly derived from observations (Table 3.1). Luo et al. (2012) suggest that a reference dataset used for benchmarking should be objective, effective and reliable. This means it is ideally derived from independent observations, reflects fundamental processes of the system and exhibits low uncertainty. Thus, if a certain data product has been derived with the same approach implemented in the models to simulate the corresponding property, it does not fulfil the criterion of objectivity. Furthermore, site data that are used for calibrating or driving the models in the same grid cell should be excluded.

Table 3.1: Examples of potential sources for reference datasets.

Description	**Type**	**Model aspects**	**Source**
EMDI NPP	Site measurements	Carbon fluxes	Olson (2001)
FluxNet data	Site measurements	Ecosystem fluxes	Oak Ridge National Laboratory Distributed Active Archive Center (2015)
MODIS data	Satellite data products	Vegetation dynamics, ecosystem fluxes	National Aeronautics and Space Administration (2015)
River discharge	Site measurements	Water fluxes	Dai et al. (2009)

The usage of the term 'benchmark' in the vegetation model community seems to be inconsistent. Some authors equate measurements, or other reference datasets, that the models are compared with directly, with benchmarks or benchmark datasets (e.g. Luo et al., 2012; Kelley et al., 2013). In the context of this thesis, a more target-aimed definition is applied: a benchmark is the reference for a pre-defined performance expectation that is derived from the analysis purpose (e.g. Abramowitz, 2005; Best et al., 2015). This definition is illustrated by the following example: It is assumed that a complex model, which uses several meaningful variables to describe a certain process, performs better than a simple model, which uses fewer or none, when compared against observations. In this case, it makes sense to define the level of performance of the simpler model, perhaps expressed in terms of an error metric, as the benchmark.

Benchmarks that have been used in vegetation-model benchmarking applications include a statistically-based artificial neural network (Abramowitz, 2005), different regression models (Abramowitz et al., 2008; Abramowitz, 2012), and randomised datasets, constructed by bootstrapping observational data sets (Kelley et al., 2013). Another option is to set the performance of another model, perhaps an older model version or a well-accepted model, as the benchmark. However, this approach has been described as 'weak', since it is hardly distinguishable from an ordinary model comparison, and it does not rule out the possibility that all participating models are poor performers or that they are already within observational uncertainty ranges (Best et al., 2015).

3.3.3 Scoring system of metrics

A collection of numerical metrics form the basis of a scoring system that can be used to express the performance of a model relative to the benchmark level in meaningful figures. The choice of metrics depends on the analysis purpose, and determines the success of targeted model improvement. In this section, a set of readily understandable and frequently used metrics that are suitable for a comprehensive scoring system will be discussed. In order to structure the discussion of the scoring system, selected metrics have been divided into three general groups (Table 3.2): Metrics that can be used (1) to inspect and compare benchmark, model and reference value-distributions, (2) to measure the deviance, and (3) to scrutinise the similarity or agreement between models and reference data.

The first group includes data mean, standard deviation, skewness and kurtosis. Latter metrics characterise the shape of a distribution, and are therefore well-suited, if the similarity of benchmark, model and reference value-distributions should be compared.

The second group of metrics is often employed in model evaluation, and includes measures of the model bias, i.e. the tendency to over- or underestimate reference data, and the error magnitude, i.e. the absolute or squared difference between model results and reference data. They are all based on mean differences, yet vary in the treatment of sign and magnitude of individual errors, and therefore require different interpretations. Some well-established representatives of this group are the mean bias (MB), the mean absolute error (MAE) and the root mean square error (RMSE).

Table 3.2: Description of metrics for a scoring system.

	Name	Formula	Description
1. Value distribution	Mean	$\frac{1}{N}\sum_{i=1}^{n} x_i$	Arithmetic average of the values, possibly biased by outliers. x_i = i-th value of a sample, N = number of values in sample.
	Standard Deviation (SD)	$\sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i-\bar{x})^2}$	Measure of the average deviation from the mean in a given sample. $\bar{x}$ = sample mean.
	Skewness	$\frac{\frac{1}{N}\sum_{i=1}^{n}(x_i-\bar{x})^3}{SD^3}$	0 = normal distribution. Measures asymmetry in a data set: a negative skew has a long left tail (values concentrate on right); a positive skew a long right tail (values concentrate on left).
	Kurtosis	$\frac{\frac{1}{N}\sum_{i=1}^{n}(x_i-\bar{x})^4}{SD^4} - 3$	0 = normal distribution. 'Pointedness' of a data set. Positive domain: kurtosis increases with sharper peaks and longer tails in the distribution. Negative domain: kurtosis decreases with roundness and shorter tails.
2. Deviance	Mean Bias (MB)	$\frac{1}{N}\sum_{i=1}^{n}(M_i - R_i)$	$(-\infty, +\infty)$. Average signed error, possibly biased by cancellation. M_i = i-th model value, R_i = i-th reference data value. Also called: bias, model error.
	Mean Absolute Error (MAE)	$\frac{1}{N}\sum_{i=1}^{n}\lvert R_i - M_i\rvert$	$(0, +\infty)$. Mean error of unsigned values in data units, which is not effected by cancellation. Also called: mean error.
	Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{N}\sum_{i=1}^{n}(R_i - M_i)^2}$	$(0, +\infty)$. Similar to MAE, but possibly biased towards large errors.
3. Similarity	Coefficient of Determination (r^2)	$\left(\frac{\sum_{i=1}^{n}(M_i-\bar{M})*(R_i-\bar{R})}{\sqrt{\sum_{i=1}^{n}(M_i-\bar{M})^2}\sqrt{\sum_{i=1}^{n}(R_i-\bar{R})^2}}\right)^2$	$(0, 1)$. Proportion of the variance in a set of modelled values (M) that is explained by the variance in the reference data (R). $\bar{M}$ = mean of modelled values, $\bar{R}$ = mean of reference data.
	Adjusted Index of Agreement (d_r)	$\begin{cases} 1 - \frac{A}{cB} & \text{when } A \le cB \\ \frac{cB}{A} - 1 & \text{when } A > cB \end{cases}$ with $A = \sum_{i=1}^{n}\lvert M_i - R_i\rvert$, $B = \sum_{i=1}^{n}\lvert R_i - \bar{R}\rvert$, and $c = 2$	$(-1, 1)$. Positive values: Proportion of the total or average error in the reference data that is accounted for by the modelled values. Negative values: Proportion to which the average error of the reference data under-represents the actual average-error magnitude, i.e. MAE is larger than the average reference deviation.

Squared-difference measures, especially RMSE, have been criticised for their frequent misinterpretation and bias towards large errors, which is why Willmott and Matsuura (2005) recommend the utilisation of MAE for model evaluations. In contrast to that, Chai and Draxler (2014) state that the RMSE is more appropriate than the MAE when errors are normally distributed and if higher variation in the metric are desirable. However, using MB, MAE, and RMSE in concert can avoid misinterpretation of a single metric and simultaneously provides additional information for the evaluation. For example, if MB is 0 or very small, and MAE is large, one can infer that the calculation of a zero- or almost-zero bias was likely due to cancellation, rather than small errors. In addition, the difference between MAE and RMSE constitutes a measure for the variance in individual errors, since larger model-reference differences have a higher effect on RMSE. It should be noted that several inconsistent names are used for the metrics presented here as MAE and MB. The description of these metrics in Table 3.2 includes other names that have been found in the literature. For this work, the notation used by Jachner and v. d. Boogaart (2007) is adopted for consistency.

Besides providing metrics to evaluate the performance of a particular process variable in different models (e.g. NPP in Model A vs. NPP in Model B), a comprehensive scoring system should also enable the user to compare the relative performance of different process variables within a model (e.g. NPP in Model A vs. river discharge in Model A). Since the aforementioned measures of deviance depend on the unit of the variable, one needs to normalise these metrics, i.e. convert them to dimensionless measures, before a relative comparison is possible. This can be achieved by dividing the metric with a term that carries the same unit. Simon et al. (2012) suggest to divide MB by the mean of the reference data, since this minimises distortion of the original metric.

To enable a consistent interpretation of the three deviance measures, MAE and RMSE are also normalised by the reference mean. Table 3.3 contains formulae and descriptions of the discussed normalised versions of deviance measures.

Table 3.3: Normalised measures of deviance.

Name	Formula	Description
Normalised MB	$\frac{\frac{1}{N}\sum_{i=1}^{n}(R_i - M_i)}{\frac{1}{N}\sum_{i=1}^{n} R_i}$	Average model error, expressed as a fraction of the reference mean.
Normalised MAE	$\frac{\frac{1}{N}\sum_{i=1}^{n}\lvert M_i - R_i\rvert}{\frac{1}{N}\sum_{i=1}^{n} R_i}$	Average unsigned model error, expressed as a fraction of the reference mean.
Normalised RMSE	$\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{n}(R_i - M_i)^2}}{\frac{1}{N}\sum_{i=1}^{n} R_i}$	Root mean square model error, expressed as a fraction of the reference mean.

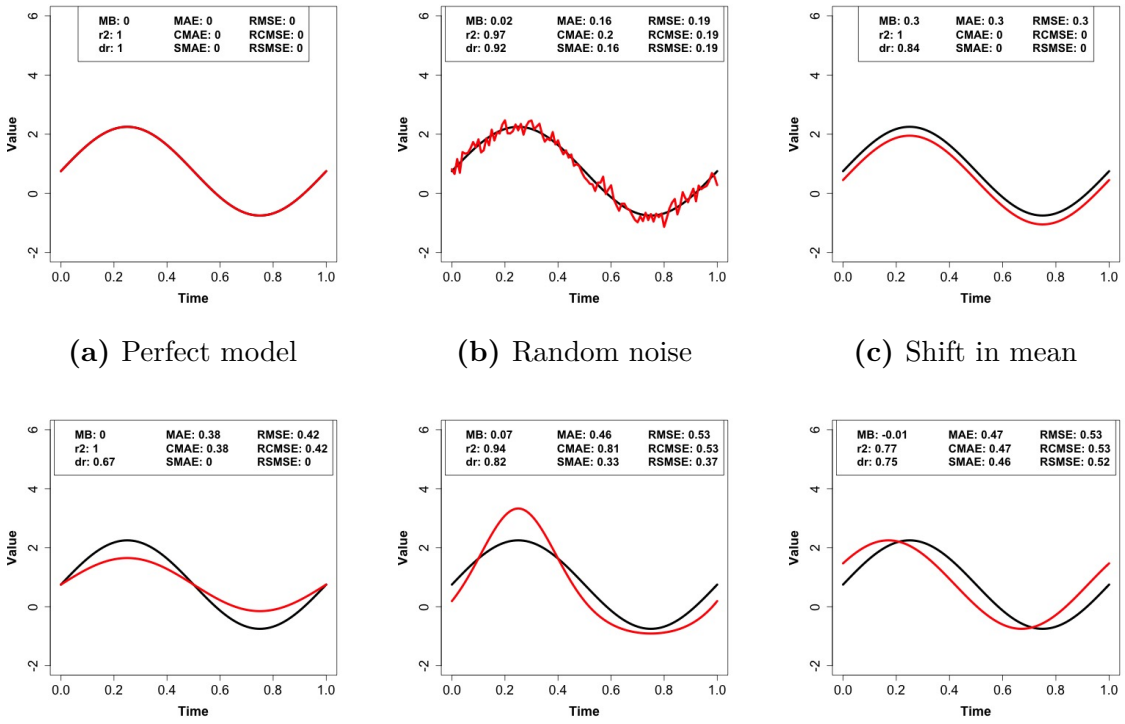
Additionally, one can reduce the effect of systematic bias from the analysis, e.g. due to a model-reference difference in mean or scaling, by employing transformed versions of MAE and RMSE. Table 3.4 lists formulae for the centred and scaled versions of RMSE and MAE, as suggested by Jachner and v. d. Boogaart (2007).

Table 3.4: Centred and scaled mean absolute error and root mean square error.

Name	Centred	Scaled	Description
MAE	$\frac{1}{N}\sum_{i=1}^{n}\lvert R_i - M_i - Md(R_i - M_i)\rvert$	$\frac{1}{N}\sum_{i=1}^{n}\lvert r_i\rvert$	MAE is centred by removing median of reference-model differences, and scaled by using reference-model residuals (r_i) instead of errors.
RMSE	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{n}\left(R_i - M_i - \overline{(R_i - M_i)}\right)^2}$	$\sqrt{\frac{1}{N-2}\sum_{i=1}^{n} r_i^2}$	RMSE is centred by removing mean of reference-model differences, and scaled by using reference-model residuals (r_i) instead of errors.

The third group of metrics are dimensionless measures of the agreement, or relationship, between model results and reference data, where 1 typically indicates a perfect fit. They are commonly derived from a normalised measure of deviance (e.g. Table 3.3), or a modification thereof. The term used for normalisation (i.e. the denominator) determines the interpretation of the metric. The widely used coefficient of determination (r^2) - the square of the Pearson product-moment correlation coefficient r - is popular due to its straightforward interpretation: it represents the fraction of the total variability in the model data that is explained by the variability in the observations (Table 3.2). Another commonly used metric is Willmott's index of agreement (Willmott, 1981), which is still often used or cited in its original form (e.g. Bennett et al., 2013), although it has already been revised twice (Willmott et al., 1985, 2012). Table 3.2 presents the latest, refined index of agreement (d_r), which is logically related to MAE. Willmott et al. (2015) claim that it is more widely-applicable, and more straightforward to interpret than the related, often-used Nash-Sutcliffe efficiency factor E (Nash and Sutcliffe, 1970), and more sensitive to differences between model and reference means and variances, than measures based on sum-of-squared errors (Willmott et al., 2015).

In order to demonstrate the behaviour of discussed measures of deviance (Group 2) and similarity (Group 3) in several possible scenarios of model-reference differences, artificial model and reference data has been produced and plotted (Figure 3.1), along with the metrics. It should be noted that in real-world applications it is more likely that combinations of the error-scenarios in Figure 3.1 will occur.



(a) Perfect model **(b)** Random noise **(c)** Shift in mean

(d) Shift and scaling **(e)** Nonlinear scaling **(f)** Time shift

Figure 3.1: Sensitivity of metrics to various error types between model (black line) and reference data (red line). Adapted from Jachner and v. d. Boogaart (2007)

Scenario a) - Perfect model: In the simplest case, i.e. the model exactly reproduces the reference data, all measures of deviance are 0, and similarity measures have the highest possible score of 1.

Scenario b) - Random noise: In case the reference data exhibits random noise from uncontrollable measurement erros, MB, MAE and RMSE scores are larger than 0. The difference in MAE and MB, i.e. small MB and large MAE, implies no over-

or underestimation of the reference data, but rather error cancellation. The small difference between MAE and RMSE scores indicates only little variance in individual error values. Centring or scaling these metrics does not result in any performance improvement, since the error is completely random. Similarity measures differ only slightly, with d_r being lower due to higher error sensibility.

Scenario c) - Shift in mean: A difference in the mean value of model and reference data results in MB, MAE and RMSE scores being exactly equal. MB being positive and equal to MAE implies systematic overestimation of the reference data by the model, and MAE and RMSE being equal indicates no variance in individual errors. Transformation of MAE and RMSE results in scores of 0, due to the purely systematic error. Thus, improvement of the model can be achieved by adjusting the mean. The higher error-sensitivity of d_r compared to r^2 is obvious in this scenario.

Scenario d) - Shift and scaling: If the systematic error between model and reference data is due to differing mean values and scaling, only the scaled MAE and RMSE, and the index of agreement d_r remain sensitive. This indicates that a parameter adjustment in the model can result in performance improvement with respect to the reference data.

Scenario e) - Nonlinear scaling: In case the model error is due to non-linear scaling, although purely systematic, scaled MAE and RMSE are less sensitive, but still imply a potential performance improvement by parameter adjustment.

Scenario f) - Time shift: Phase differences, for example due to different timing of processes in model and reference data, cannot be discerned by looking at differences in the measures of deviance or similarity.

Taylor diagram

This last section introduces the Taylor diagram (Taylor, 2001), a concise visual method for model performance assessment that combines metrics from each of the previously mentioned groups, i.e. distributional, deviance and similarity measures. It is based on the geometric relationship between Pearson's correlation coefficient, model and reference standard deviation, and the centred pattern RMSE[1] . Using the law of cosines, a diagram as shown in Figure 3.2 can be constructed. Statistics for a model are represented by a point in the diagram, and can be derived from the point location relative to the origin (standard deviation), the reference point along the x-axis (centred pattern RMSE) and the radial line (correlation coefficient).

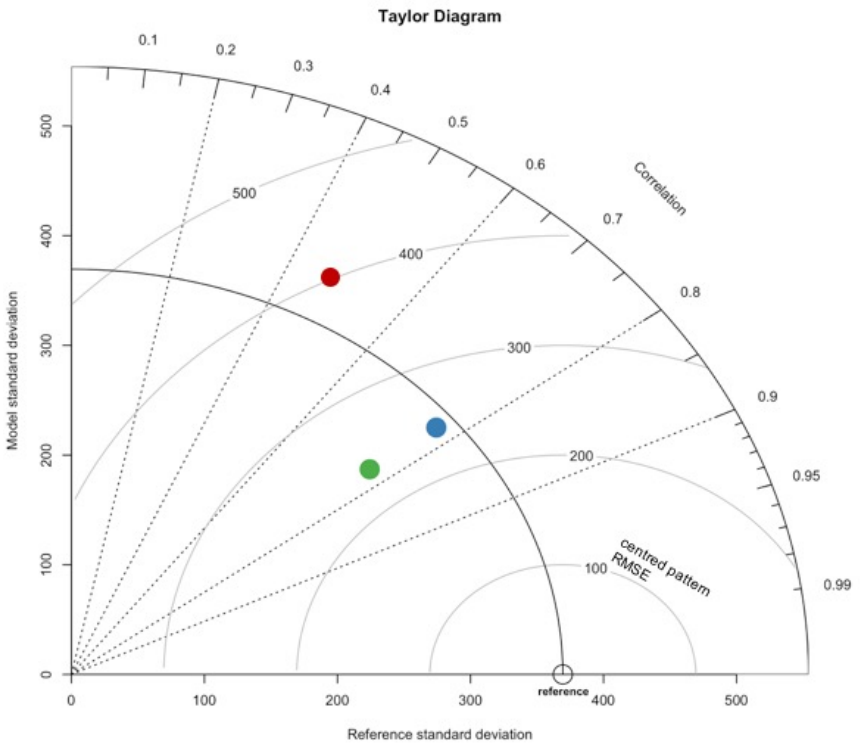


Figure 3.2: Example of a Taylor diagram: standard deviation in reference and model data (arcs around origin), correlation coefficient (radial lines) and centred pattern RMSE (arcs around reference point).

[1]The centred pattern RMSE has the same formula as the ordinary RMSE in Table 3.2, with R_i becoming (R_i - $\bar{R}$) and M_i becoming (M_i - $\bar{M}$.)

The Taylor diagram is especially suitable for an initial evaluation of relative model performances. In case different processes, or the same process at different scales (e.g. global vs. regional) should be compared, metrics in the diagram can be normalised by the standard deviation of the reference. As a result, the reference standard deviation (point on x-axis in Figure 3.2) of all reference datasets will be at unity.

3.3.4 Targeted improvement and improvement tracking

Model improvement

Systematic analysis, preferably at different spatial and temporal scales, of the metric scores enables model developers to evaluate the potential for model improvement and to gain insights into the dominant error type that is present in a process, i.e. is the error of systematic or random nature. Systematic errors can often be addressed, at least to a certain degree, by parameter adjustments in the equations that are used to calculate a process variable.

Parameter adjustment can be achieved through a variety of calibration methods, ranging from simple regression analyses to a combination of inverse modelling and Bayesian methods (e.g. Hartig et al., 2012), which are especially useful if observational data for calibration is sparse.

Improvement tracking

An essential part of the quality control within a modelling group is the tracking of improvements by way of comparing model results with old and new parameter configurations in model equations. A clear and straightforward method for improvement tracking is the simultaneous display of statistics for original and adjusted model simulations in a modified version Taylor diagram (Figure 3.3).

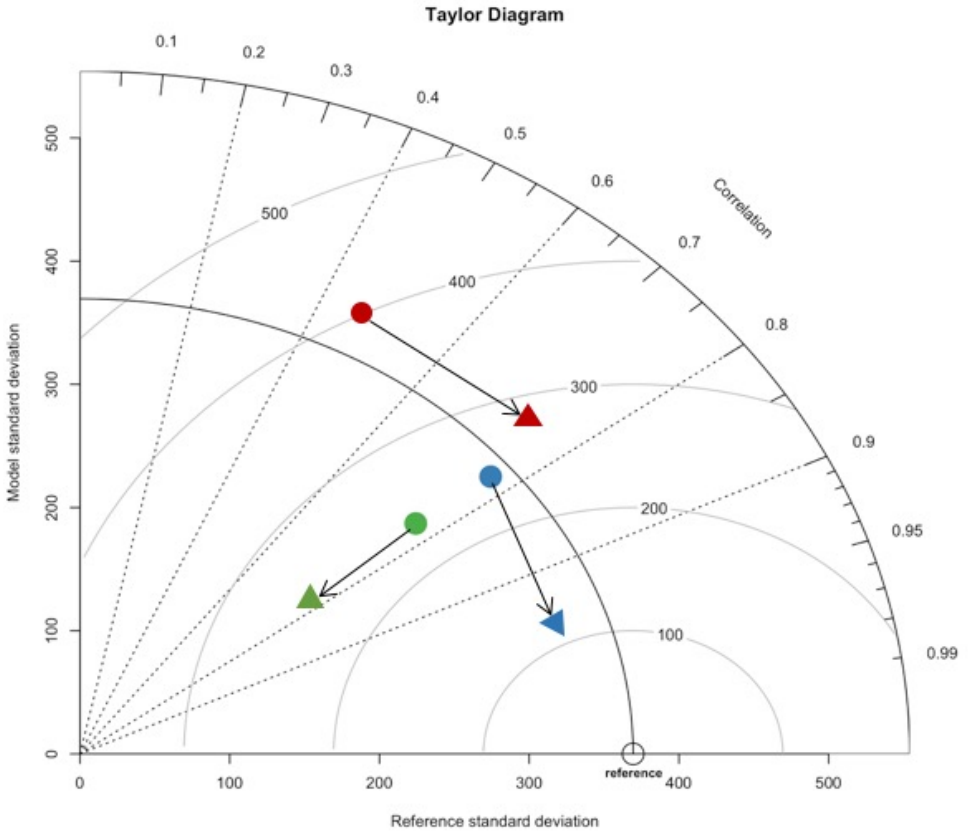


Figure 3.3: Example of a modified Taylor diagram for improvement tracking, as suggested by Taylor (2001).

In this diagram, henceforth referred to as improvement-tracker Taylor diagram, changes in performance are indicated by arrows, which connect the statistics for original model results (tail) with the statistics for adjusted model results (head). Therefore, length and direction of the arrow signify the amount of change in each statistic.

3.3.5 Components of a benchmarking system

Ideally, a benchmarking system that builds upon the four-component framework described in the previous sections (Figure 3.4) provides a host of well-accepted reference datasets, suitable for the analysis of different model aspects, a set of community-defined benchmarks, a comprehensive scoring system of complementary metrics, which supports targeted model improvement, and tools for model improvement tracking.

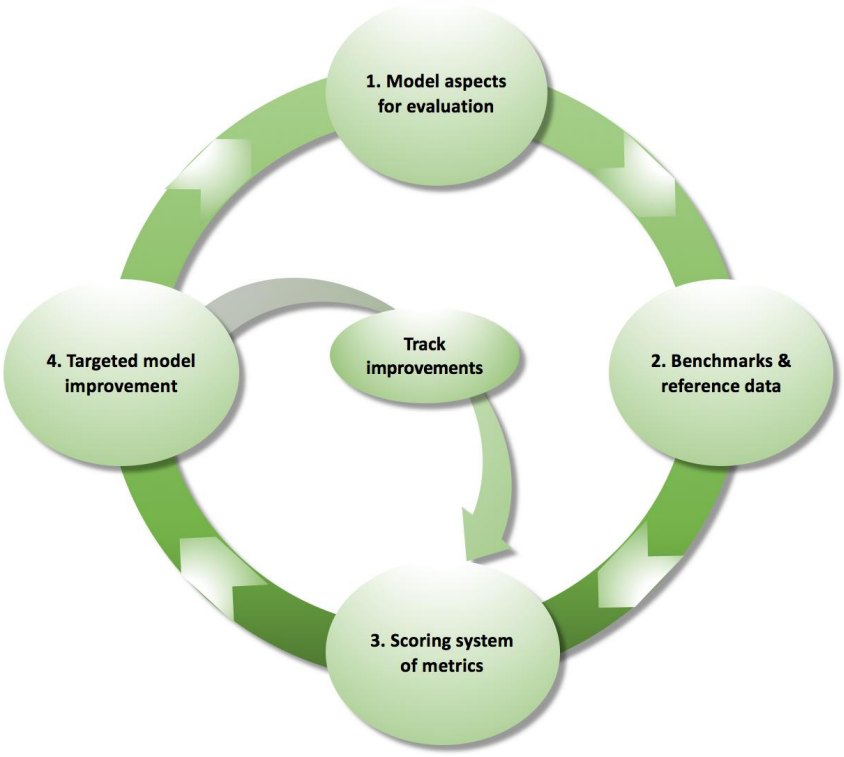


Figure 3.4: Summary of the vegetation-model benchmarking framework. Adapted from Luo et al. (2012)

The benchmarking system application that has been developed as part of this thesis is described in chapter 5. It provides solutions for framework components 3 and 4 (Figure 3.4), and includes dataset integration functionality. The identification of model aspects for evaluation, target-aimed benchmarks and suitable reference datasets needs to be a community-wide effort, advised by model developers.

Chapter 4

Spatial data integration

This chapter introduces a selection of concepts related to spatial data, before discussing methods that are relevant for the integration of model results and reference data in vegetation model benchmarking.

4.1 Concepts for spatial data integration

4.1.1 Geographic data: models and functionality

Dealing with geographic information implies two underlying questions about the data concerned: *What* does it represent and *where* has it been observed? The *what*-question asks about the nature and meaning of the data, i.e. what is the type (e.g. qualitative → nominal vs. ordinal, or quantitative → discrete vs. continuous) and purpose (e.g. unit or title) of the observation. The *where*-question refers to the direct (e.g. a range of or single coordinate pairs) or indirect (e.g. a known place or an address) location and boundary of the observation. These characteristics further determine how the data is appropriately handled and analysed. Geographic information systems (GIS) provide functionality to create, store, present, manage,

manipulate or analyse the spatial and non-spatial component of geographic data. In the remainder of this section, the two fundamental data-model groups used to represent geographic objects or phenomena in a GIS will be discussed.

Vector model

Real-world objects or phenomena that possess clearly definable boundaries in space (e.g. countries, rivers, buildings or streets) are primarily represented in the vector data model (Longley et al., 2010), which abstracts shapes and extensions in the form of point-, line- or polygon- (i.e. area) features. Points are the basic building block for other geometries and are described by a coordinate pair. Lines have at least two endpoints, and can possess several ordered vertices to describe more complicated shapes. Polygons are constructed in a similar manner, with at least two vertices and an identical start- and endpoint to ensure a closed area. Location specific, non-spatial data is attached to the features in an attribute table, in which columns represent attributes and rows correspond to different features.

The same real-world object or phenomenon can be represented by different feature types, depending on the scale at which a certain task or analysis is carried out, or a process of interest operates. For example, cities are usually represented as polygons or collections of polygons in cadastral maps, and as points on country-level weather maps. Data from measurement stations that monitor energy, water or carbon fluxes, such as FluxNet towers (Table 3.1), is typically represented as point data in the context of climate and vegetation model validation or benchmarking.

Raster model

In contrast to vector features, the raster data model places more emphasis on the *what*-question, rather than depicting real-world boundaries adequately (Maguire and Dangermond, 1991), and are therefore more suitable for the representation of phenomena that vary continuously over space (e.g. temperature, concentration, or elevation gradients). However, the raster model is also frequently used for thematic data, like land-use or soil data, that can be derived from satellite images or multiple other sources. A raster partitions space into an array of equally-sized cells, which can have different forms (e.g. quadratic, rectangular, triangular, hexagonal), depending on the application. Each cell usually contains an attribute value and has a location that is implicitly determined by its position relative to the raster origin and cell size. Furthermore, the smallest object dimension that can be resolved by a given raster is likewise determined by its cell size, which is therefore synonymously called spatial resolution.

4.1.2 Reference systems

In order to ensure exact and consistent positioning of objects or phenomena represented by vector or raster data on the Earth's surface, their coordinates must refer to and comply with a well-known or sufficiently specified coordinate reference system (CRS). The most common types of CRS are geographic and projected coordinate systems.

Geographic coordinate systems (GCS) are generally based on three-dimensional, mathematical approximations of the Earth's surface, usually in the form of an ellipsoid. The Earth's centre of mass constitutes the origin of coordinates, which are expressed in degrees of latitude, relative to the equatorial plane, and longitude, relative to an arbitrary prime meridian. Dimension and shape of an ellipsoid are de-

scribed by its semi-major (a) and -minor (b) axis, with the ratio (a-b)/a expressing the degree of flattening. Several ellipsoids, typically named after their originators, have emerged in history, either to obtain the best surface-approximation for a particular region or the global surface (e.g. WGS84). Additional rotation and scaling of the ellipsoid axes and shifting it with respect to the Earth's centre of mass, allows for even more adjustment to local conditions. The exact specification of an ellipsoid and last-mentioned adjustments is called a geodetic datum, and is obligatory for a complete description of a GCS, besides the coordinate units (e.g. decimal degrees) and prime meridian (e.g. Greenwich).

Although a GCS can precisely describe positions on the Earth's surface, there are several, primarily local to regional applications that require a plane, two-dimensional reference system. This is mainly due to the fact that geographic coordinates are not constant across the globe in terms of length, i.e. the distance between lines of longitude becomes subsequently shorter when moving north or south of the equator, angle and area units. Projected coordinate systems (PCS) transform the three-dimensional surface of a GCS to a plane surface (i.e. a mathematical projection), which always results in some sort of geometric distortion. However, transformations can be performed in such a ways that either length (equidistant projections), area (equal-area projections), angle (conformal projections) or direction (true-direction projections) are adequately preserved for a certain area. Besides the source geodetic datum, a PCS requires several other parameters (e.g. relating to projection, coordinate axes-shift, origin) for a full specification.

4.1.3 Metadata

Geographic datasets can easily be misused, diminish in value, or become completely useless, if they do not provide sufficient detail about their content, such as a title, attribute names, units, data quality, temporal extent and validity, creator, copyright or CRS. These textual and numerical descriptors are referred to as metadata, and form the context that turns any data into information. In order to ensure the compatibility, transparency, completeness and integrity of metadata, several standards evolved, most importantly the ISO 19115-1 schema 'Geographic Information - Metadata' (International Organization for Standardization [ISO], 2014). GIS include tools to create, view, manage and store metadata of vector or raster datasets.

4.2 Methods for spatial data integration

Before heterogeneous spatial data from multiple sources (e.g. Table 3.1) can be successfully integrated, several issues need be resolved. While these can include institutional, policy, legal and social aspects (Mohammadi et al., 2010), only the technical means for data integration that are offered by GIS and are relevant to a benchmarking system for vegetation models will be discussed. The prerequisite for applying any of the methods described below is a sufficient knowledge of the dataset properties, which are ideally described in the attached metadata.

4.2.1 Resolution, origin and orientation

Raster datasets can come in different resolutions (i.e. cell size), depending for example on the device that recorded the data (e.g. scanner-, or sensor resolution) or the application for which a dataset has been created (e.g. small vs. large scale). In a given set, one would ordinarily alter the cell sizes of the finer rasters to match the cell size of the coarsest raster. There are two principal ways to achieve this, assuming the datasets have the same CRS: aggregation or resampling by interpolation. In aggregation, a new grid is created, whose resolution is a multiple of the finer source grid. Cell values are then determined by an ordinary statistical operator (e.g. sum, mean, minimum, or maximum) on all source grid cell values that fall within the extent of a new, larger cell. If the coarser grid does not align with the finer source grid, resampling can be used, where new cell values are determined by interpolation. Common resampling techniques are the so-called nearest neighbour assignment, predominantly used for categorical data, and bilinear or cubic interpolation for continuous data.

Furthermore, rasters can have a distinct coordinate origin or orientation. Coordinate origins can be adjusted via resampling, while the orientation of a raster can be changed by 'flip' or 'rotate' functions, commonly found in GIS.

4.2.2 Difference in CRS

Raster or vector datasets with different CRS must be harmonised for the purpose of comparison or analysis. If the datasets have distinct GCS, a datum transformation must be performed. In many cases, there are several methods for the transformation of a particular GCS to another, and often the user has to choose the most suitable one from a list. Sometimes, no direct transformation method between two datums is available, which requires a two-step transformation via a datum for which the

transformation methods are known. When dealing with PCS, the process is more cumbersome, if the projections are based on different GCS, since this comprises a reprojection in addition to a datum transformation. PCS that are based on the same GCS, or datum, can be matched by a plane ('affine') transformation.
Resolution and CRS of a raster can usually be changed at the same time and requires the specification of a resampling method.

4.2.3 Spatial subsetting

When comparing different rasters with each other or with point data, one typically wants to restrict the analysis to cells or locations where values are present for all considered datasets. Furthermore, in some cases only cell or point values within a certain study area (polygon) should be considered for an analysis. For the purpose of creating a consistent and bounded set of values, GIS commonly offer extract tools that enable the user to retrieve a value subset for specific locations. For example, if a satellite image that is used for comparison with model output lacks some data in the area of interest, an extract-by-mask[1] tool can be employed on the model-output raster (Figure 4.1). The product is a raster that retains only the cell values that coincide with the mask raster, i.e. in this case the satellite image.

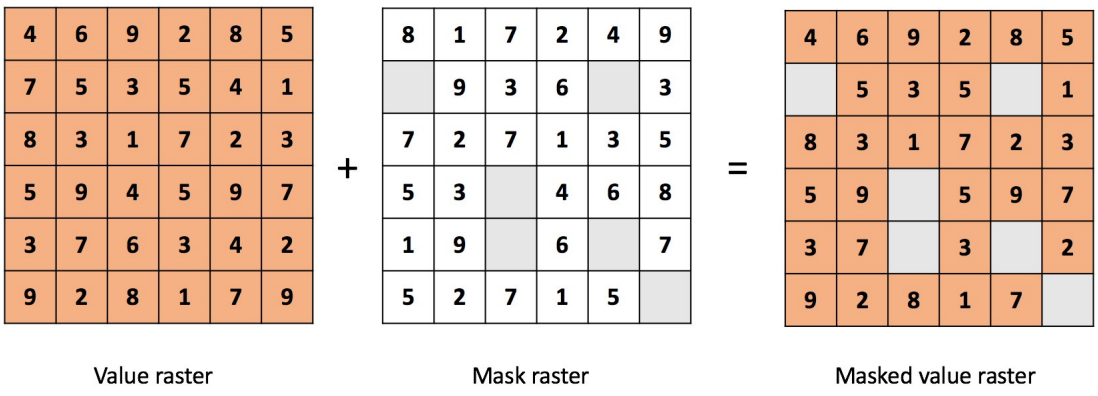


Figure 4.1: Cell value extraction by a mask.

[1]Tool names in different software products can vary from the descriptive names used here.

The extract-by-points[1] and extract-by-polygons[1] tools can be used to create a subset of raster values from point or polygon locations, respectively. In scenario a) in Figure 4.2, only values for cells that contain a point are returned, and values from cells that contain two points are returned twice. In scenario b), only values from cells whose centres are within the arcs of the polgon are returned.

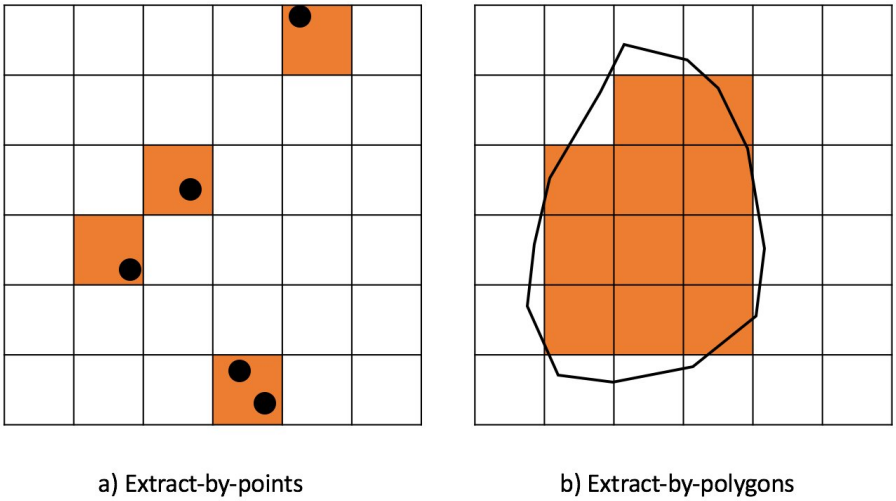


Figure 4.2: Cell value extraction by point (a) and polygon (b) features.

Chapter 5

Benchmarking system application

In this chapter, the general development process of the benchmarking system is outlined, and the object-oriented programming paradigm as well as the unified modelling language are introduced. The chapter is then concluded with an in-depth description of the design and implementation of the system in.

5.1 Application development process

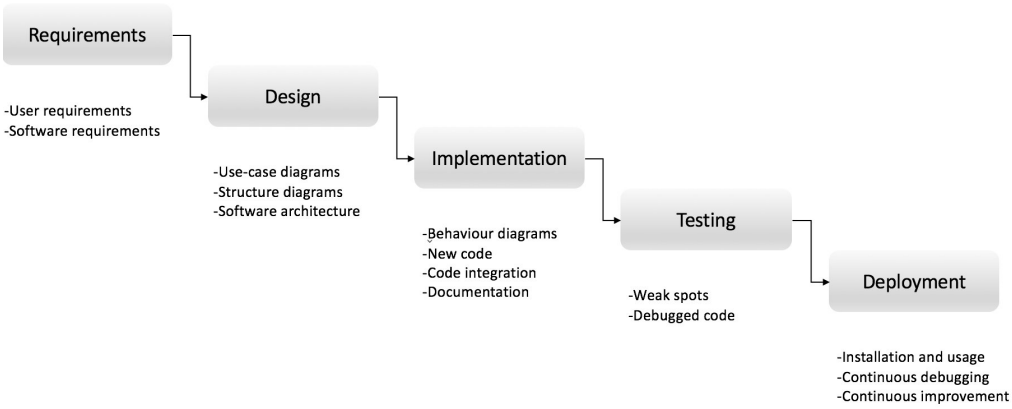
5.1.1 Software development life cycle

Depending on the scope and level of difficulty of the problem or application that a software solution should address, software development can easily become a complex undertaking with multitudinous tasks and related risks. In order to facilitate better scheduling, cost estimation, task sharing and risk evaluation, several, so-called process models have been developed that systematise the development process. Process models, also called software development life-cycles (SDLC), are commonly based

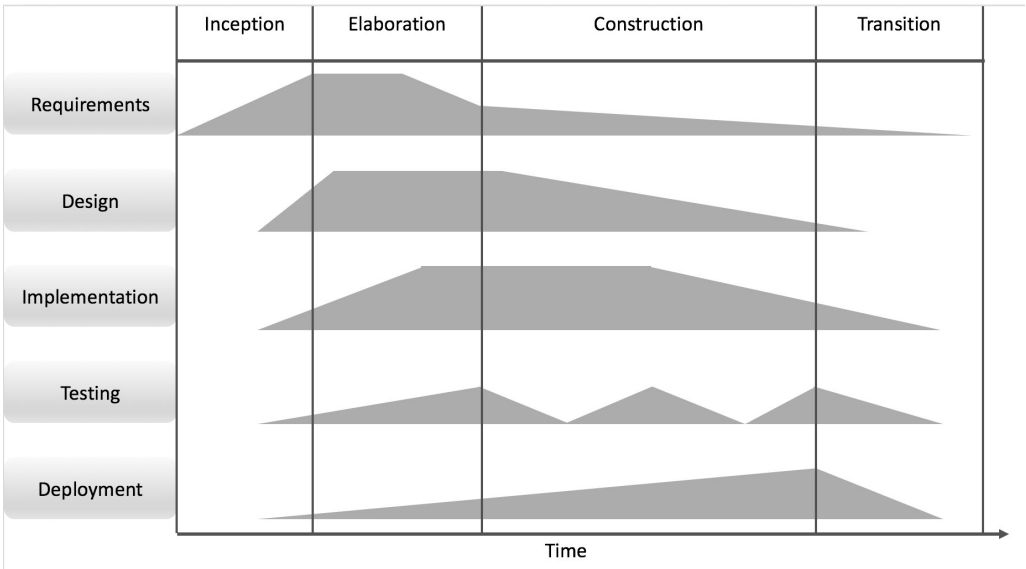
on definable development stages or phases, and can differ in several aspects, such as the chronological and logical arrangement of phases, emphasis or strictness.

For example, the so-called waterfall model (Royce, 1987) describes a sequence of distinct process stages, which are to be completed consecutively, and are usually not revisited (Figure 5.1a). In contrast to that, the unified process (UP), developed by Rumbaugh et al. (1999b), assumes overlapping process stages, with increasing or decreasing emphasis during four different chronological phases (Figure 5.1b). Since it is presumed that additional requirements can arise during the development process, the UP also allows for multiple iterations, i.e. going through one or more process stages repeatedly, especially during the elaboration and construction phase (Rumbaugh et al., 1999b). While the UP has been designed to be adaptable to the needs of a specific organisation or project, the relative extent of the four phases always should remain approximately constant: The inception phase is comparatively short and focusses primarily on the problem definition, the identification of key requirements and risks, and a rough cost estimation. It is followed by the slightly longer elaboration phase, during which the focus shifts to a more detailed requirement analysis and the design of the software solution. The third, and most extensive phase is the construction phase, and comprises intensive coding and testing, in addition to continued software design, albeit becoming decreasingly important. Deployment also begins in this phase, yet iterative changes in the software (i.e. incremental releases) still occur. Lastly, the transition phase includes the verification of the software product and can comprise demonstrations or trainings for end-users and administrators. A milestone can be assigned to each of the four different phases, which have to be passed in order to complete a stage and proceed to the subsequent one.

Numerous other approaches exist and the choice of a suitable process model depends on several factors, such as the applied programming style. For example, the UP has been developed for usage in concert with the object-oriented programming style. The design and implementation of the benchmarking system discussed in this thesis was accomplished with an object-oriented programming language and the development process was conducted in the fashion of UP (Figure 5.1b).



(a) Example of a 'waterfall' model for system development, including possible outcomes for each development stage.



(b) Possible scenario of software development according to the unified process.

Figure 5.1: Comparison of a linear (a) and an iterative and incremental (b) software development life cycle.

5.1.2 Object-oriented programming

The manner in which the actual design and implementation of a software is conducted, will largely depend on the chosen programming style or 'paradigm'. In turn, scope and complexity of the development project determine the suitability of a programming paradigm for the given tasks. The widely-used object-oriented paradigm is based on the definition of objects, which are abstractions of real-world entities, and their interactions. Objects are characterised by attributes and behave according to functions that the developer prescribes for them in a template, referred to as a class. Thus, objects of a certain class share the same attribute types and functions, but differ in attribute manifestation and function output. For example, cars share a set of common attributes (e.g. make, colour or motorisation) and functions (e.g. driving). A specific car model, with a specific colour, motorisation and driving behaviour (e.g. acceleration or maximum speed) therefore represent an instance, or object, of the car class.

Besides its implicit structuredness, the object-oriented paradigm provides a number of features that support a clear and efficient implementation of a programme. For instance, inheritance enables the developer to pass the characteristics and behaviours of an existing class to a new one, and to subsequently extend it by adding attributes or functions. Function overloading facilitates the usage of one designation for several implementations of a function, which is useful when the same or similar tasks should be performed on distinct objects. Encapsulation makes it possible to assign different access privileges to an object, or selected parts of it (attributes, variables, functions). This enforces object integrity and can lead to a more user-friendly operation of the software.

5.1.3 Unified Modelling Language

Consistent and intelligible communication between all parties involved in a software development process, including developers and clients, and thoughtful design are preconditions for a successful project outcome. Modelling languages for software engineering are support tools that can be used to conceptualise a software system or convey different types of information in a standardised visual, textual or any other form. The Unified Modelling Language (UML) was primarily developed for use with UP as an object-oriented modelling approach (see previous section) and provides a set of diagrams that are applicable for different purposes and design stages (Rumbaugh et al., 1999a). Since they allow for an effective communication between developers and users, and three of the most common diagram types will used in the following sections:

- *Package diagrams* provide a high-level perspective on the system, by combining logically related elements in so-called packages, and by showing the relationships between these packages.
- *Class diagrams* detail the structure (e.g. attributes and functions) of objects, and the relationships (e.g. inheritance) between different software system components.
- *Use-case diagrams* illustrate interactions between users and the software system in specific use-case scenarios.

5.2 Requirements

The requirements for the benchmarking system have been gathered and analysed by surveying members of the target group, i.e. in this case the developers and users of LPJmL at PIK, in several one-to-one conversations and during two presentations of the thesis project in front of a larger audience. Moreover, complementary requirements were derived from the ILAMB statement of purpose (see section 3.2) and several publications that discuss benchmarking frameworks and applications for vegetation models, especially Luo et al. (2012) and Kelley et al. (2013).

5.2.1 System qualities

Access and usability

In order to promote widespread acceptance and utilisation of the benchmarking system by the vegetation model community, the software should be easily accessible and comprehensible. One obvious solution is to develop the application in an open-source software environment that is widely used and facilitates customisation and extension of existent functionality. Moreover, usability can be enhanced by providing quick-access help pages and by hiding functions and classes that are not essential for user-operations.

Modularity and extensibility

Logical separation of functionality into modules, or packages, and the possibility to change and extend the software system's capabilities simplify its maintainability, and consequently prolong its lifetime. In addition, copyright issues concerning datasets that are to be included in the system can also be resolved by a modular design that comprises packages with different access rights.

5.2.2 Functional requirements

Harmonisation of datasets

Model output and datasets for benchmarking do not always have the same resolution, origin, orientation or CRS. It is therefore necessary to include methods for the adjustment of raster-specific properties (see section 4.2.1), and for the CRS transformation of vector and raster data (see section 4.2.2). Furthermore, these methods should ideally be implemented in such a way that users, which are not familiar with the details of transformation or resampling techniques, can readily employ them.
In Figure 5.2 the user compares several properties of two or more datasets, and the function must return (indicated by << *include* >>) a confirmation or a suggestion on how to proceed. Based on the suggestions, the user performs raster adjustments or CRS transformations on 1 or more of the datasets.

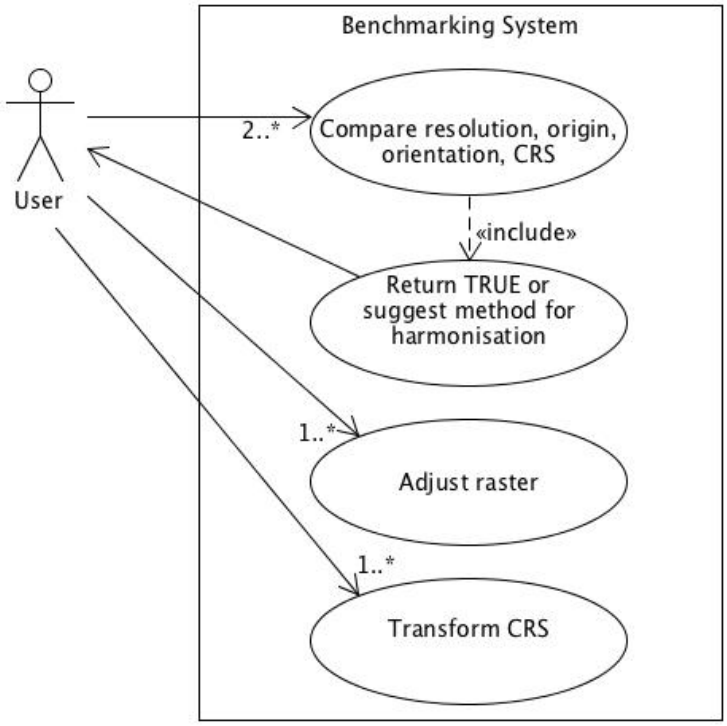


Figure 5.2: Use-case diagram showing the process of dataset harmonisation.

Extraction of cell values

In cases where comparison data is only available for certain raster cells or points, or when a benchmark analysis should be carried out for a specific area, usually a natural or political entity, spatial extraction methods are needed (see section 4.2.3). While vegetation models are always raster-based, comparison datasets for benchmarking can come in the form of raster (e.g. satellite images) or point data (e.g. site measurements). Raster cell values at certain cell- or point locations can be extracted using an extract-by-mask or extract-by-points method, respectively, whereas raster cell values within a certain region, for example a country or watershed, can be extracted using an extract-by-polygons method. Combining this functionality with a predefined polygon database for natural and political entities that are provided by the system allows for a consistent work-flow.

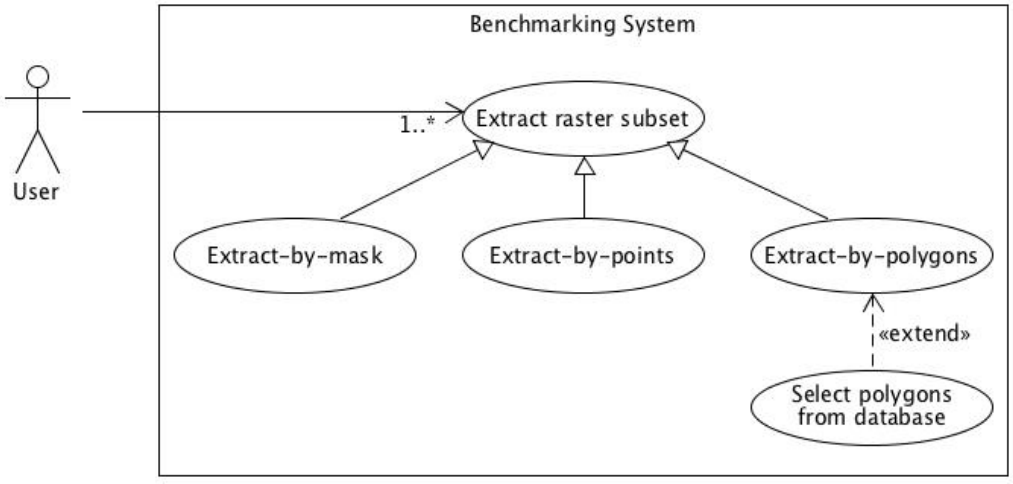


Figure 5.3: Use-case diagram of a raster cell-value extraction procedure.

In Figure 5.3, the user retrieves a subset of values from one or more rasters using the extract-by-mask, extract-by-points or extract-by-polygons method. In addition, the user can select a polygon for value extraction from a provided database (<< *extends* >> denotes an option).

Metadata

The prerequisite for the alteration, transformation and correct usage of datasets is the knowledge of associated metadata, such as units, spatial and temporal resolution, CRS, and others (see Section 4.1.3). Including a standardised routine and structure for metadata attached to data objects (Figure 5.4) can further contribute to the user-friendliness of the system, to data integrity and to the automation or semi-automation of processes, due to the dataset's capability to describe itself.
In Figure 5.4, the user sets the metadata of a dataset using a standardised routine, whereupon the dataset metadata can retrieved at any time.

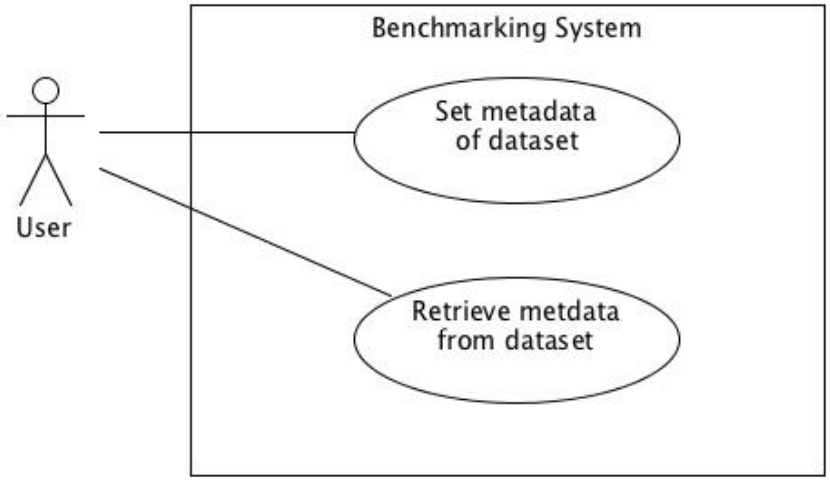


Figure 5.4: Use-case diagram illustrating the handling of metadata.

Statistical tools

Statistical tools for model performance assessment are an essential component of a benchmarking system for vegetation models (Figure 3.4). Besides providing a set of complementary metrics that can be used to analyse the data, and rank model or process performances with respect to a benchmark, the application should also comprise tools that help to track improvements in a model, like a Taylor diagram (section 3.3.3).

In Figure 5.5, the user computes metrics to evaluate model performance, and views a Taylor diagram to evaluate model performance improvement.

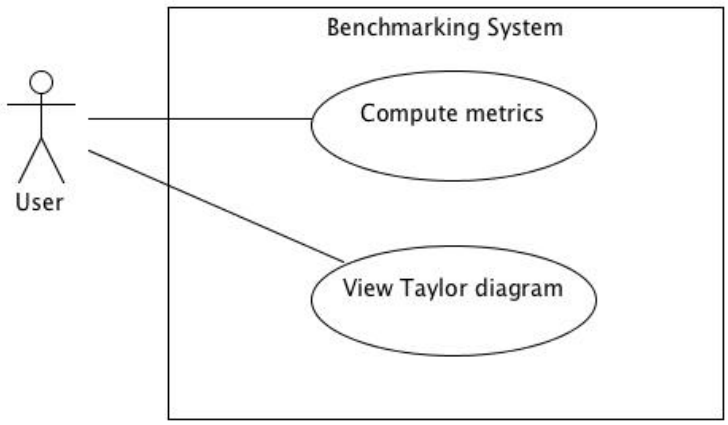


Figure 5.5: Use-case diagram of a model performance evaluation.

Read and write functionality

Another factor that has to be considered when dealing with datasets from multiple sources is that they often come in different file formats. NetCDF libraries and file formats are commonly used for raster data exchange, and an interface should be part of the benchmarking system. Additional, common file formats are the Geo-Tagged Image File Format (GeoTIFF) for raster data, or the simple text (TXT) or comma-separated value (CSV) files, often used for point data.

5.3 Design and implementation

5.3.1 Software framework

The *R* software environment (R Core Team, 2015a) has been chosen to implement the benchmarking system, due to its extensive features, wide distribution and free software licence (GNU General Public Licence). It is an implementation of the statistical language *S* and enables interactive operation through a command-line interpreter. Basic data objects include vectors, matrices, arrays, data frames and lists, but users can customise the working environment by creating additional object classes and functions. Comprehensive extensions can be written and added to *R* in the form of packages using, for example, the advanced object-oriented features of the *S*-language version 4 (*S*4 described in Chambers (1998)), comprising class definition with inheritance, function overloading through generic functions, and, at least to some degree, encapsulation. Furthermore, packages with a variety of custom functionalities, such as specialised statistical or spatial data analysis, can be obtained from different repositories and employed in the development of a custom extension. In the following section, *R*-packages that have been used for the implementation of the benchmarking system application are described.

5.3.2 R-Packages

raster package

The **raster**-package (Hijmans, 2015) provides classes in the raster format and includes functionality to manipulate the extent, resolution and origin of raster objects, to define and change CRS, and to extract raster cell values. Furthermore, the package supports the handling of very large raster files by reading, processing and writing them in chunks.

sp package

The **sp**-package (Pebesma and Bivand, 2005; Bivand et al., 2013) contains classes for point, line and polygon vector geometries, and several utility functions for, amongst others, spatial selection and sub-setting.

ncdf and ncdf4

Both packages provide an interface to the platform-independent NetCDF libraries and array-oriented file formats, which can be used to exchange raster datasets and corresponding metadata. NetCDF is an open standard (Open Geospatial Consortium [OGC], 2011) and many software products for raster data handling provide functionality to create, read, write or manipulate NetCDF files. Since the **ncdf**-package (Pierce, 2014b) does not support version 4 of the NetCDF file format, the **ncdf4**-package (Pierce, 2014a), currently only available for MacOS, is also included.

Hmisc, weights, qualV, plotrix

The **Hmisc**- (Harrell Jr et al., 2015) and **weights**- (Pasek et al., 2014) packages provide functions for data analysis, including simple weighted statistics, while **qualV** (Jachner and v. d. Boogaart, 2007) comprises similarity metrics and centred, scaled and normalised measures of deviance for model comparisons. Functions for graphical interpretation of model performance, such as the Taylor diagram, are contained in the **plotrix**-package (Lemon, 2006) .

RColorBrewer

RColorBrewer (Neuwirth, 2014) is a collection of functions for the creation of colour palettes, which are especially suitable for thematic maps.

methods

The **methods**-package (R Core Team, 2015a) is required for the S4 programming approach in *R* (see section 5.3.1), and provides classes, methods, generic functions and other programming tools for the creation of custom packages.

5.3.3 Benchmarking system architecture

The benchmarking system has been designed with due regard to the requirements outlined in section 5.2 and pre-existing functionality in *R*-packages that can be used to address these requirements. Packages that have been deemed useful or are required for the implementation of the benchmarking application in *R* are described in the previous section (5.3.2). In order to integrate pre-existing and custom-built functionality, two packages were created: **LandMark** and **PIKTools** (Figure 5.6). The **LandMark**-package comprises two spatial data classes, *LandGrid* and *LandPoint*, and functions for the creation, reading and writing, conversion, manipulation and analysis of their instances, i.e. data objects. *LandGrid* inherits the *RasterBrick*-class included in the **raster**-package, and *LandPoint* inherits the *SpatialPointsDataFrame*-class for point vector data, contained in the **sp**-package. Consequently, functions belonging to the *RasterBrick*- and *SpatialPointsDataFrame*-class, and most other functions in the corresponding packages, can be applied to *LandGrid*- and *LandPoint*-objects, respectively. This circumstance is implied by the $<< merge >>$-association[1] between **LandMark** and **raster**, and **LandMark** and **sp** in Figure 5.6. In addition, the **LandMark** package accesses functionality from the **ncdf**-, **Hmisc**- and **RColorBrewer**-packages.

The packages **qualV**, **weights** and **plotrix** are not shown in the diagram, since only a few functions, which do not depend on other contents of their source package, were needed. Thus, useful function code of these packages was simply copied to **LandMark** and annotated with the corresponding author and source. By doing this, **LandMark** is independent of possible changes in these packages and the web of relationships with external content remains manageable.

[1]The definitions of the package associations used in this context do not exactly conform with the UML specification (Rumbaugh et al., 1999b), since object-oriented programming in *R* differs from more traditional object-oriented languages.

Due to its lack of compatibility with operating systems other than MacOS, **ncdf4** is only used, if it is available in the working environment, and hence left out from the package diagram.

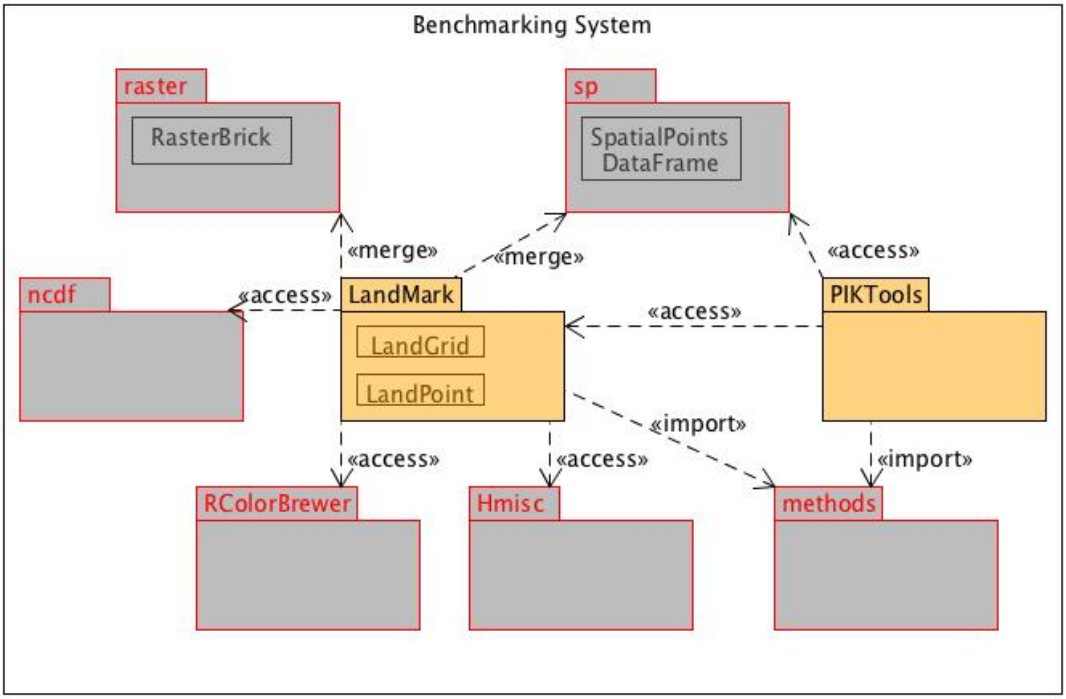


Figure 5.6: Package diagram showing the relationships between LandMark and PIKTools, and other integrated R-packages.

The **PIKTools**-package contains functions and datasets specific to LPJmL model results, and is designed to work in concert with **LandMark**, although independent operation is possible.

Both, **LandMark** and **PIKTools** import the **methods**-package (Figure 5.6), needed for S4-implementation of classes and functions. The $<< import >>$[1] designation specifies that the package is loaded and attached to the working environment in *R*, making its public functions and classes available to all other objects in the same environment. This is required for the usage of **methods** by other packages in older *R*-versions, and implemented for backward compatibility. In contrast to that,

$<< access >>$[1] denotes that functions and classes do not need to be attached to the working environment, and are accessed directly via so-called qualified names. These are indicated by a double-colon for public functions (e.g. 'package::functionName'), and a triple-colon for private functions (e.g. 'package:::privateFunctionName'), i.e. functions that are only accessible from within a package. Using qualified names is slightly less efficient than a package import (R Core Team, 2015b), but enables a clearer code, since all employed functions are labelled with their source package, and avoids potential naming conflicts.

5.3.4 LandMark

This section describes the main components of the **LandMark** package: a data type for metadata, two customised spatial data classes with corresponding utility methods, and methods for spatial data integration and statistical analysis that are useful for benchmarking.

MetaData data type

As mentioned in the requirements section (5.2.2), the benchmarking system application should provide standardised routines to manage and view metadata of data objects. In order to achieve this, the *.MetaData* data type has been created, and is employed as an attribute by the *LandGrid* and *LandPoint* classes, which will be described subsequently. The data type comprises a list of descriptor attributes (Figure 5.7) that conform with the suggestions of the NetCDF Climate and Forecast (CF) conventions for metadata (Open Geospatial Consortium [OGC], 2011), and facilitates a comprehensive description of the data, including source, type and application. Two important metadata attributes, *title* and *unit*, are not included in *.MetaData*, since they are already contained in the *RasterBrick* class, which the *LandGrid* class inherits (see next section).

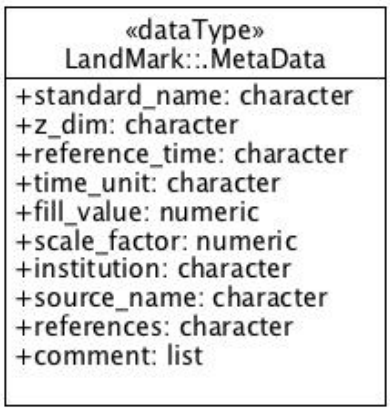


Figure 5.7: Attributes in the *.MetaData* data type.

LandGrid class

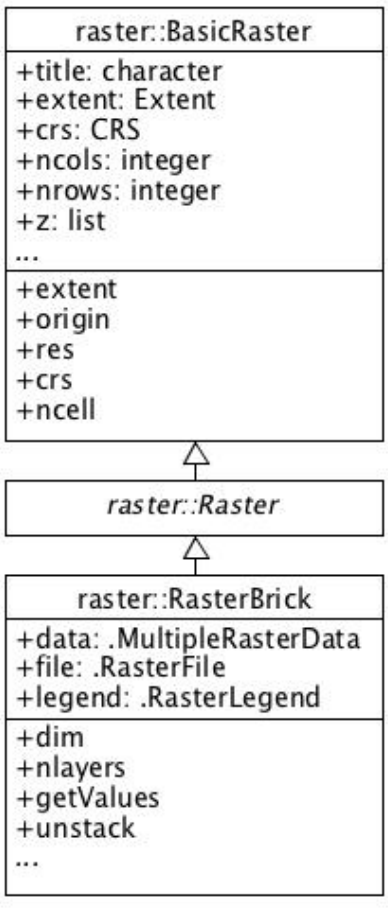


Figure 5.8: Class diagram of the *RasterBrick* class with some important attributes and methods.

The *LandGrid* class enables the user to correctly handle raster data, such as vegetation model output or satellite images, within the benchmarking system. It takes advantage of already existing functionality, by extending the *RasterBrick* class from the **raster**-package (Hijmans, 2015). An instance, or object, of the *RasterBrick* class can contain multiple data layers, making it possible, for example, to place a time series or separate layers for different characteristics of a variable in a single data object. Figure 5.8 shows the structures of *RasterBrick*: it inherits (denoted by the white arrowhead) attributes, which are shown in the upper sections of the boxes, and methods (lower sections) from the *BasicRaster* class through the *Raster* class template. The attributes of *BasicRaster* comprise several spatial (*extent, CRS*, etc.) and non-spatial (e.g. *title*) raster properties, which can be retrieved or changed with corresponding utility methods. *RasterBrick* extends *BasicRaster* by the *data* attribute (*.MultipleRasterData* type), which stores the actual raster cell data in a matrix, by the *file* attribute (*.RasterFile* type), holding information about the file from which the data was read, and by *legend* (*.RasterLegend* type), containing legend attributes for plotting (Figure 5.8).

Methods that are specific to *RasterBrick* can be used to retrieve or change cell values and properties of raster objects with multiple layers. For example, *dim* returns object dimensions including the number of layers, and *unstack* creates separate one-layer data objects from a *RasterBrick* object.

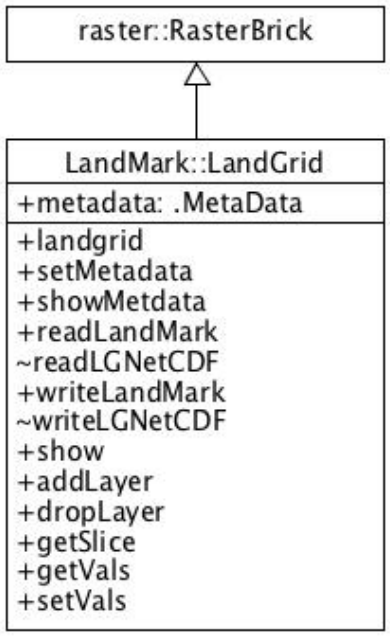


Figure 5.9: Class diagram of the *LandGrid* class.

The *LandGrid* class inherits all attributes and methods from *RasterBrick* and its parent classes, and further extends these by a metadata attribute of the previously described *.MetaData* type, and several utility methods (Figure 5.9): *landgrid* facilitates the creation of *LandGrid* objects from scratch, or by coercing instances of other classes, like *RasterBrick*. The methods *setMetadata* and *showMetadata* allow the user to enter and view the metadata of a *LandGrid* object, respectively. Raster data can be read into a *LandGrid* object from several file formats with *readLandMark*, while *writeLandMark* only permits to save or write data in NetCDF files. Read and write interfaces to NetCDF files are provided to *readLandMark* and *writeLandMark* by the private (denoted by the tilde) helper methods *readLGNetCDF* and *writeLGNetCDF*. The remaining methods enable the user to view different properties of a *LandGrid* object (*show*), to add, remove or extract selected layers (*addLayer, dropLayer, getSlice*), and to set or retrieve object values (*setVals, getVals*).

LandPoint class

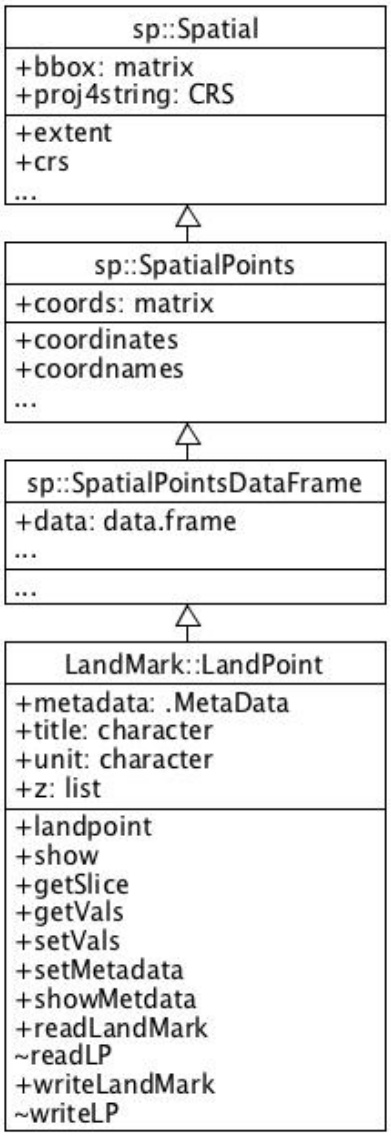


Figure 5.10: Class diagram of the *LandPoint* class.

Site data for benchmarking, for example flux measurements from stations at different locations, is handled by the *LandPoint* class, which inherits, directly or indirectly, attributes and methods from classes in the **sp**-package. Figure 5.10 shows the complete sequence of inheritance: *Spatial* merely specifies an extent (*bbox*) and a reference system (*proj4string*), while *SpatialPoints* includes a coordinate matrix for point geometry. *SpatialPointsDataFrame*, in turn, extends *SpatialPoints* by an attribute that can hold different types of data in separate columns, with each row referring to a point location in the corresponding coordinate matrix. The *LandPoint* class has a structure similar to *LandGrid*, but further includes the metadata attributes *title*, *unit*, and *z*, i.e. a list of names for the different data layers/columns in an object (Figure 5.10). Including these attributes in *.MetaData* would be redundant, since *LandGrid* already inherits them from other classes.

In order to simplify object handling in the benchmarking system application, all public method names in *LandGrid* (Figure 5.9) and *LandPoint* (Figure 5.10) match, except for the creator functions.

Data integration functionality

Owing to the inheritance of the *RasterBrick*- and *SpatialPointsDataFrame*-class by *LandGrid* and *LandPoint*, respectively, a variety of tools that are useful for spatial data integration become available within the **LandMark**-package. Figure 5.11 gives an overview of relevant functions in the **raster**- and **sp**-package, sorted according to different aspects of spatial data integration discussed in section 4.2.

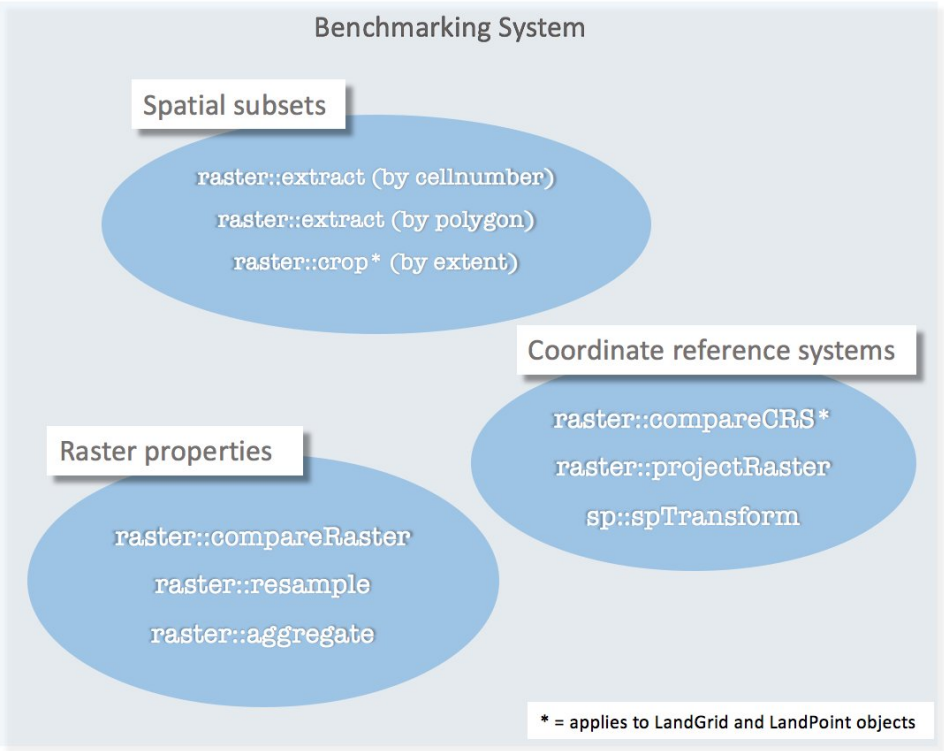


Figure 5.11: Relevant functions in the **sp** and **raster** package. (asterisk = applies to LandGrid and LandPoint objetcs)

In order to comply with the functional requirements mentioned in section 5.2, and to facilitate straightforward execution of the different processing steps involved in spatial data integration, i.e. transformation, raster adjustment and spatial subsetting, the *harmonise* function is provided to users of the **LandMark**-package. It utilises the functions listed in Figure 5.11 and executes or suggests them, through a message on the command prompt in a logical sequence (Figure 5.12).

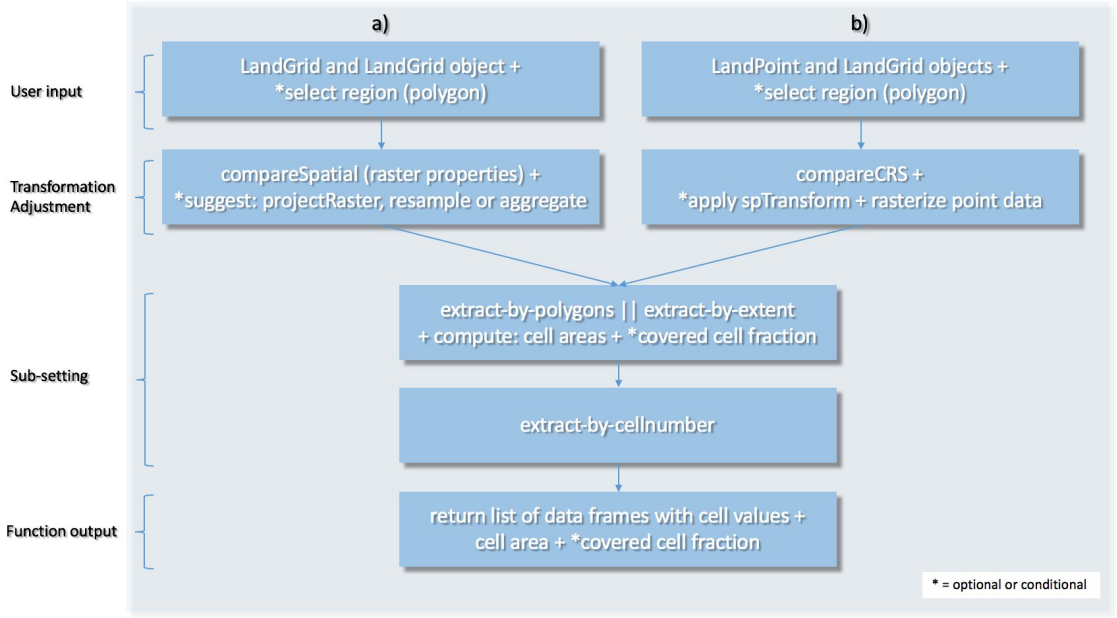


Figure 5.12: Processing sequences in the *harmonise* function (**LandMark**-package), for a) LandGrid-LandGrid and b) LandPoint-LandGrid.

The user supplies the model output and comparison dataset in the form of *LandGrid* or *LandPoint* objects to *harmonise*, and selects, if desired, a polygon that represents the region for which cell values should be returned. Since the integration of two raster datasets requires different processing than the integration of point and raster data, *harmonise* follows a different sequence of operation, depending on whether a) two *LandGrid*-objects, or b) a *LandGrid*- and *LandPoint*-object are supplied. In case a), shown in Figure 5.12, *harmonise* calls the helper function *compareSpatial*, which matches raster properties and determines whether a transformation (→*projectRaster*) or other raster adjustment (→*resample* or *aggregate*) is needed.

If no further changes are required, the procedure continues and cell values from the *LandGrid*-object that contains the comparison data are extracted, using the *extract*-by-polygons or *extract*-by-extent function, depending on whether values from a certain region (i.e. polygon) or from the whole extent of the dataset are needed. These functions already exclude no-data cells and return a data frame of values and

corresponding cell numbers, which refer to locations in the raster. In addition, cell areas are computed, which might be used as weighting factors in calculations of, for example, statistical measures. The *extract*-by-polygons function in the **raster**-package allows the user to choose whether the extraction is restricted to values of cells whose centres fall within the polygon arcs, or if the values of all cells covered by the polygon(s) are extracted, even if it is only a small fraction. In the latter case, covered cell fractions are approximated and attached to the data frame.

Cell values from the *LandGrid*-object that holds the model output are extracted using the cell numbers retrieved in the previous step. This is similar to raster masking (see section 4.2.3), since the cell numbers only refer to comparison-dataset cells that contain values.

Case b), i.e. a *LandPoint*- and *LandGrid*-object is supplied to *harmonise*, differs from a) in that only the CRSs of the two objects are compared and transformation of the *LandPoint*-object CRS is performed automatically, if they do not match (Figure 5.10). After that, the *LandPoint*-object is rasterized and converted to a *LandGrid*-object, with point values being assigned to cells into which they fall. This is done, since raster data operations are significantly more efficient for this purpose, which becomes important when large datasets are supplied. Consequently, the subsequent processing steps are performed as in case a).

The function returns the values extracted from both objects and corresponding cell areas together with, if applicable, covered fractions in a list of three separate data frames.

Analysis functionality

The list of data frames returned by *harmonise*, containing extracted values from the comparison dataset, model output and cell attributes (cell number, area and covered fraction), can be directly supplied to two customised statistical functions in **LandMark**: *computeMetrics* and *taylorDiagram*. First-mentioned function calculates a set of complementary metrics, including normalised, centred and scaled measures of deviance (see section 3.3.3), whereas the latter can be used to create Taylor diagrams (Figure 5.13). Optionally, values are weighted during the calculation of metrics, using either user-supplied factors or the cell areas and -fractions included in the list.

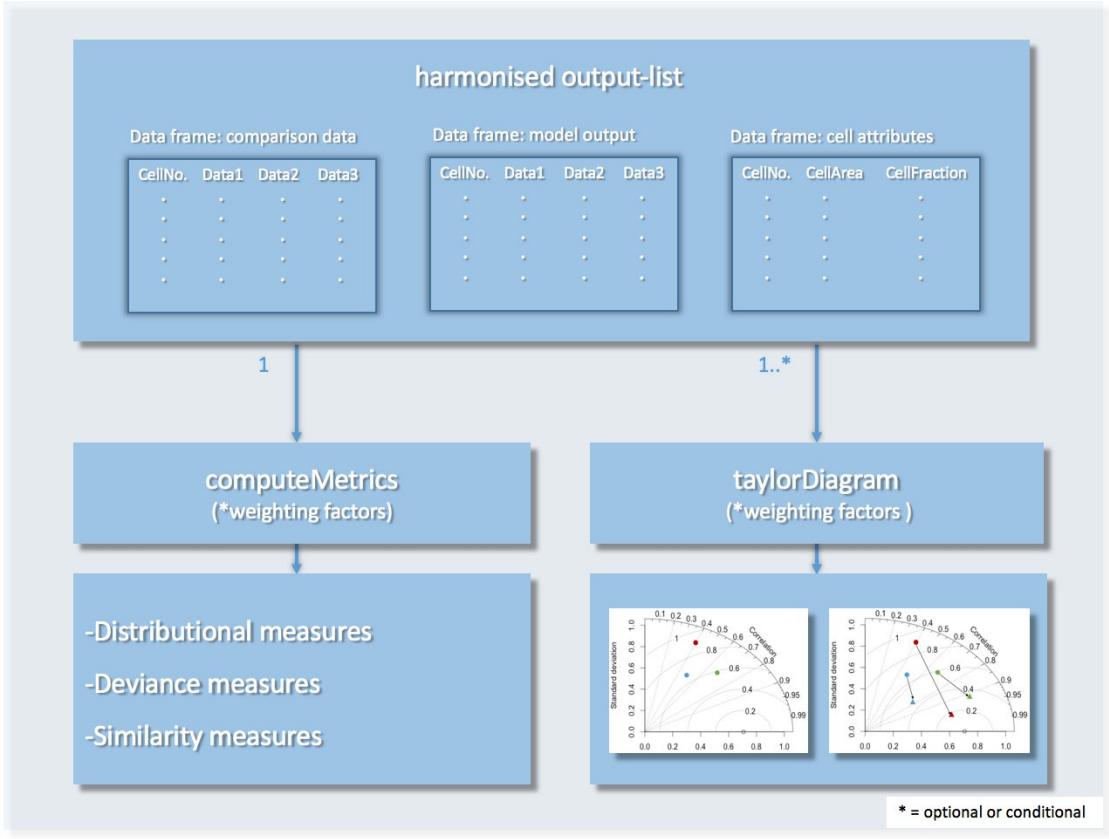


Figure 5.13: Application of statistical functions after dataset harmonisation.

In contrast to *computeMetrics*, multiple output-lists can be supplied to *taylorDiagram* (Figure 5.13) to facilitate a straightforward simultaneous performance assessment of different models or processes. Besides output-lists from *harmonise*, both functions accept *vector* and *data.frame* objects, which are commonly used in *R*.
The bulk part of the basic sub-functions in *computeMetrics* has been adopted from the **qualV**-, **weights**- and **Hmisc**-packages, while the *taylor.diagram* function in the **plotrix**-package constitutes the scaffold of *taylorDiagram* in **LandMark**. Existing code snippets and chunks were further extended to work seamlessly with the output of other functions in **LandMark** and to support straightforward tracking of performance improvements (see discussion on Taylor diagrams with arrows in section 3.3.3).

5.3.5 PIKTools

PIKTools can be regarded as an add-in package that extends **LandMark** with utilities for LPJmL model results and polygon datasets of natural and political entities. The content of the package will be described in more detail in this section.

LPJmL utilities

The utilities in **PIKTools** comprise a read-function for LPJmL results in binary data files (*readLPJBin*) and a database that contains raster templates and some auxiliary data relating to LPJmL (*lpjmlinfo*). Table 5.1 gives an overview and a description of the *lpjmlinfo* content. Most of the included datasets can be passed directly to some of the important functions in **raster** and **sp** (see Figure 5.11), making it easier for the user to perform certain tasks. For example, the *projectRaster* function in the **raster** package allows the user to supply a template raster with the desired spatial configuration, instead of supplying CRS and resolution specifically.

Other possible applications of the datasets included in *lpjmlinfo* are given in the comment field of Table 5.1.

Table 5.1: Description of the datasets contained in *lpjmlinfo*.

Dataset	**Description**	**Comment**
coords_full	Coordinate pairs referring to centres of land cells in a 'full' LPJmL raster	Use to create LandGrid or LandPoint dataset from scratch
coords_reduced	Coordinate pairs referring to centres of land cells in a 'reduced' LPJmL raster	Use to create LandGrid or LandPoint dataset from scratch
cellnumbers_full	Cell numbers referring to land cells in a 'full' LPJmL raster	Use to set (in *raster::setValues*) or extract (in *raster::extract*) cell values
cellnumbers_reduced	Cell numbers referring to land cells in a 'reduced' LPJmL raster	Use to set (in *raster::setValues*) or extract (in *raster::extract*) cell values
soildata	Binary vector specifying which cells in a 'full' LPJmL raster contain soil data (1) or no soil data (0)	Use to set (in *raster::setValues*) or extract (in *raster::extract*) values for cells with soil data
crs	Detailed specification of the CRS used for LPJmL in proj4s-format	Use as CRS argument in *projectRaster* or *spTransform*
epsg	EPSG code of the CRS used for LPJmL	Use as CRS argument in *projectRaster* or *spTransform*
full_raster	'Full' LPJmL raster. Cells corresponding to coords_full have value '1', everything else is NA	Use for masking (in *raster::mask*) or as a template in *raster::projectRaster* or *raster::resample* instead of supplying resolution or CRS of LPJmL raster
reduced_raster	'Reduced' LPJmL raster. Cells corresponding to coords_reduced have value '1', everything else is NA	Use for masking (in *raster::mask*) or as a template in *raster::projectRaster* or *raster::resample* instead of supplying resolution or CRS of LPJmL raster
soildata_raster	Soildata raster. Cells corresponding to soildata have value '1', everything else is NA	Use for masking cells (in *raster::mask*) without soil data

Polygon database

In addition to LPJmL-specific utilities, **PIKTools** provides polygon datasets that can be used, for example, in the previously described *harmonise* or *extract*-by-polygon function in order to extract cell values for certain regions. Three polygon datasets have been compiled from different sources and are stored in a database, called *geodata* (Table 5.2). Datasets contained in *geodata* can be viewed using the *showGeodata* function, and individual or batches of polygons for specific regions can be retrieved using the *getGeodata* function.

Table 5.2: Description of polygon datasets contained in *geodata*.

Dataset	**Description**	**Source**
gadm_adm0	Polygons for level 0 administrative areas (i.e. countries)	Global Administrative Areas (2015)
gadm_adm1	Polygons on level 1 administrative areas (i.e. provinces, Bundesländer, etc.)	Global Administrative Areas (2015)
grdc_basins	Polygons of major river basins	Global Runoff Data Centre (2007)

Since polygon datasets can be subject to copyrights and licenses, the database has been deliberately included in **PIKTools** and not in **LandMark**. The distinction between core- and add-in packages enables a clear separation of public and institution-specific content, and avoids potential legal issues.

5.3.6 Summary

The benchmarking system consists of the custom packages **LandMark** and **PIKTools**, and several other *R*-packages, most importantly **raster** and **sp**, from which a selection of suitable functionality is employed (see Figure 5.6). While **LandMark** provides core functionality useful for benchmarking, including tools for spatial data integration and statistical analysis, **PIKTools** contains LPJmL-specific functions and data, in addition to a polygon database. In order to extend the functionality and compatibility of the benchmarking system, customised add-in packages similar to **PIKTools** can be created in the freely available *R* software environment.

Figure 5.14 shows a benchmarking workflow according to the framework introduced in chapter 3 (Figure 3.4), using the presented benchmarking system.

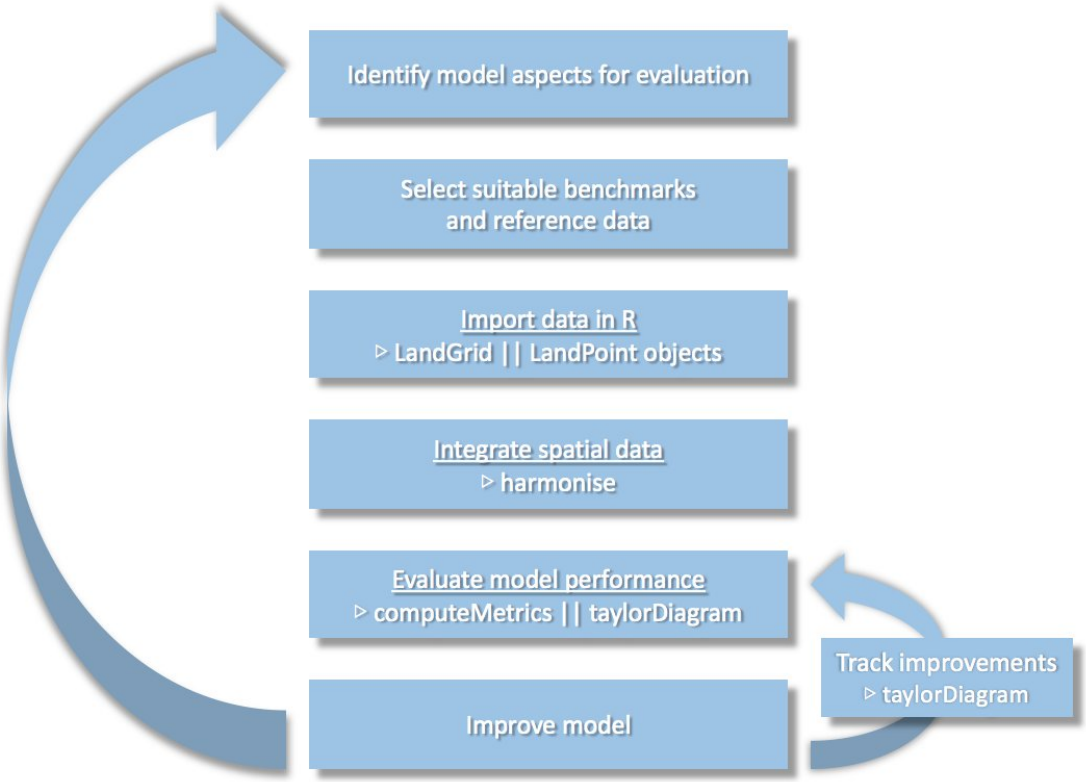


Figure 5.14: Benchmarking workflow using functions from **LandMark**.

5.3.7 Documentation

An important feature of a developer- and user-friendly application is a clear documentation of functions and corresponding code. The *R* software environment offers tools to create help pages for packages, which can be accessed by typing a command of the pattern `?function_name` in the command prompt. For every public class and function in **LandMark** and **PIKTools** a help page has been written, including usage explanations and examples. Code of classes and functions in **LandMark** and **PIKTools** has been documented using comments (starting with `#`) above or next to the concerned chunks or lines of code. In some cases, where undocumented code has been copied from other packages, proper documentation was not possible since the exact meaning of every chunk or line of code was not always clear. However, these code segments have been tested for proper functioning.

Commented code of classes and functions[2] that have been discussed in detail in the previous sections is listed in Appendix A. Corresponding *R*-help pages have been inserted before each function code.

[2]Enclosing the code of all functions in **LandMark** and **PIKTools** has been avoided, since it would fill several hundred pages. The complete code and corresponding help pages of the two packages can be requested at `jan.kowalewski@live.com`

Chapter 6

Use-cases

6.1 Grassland productivity

In section 2.2.1, two important output variables of vegetation models were mentioned: gross primary productivity (GPP), the rate at which carbon dioxide is converted to organic compounds through photosynthesis, and net primary productivity (NPP), the rate at which carbon dioxide is actually stored by plants - in the form of organic compounds - after autotrophic respiration. Since these organic compounds contain energy in the form of chemical bonds, NPP also represents the net rate at which energy is captured and stored by primary producers. Both rates are important components of the global carbon budget and interact with other processes of the climate system, which explains the particularly strong interest in the accurate estimation of these variables. Many approaches for modelling primary production have emerged with time, ranging from simple, empirically derived models (e.g. Lieth, 1972) to mechanistic descriptions of the photosynthetic process (e.g. Farquhar et al., 1980) and autotrophic respiration. Consequently, there is a large number of validation and benchmarking studies that aim to evaluate (e.g. Randerson et al., 2009; Blyth et al., 2011) or compare (e.g. Cramer et al., 1999; Adams et al., 2004;

Kelley et al., 2013) GPP and NPP model results of various models.

Grasslands are particularly interesting when it comes to the relationship of primary productivity and agricultural management practices. These ecosystems, which cover more than 25 % of the Earth's land area, carry large soil carbon stocks and have a high economic value when they serve as pastures for livestock that yield meat and dairy products. Furthermore, it has been suggested that the capacity of grasslands as carbon sinks, i.e. the time frame in which they act as a net carbon storage, could be increased and prolonged by well-considered management practices (Smith, 2014). Since approximately 70% of the global agricultural area is used as pastures, management practices that aim at maintaining carbon stocks and increasing sequestration of atmospheric CO_2 in grasslands could have a significant effect on the global carbon budget. Several grassland management options have already been implemented in the LPJmL vegetation model (see section 2.2.2) that allow for the evaluation of changes in grassland soil carbon stocks and primary productivity under different practices. Benchmarking LPJmL results of grassland GPP and NPP can help model developers to identify weaknesses and evaluate the potential for improvements in the model. In the following use-cases, an application of the benchmarking system described in the previous chapter is demonstrated by conducting a benchmark analysis with LPJmL grassland NPP and by evaluating LPJmL model improvements in simulating grassland GPP.

6.2 Use-case 1: Benchmarking grassland NPP

6.2.1 Grassland representation in LPJmL

Grassland ecosystems are dominated by herbaceous vegetation, mainly grasses, with a maximum of 40% tree or shrub cover, according to a UNESCO definition (White, 1983). In LPJmL, grasses are represented by two PFTs with distinct modes of photosynthesis and different responses of photosynthetic rates to elevated atmospheric CO_2 levels and changes in temperature: C3 and C4 grasses (see Table 2.2). Managed grasslands, or pastures, in LPJmL use the same parametrisation as the natural grasses, but with a daily allocation of carbon to vegetation, litter, soil and harvest-product pools. A daily carbon allocation scheme is necessary for the simulation of management practices, such as livestock grazing or mowing. GPP of natural vegetation and crop types is simulated using the mechanistic Farquhar photosynthesis model (Farquhar et al., 1980; Farquhar and von Caemmerer, 1982), and NPP is calculated by subtracting the simulated autotrophic respiration from GPP. Cell fractions for natural or managed C3 and C4 grasses are derived from a landuse input dataset and updated on an annual basis. Since managed grassland can be further subdivided into rainfed and irrigated cell fractions, each cell in LPJmL can contain up to 6 different grassland fractions.

Harvesting options

Management of pastures in LPJmL is represented by four harvesting options, which differ in harvest frequency, soil feedback and livestock unit (LSU) density, also called stocking density. For this use-case, only two management options are considered, the default (D) and the extensive, daily grazing (G_D) scheme.

In the default harvesting option D, stocking densities are ignored and harvest fre-

quency is dependent on grassland production, with a biomass-accumulation threshold set to 100 gC m^{-2}. After this threshold is exceeded, 50% of the grass leaf-carbon content is harvested and allocated daily to a harvest-product carbon pool. Furthermore, turnover of grass roots and remaining stubbles transfers carbon to the litter and soil pool on a daily basis. This management scheme has been designed to combine characteristics of both mowing and livestock grazing.

The extensive, daily grazing scheme G_D is described by a stocking density of 0.5 LSU ha^{-1}, and a feed demand per LSU fixed at 4000 gC day^{-1}, representing an average cow. Biomass removal by livestock grazing occurs daily during time periods where temperatures enable grass growth, and stops when the leaf biomass is 5 gC m^{-2}, or lower. The grazed biomass accumulated per day is distributed to different compartments with the following fixed proportions: 60% is emitted to the atmosphere, 25% is transferred to the fast soil litter carbon pool as manure, and 15% remains in the grazing animal.

Model protocol

LPJmL simulations of the D and G_D harvesting options were performed for grassland only, i.e. all other land cells have a value of 0, for the period 1901 - 2009, after running through a 390-year spinup to establish vegetation equilibrium. Monthly gridded climate data (Harris et al., 2014) for the variables temperature, precipitation, cloudiness and number of wet days were used to drive the model. Results for GPP and NPP are computed on a monthly basis (gC m^{-2} $month^{-1}$).

6.2.2 Reference datasets

In order to demonstrate the application of **LandMark** and **PIKTools** in a comprehensive benchmark analysis that involves both point and raster data, a collection of NPP site measurements and NPP satellite images were chosen as reference datasets.

EMDI NPP site data

Site data from the Ecosystem Model/Data Intercomparison (EMDI) database (Olson, 2001) are used for this analysis. The database contains annual NPP estimates from 1,014 sites in different biomes, covering the period 1931 - 1996, although the temporal coverage of individual sites within this time frame can vary considerably. Sites have been classified into 81 well-documented 'Class A' sites, and 933 'Class B' sites, with only little site-specific information. For this use-case, only the 'Class A' sites that are located in grasslands are used, leaving 31 sites for the analysis (Figure 6.1).

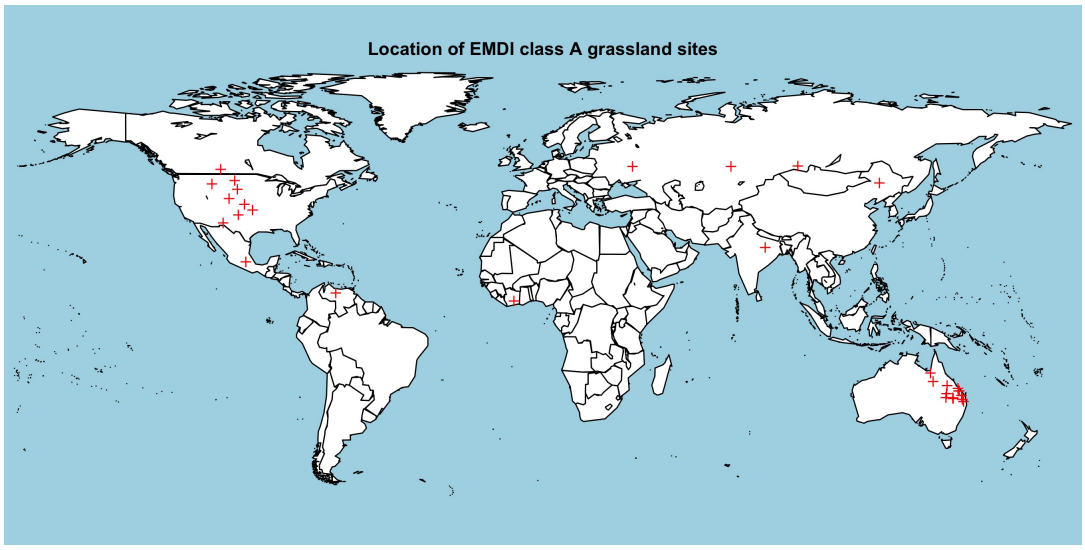


Figure 6.1: Locations of EMDI 'Class A' grassland-sites with annual NPP estimates.

The bias resulting from a concentration of sites in certain countries, in this case Australia and the U.S.A. (Figure 6.1), is a common problem that arises from the fact that site measurements from developing countries are often sparse or not available at all.

MODIS NPP satellite images

The Numerical Terradynamic Simulation Group at the University of Montana provides satellite data products of global primary production for the land surface from the EOS MODIS sensor (Running et al., 2004; Zhao et al., 2005). GPP and NPP are calculated using the MOD17 algorithm (Zhao et al., 2005), which follows the radiation use efficiency approach by Monteith (1972). It relates annual crop productivity to absorbed solar energy:

$$GPP = \varepsilon * APAR \tag{6.1}$$

$$PSN_{net} = GPP - R_{lr} \tag{6.2}$$

In equation 6.1, GPP is computed daily by multiplying the absorbed photosynthetically active radiation (APAR) with a conversion efficiency factor ε (gC MJ^{-1}) that is dependent on vegetation type, water-stress and temperature conditions. APAR (MJ day^{-1} m^{-2}) is a satellite product derived from the Normalized Difference Vegetation Index (NDVI) and photosynthetically active radiation (PAR). The daily net photosynthesis (PSN_{net}) in equation 6.2 is the difference between GPP and the 24-hour maintenance respiration in leaves and fine roots (R_{lr}). Annual NPP is then calculated using:

$$NPP = \sum_{i=1}^{365} PSN_{net,i} - (R_g + R_m), \tag{6.3}$$

where R_g is the annual growth respiration and R_m the annual maintenance respiration by all iving parts in a plant, except leaves and fine roots.

Therefore, MODIS NPP is not an objective, direct measurement, but the product of a particular modelling approach.

Annual MODIS NPP products (gC m^{-2} $year^{-1}$) are available as GeoTIFF files from the year 2000 onwards. The spatial resolution is 30 arc-seconds ($0.08\overline{333}°$) in the World Geodetic System 1984 (WGS84) reference system.

6.2.3 Benchmark: Miami model

The earliest attempt to model global terrestrial NPP was the well-known empirical relationship established by Lieth (1972), called Miami model, which relates NPP to precipitation and temperature:

$$NPP = min(NPP_T, NPP_P), \tag{6.4}$$

with

$$NPP_T = \frac{3000}{1 + exp(1.315 - 0.119 * \bar{T})}$$

$$NPP_P = 3000 * (1 - exp(-0.000664 * P)),$$

where $\bar{T}$ is the annual mean temperature (°C) and P the annual sum of precipitation (mm). In contrast to LPJmL, this model does not allow for negative NPP values and has a saturation value of 3000 g DM m^{-2} $year^{-1}$ (DM stands for dry (organic) matter). In spite of these restraints, several studies have shown that the Miami model reasonably estimates the global distribution of NPP (Adams et al., 2004). The Miami model is chosen as the benchmark in this use-case, in order to test how the mechanistic photosynthesis model employed in LPJmL performs against a simple empirical model. Since the Farquhar model is based on physical laws and the Miami model merely relies on temperature and precipitation as limiting factors, the basic assumption is that LPJmL will perform better than the Miami model.

6.2.4 Dataset preparation

Although LPJmL computes monthly NPP, total annual NPP (gC m^{-2} $year^{-1}$) is used as the basis of comparison, since EMDI and MODIS datasets are only available at this temporal aggregation level. Furthermore, two distinct time periods are used to compute the long-term average of total annual NPP of LPJmL and Miami model results, depending on the reference dataset: When EMDI site data serves as the reference, averages are computed for the period 1931 - 1996, and when MODIS NPP serves as the reference, the period 2000 - 2005 is used, which is often chosen as the baseline for scenario comparisons.

Before conducting the benchmark analysis, LPJmL and benchmark model results, as well as reference datasets have to be adjusted and harmonised. The specific steps that have been carried out to achieve this are described separately for the two reference datasets. Code used to prepare data and produce results listed in Appendix B.

Case 1: EMDI reference data

EMDI data tables contain information about the biome in which the measurement station is located, and grassland stations have been extracted prior to importing the dataset into *R* as a data frame. The data frame object containing EMDI NPP data and site coordinates is converted to a *LandPoint* object, using the *landpoint* creator and coercion function, and metadata is added using *setMetadata*. Since coordinates are already in the same reference system as LPJmL model results, i.e. WGS84, and NPP measurements are average total annual values in gC m^{-2} $year^{-1}$, no further conversions is required.

Benchmark model data for the period 1931-1996 is produced by applying equation 6.4 (i.e. the Miami model), using the same temperature and precipitation input

datasets that drive the LPJmL simulations (Harris et al., 2014). The resulting *LandGrid* object has 66 layers, with each layer containing total annual NPP values in DM m^{-2}. The cell values are then multiplied by the factor 0.45, to convert from amount of dry organic matter (gDM) to amount of carbon (gC). Subsequently, the per-cell-average of the annual NPP values is calculated by applying the *mean* function on the *LandGrid* object, producing a new *LandGrid* object with one layer.

LPJmL model results are available in a .bin file, and monthly NPP for the period 1931-1996 is loaded into a *LandGrid* object, using the *readLPJBin* function from the **PIKTools** package. Before applying the *mean* function, monthly NPP values are summed for each year.

In order ensure comparability of the different datasets, all cells in the Miami and LPJmL model results that correspond to land cells with 0% grassland (see section 6.1.1, model protocol), must obtain a non-value (NA) designation. These cells will subsequently be discarded by the *harmonise* function and are therefore excluded from the calculation of metrics and the Taylor diagram. Turning non-grassland land cells into NA-cells is achieved by applying the grassland class of the landuse input-dataset as a mask (see section 4.2.3) on the two raster datasets. Figure 6.2 shows the spatial pattern of average total annual NPP in grassland cells that the different models/scenarios produce for the period 1931 - 1996.

Each of the three *LandGrid* objects is then harmonised with the *LandPoint* object containing EMDI NPP site data before metrics are calculated and the Taylor diagram is produced.

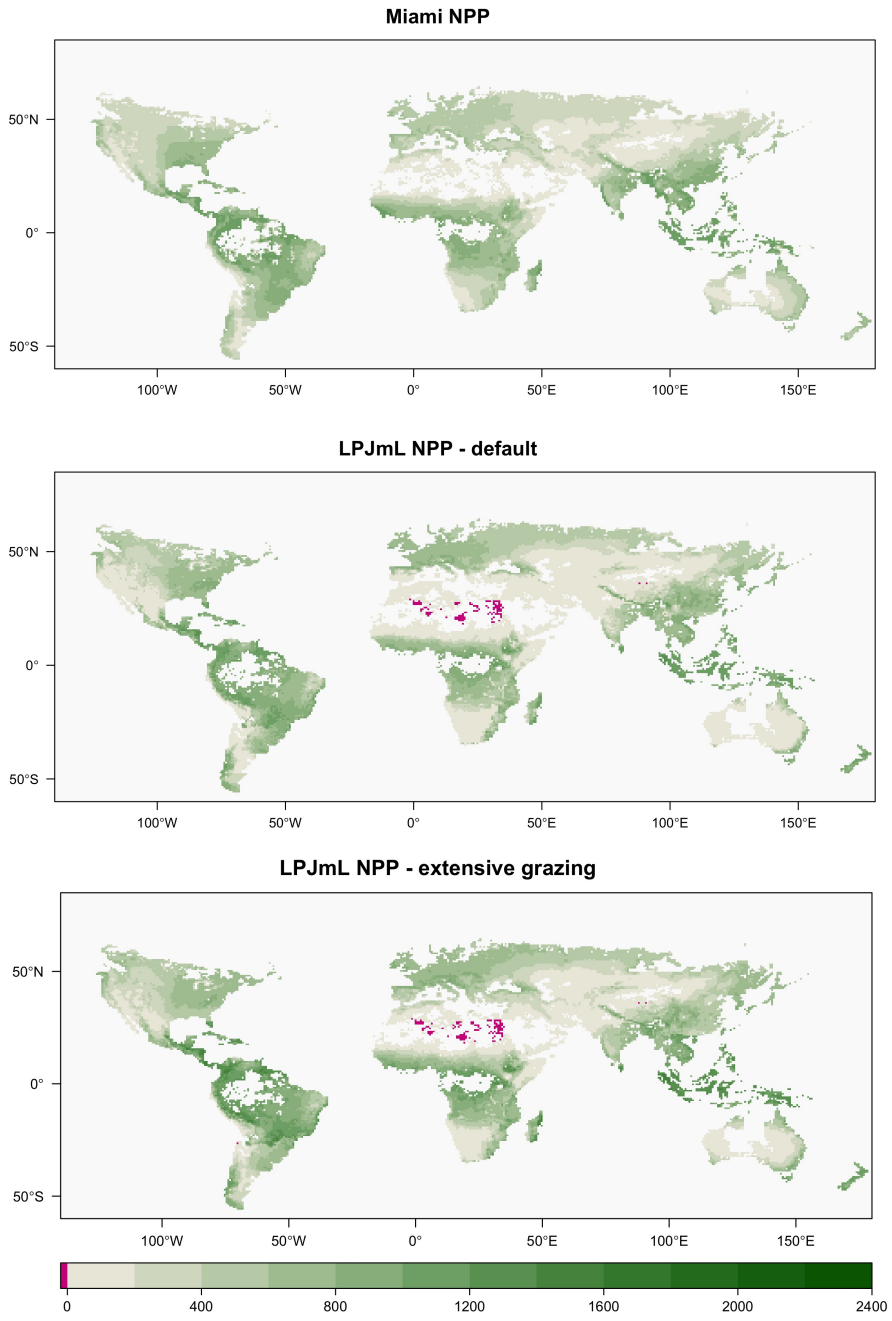


Figure 6.2: Average annual NPP (gC m^{-2} $year^{-1}$) for the period 1931 - 1996 in the benchmark (Miami) and LPJmL model.

Case 2: MODIS reference data

Annual MODIS NPP data for the period 2000 - 2005, available in the GeoTIFF file format at a spatial resolution of 30 arc-seconds, are loaded into *R* using the *brick* function of the **raster** package, which is capable of reading data from the most common raster formats into a *RasterBrick* object. The MODIS NPP high-resolution rasters need to be aggregated to the same cell size as the LPJmL and benchmark model results (0.5° in latitude and longitude). This is achieved with the *aggregate* function in the **raster** package, using an aggregation factor of 60 (30 arc-seconds * 60 = 0.5°). The aggregated MODIS dataset is then coerced to a *LandGrid* object using *landgrid* and masked by the grassland landuse dataset. In a last step, the per-cell-mean of the NPP values from 2000 - 2005 is calculated using the *mean* function.

LPJmL and Miami model results of NPP for the period 2000 - 2005 are prepared in the same way as has been described in the previous section. The distribution of average total annual NPP in grassland cells for the aggregated MODIS satellite data product, the Miami model and the LPJmL scenarios is shown in Figure 6.3.

The *LandGrid* objects containing Miami and LPJml model results are harmonised with the MODIS NPP *LandGrid* object, and subsequently metrics are computed and Taylor diagrams produced. In order to demonstrate the capability of the benchmarking system in conducting analyses at different spatial scales, the same procedure is performed for three different countries with a large area of managed grasslands: Argentina, Kazakhstan and Mongolia.

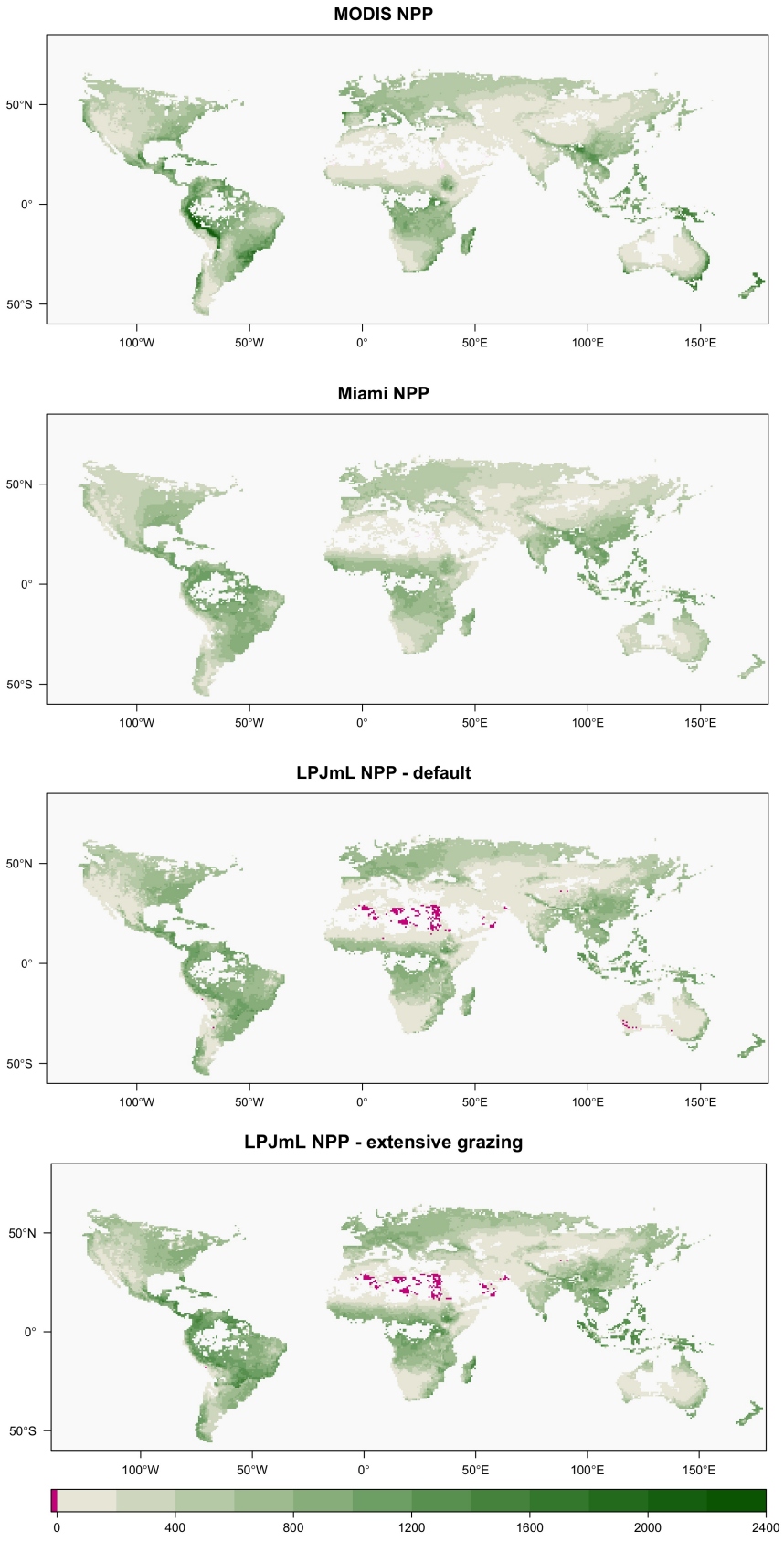


Figure 6.3: Average annual NPP (gC m^{-2} $year^{-1}$) for the period 2000 - 2005 in reference dataset (MODIS) and models.

6.2.5 Results and discussion

Taylor diagrams and tables of metrics for the comparison of benchmark (Miami model) and LPJmL model results with EMDI site data (1931 - 1996) and MODIS NPP (2000 - 2005) have been produced using the *taylorDiagram* and *computeMetrics* functions, respectively. Results are presented separately, according to reference dataset and spatial scale, and the discussion focusses on the performance of LPJmL scenarios relative to the benchmark, rather than on absolute values.

Case 1: EMDI reference data (1931 - 1996)

In comparison with EMDI site data, the LPJmL management scenarios exhibit very similar performances, but fall behind the benchmark in terms of the centred pattern RMSE and standard deviation, but very similar in terms of correlation (Figure 6.4). Therefore, the two LPJmL scenarios do not capture the spatial variability in the reference data as well as the benchmark, at least when the effect due to a difference in the data mean is removed. Correlation coefficients indicate that the benchmark model and both LPJmL scenarios have a weak linear relationship with the reference data.

Looking at the metrics in Table 6.1 one can derive a more comprehensive picture of the model-reference differences than from looking at the Taylor diagram alone, which is especially helpful for a more targeted model assessment. For example, the LPJmL D and G_D scenarios perform better than the benchmark in terms of similarity with reference mean and shape of the distribution (skewness and kurtosis), respectively. From the difference in mean bias (MB) and mean absolute error (MAE) it can be inferred that benchmark model and LPJmL G_D tend to overestimate reference data (i.e. high proportion of positive errors), and LPJmL D exhibits a more balanced distribution of positive and negative error values.

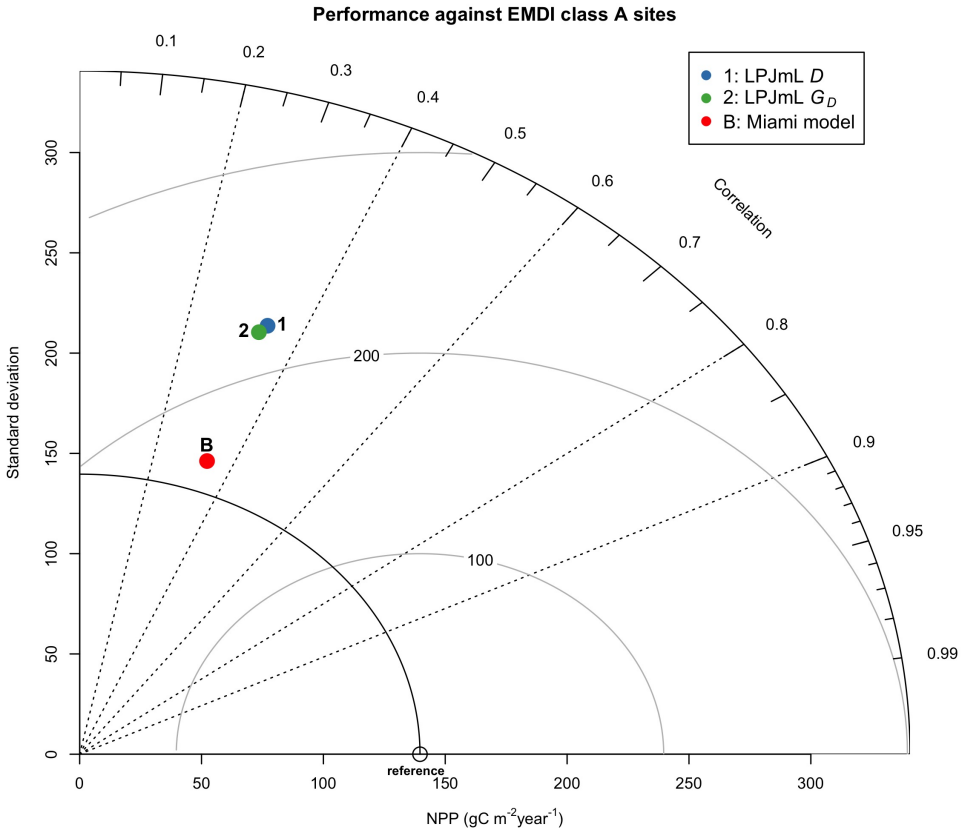


Figure 6.4: Taylor diagram showing statistics for benchmark (Miami model) and LPJmL model results compared to EMDI site data (reference point).

Comparing MAE and root mean square error (RMSE) scores reveals that the variance in individual errors is higher between LPJmL scenarios and reference data than between the benchmark and reference data. Relatively large individual errors seem to be present between LPJmL G_D and the reference data, which is indicated by the magnitude of the difference between MAE and RMSE.

In absolute terms, the two LPJmL management scenarios perform better than the benchmark when looking at MAE and RMSE scores. However, removing the effect of data-mean differences results in a poorer performance compared to the benchmark, at least for the centred RMSE. When scaling is ignored, the performance of benchmark model and LPJmL scenarios significantly improve, with negligible differences in scaled MAE and RMSE scores. This suggests that adjustment of model

parameters can result in an improved simulation of NPP, at least in relation to the EMDI NPP reference data used here.

Table 6.1: Distributional measures and scores for the comparison of benchmark (Miami model) and LPJmL scenarios with NPP data from EMDI class A sites.

Metric	**Benchmark**	**Default scenario *D***	**Extensive grazing scenario G_D**
Mean	451 *(1.86)*	313 ***(1.28)***	383 *(1.58)*
SD	155 ***(1.11)***	227 *(1.63)*	223 *(1.60)*
Skewness	0.28 *(0.12)*	0.56 *(0.25)*	0.70 ***(0.31)***
Kurtosis	-0.68 *(-0.11)*	-0.60 *(-0.10)*	-0.23 ***(-0.04)***
MB	208	**69.7**	140
MAE	224	**172**	181
Centred MAE	448	**173**	263
Scaled MAE	94.4	**94.2**	94.6
RMSE	269	**233**	261
Centred RMSE	**170**	223	221
Scaled RMSE	132	**131**	132
r^2	**0.113**	**0.113**	0.109
d_r	-0.112	**0.122**	0.075

Note: Values in brackets are ratios between for model and reference. Best scores are in bold.

Both similarity measures, i.e. coefficient of determination (r^2) and index of agreement (d_r), indicate a generally poor representation of the reference data by the benchmark and LPJmL model. However, LPJmL d_r scores suggest a better performance than the benchmark.

In summary, the metric scores in Table 6.1 imply that the EMDI NPP reference data is not represented well by either the benchmark model or the LPJmL scenarios. Except for the similarity in standard deviation, error variance and centred (pattern) RMSE scores, the LPJmL scenarios perform better than the benchmark. There is potential for a significantly improved performance of the models with respect to the reference data through parameter adjustment, as indicated by the difference in ordinary and scaled MAE and RMSE scores.

Case 2a: Global comparison with MODIS data (2000-2005)

Comparing the global-scale performance of benchmark model and LPJmL scenarios in a Taylor diagram indicates that they behave similarly with respect to the MODIS NPP reference (Figure 6.5). While the two LPJmL scenarios are hardly distinguishable, there is a perceptible difference between the benchmark and LPJmL model standard deviations: standard deviations in LPJmL D and G_D are closer to the reference. The correlation coefficients are very similar and suggest a strong linear relationship between benchmark and LPJmL model results and MODIS reference data. Differences between LPJmL D, LPJmL G_D and benchmark centred pattern RMSE scores are hardly noticeable.

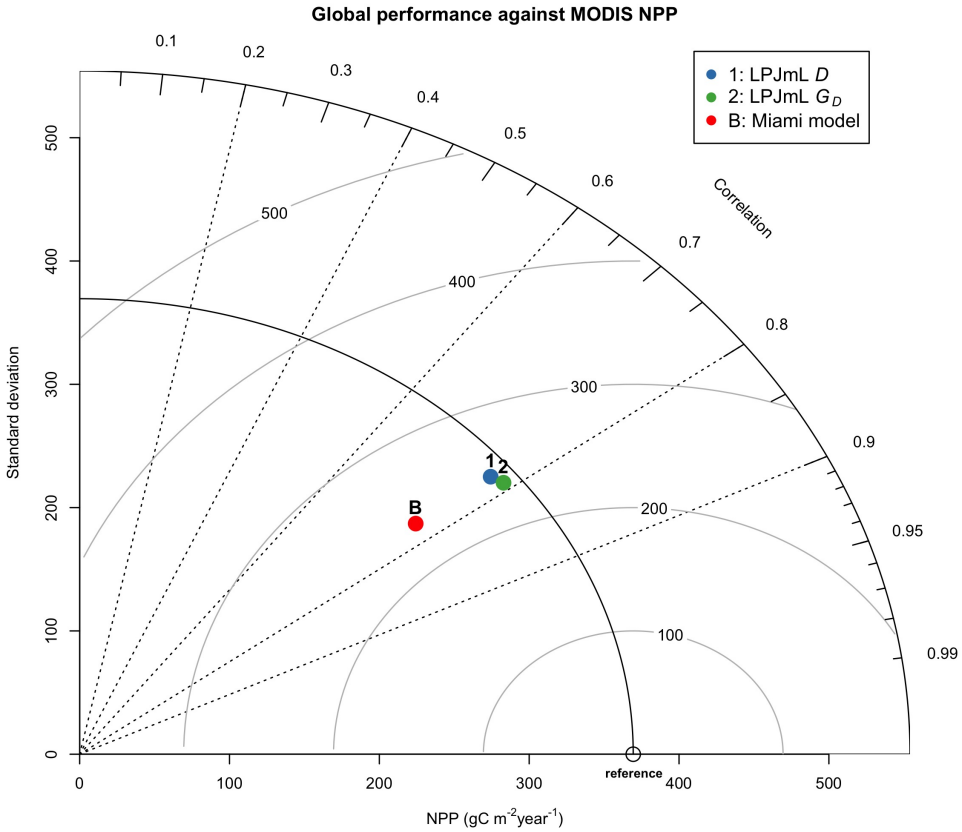


Figure 6.5: Taylor diagram showing statistics for benchmark (Miami model) and LPJmL model results compared to MODIS NPP data (reference point).

The distributional metrics in Table 6.2 show that mean values of benchmark and LPJmL model results are very similar to the reference mean. Value concentration relative to the mean (skewness) in the reference data is best represented by the benchmark, but when looking at kurtosis scores, the LPJmL G_D is closest to the reference. However, in general, the differences between benchmark and LPJmL scenarios are relatively small.

Differences between MB and MAE scores do not indicate a tendency in any of the models to over- or underestimate MODIS reference data (Table 6.2). The variance in individual errors (i.e. MAE vs. RMSE scores) is generally high, with no significant difference between benchmark and LPJmL scenarios. Likewise, MAE and RMSE scores are very similar, and removing the effects of differences in mean (centred

measures) or scaling (scaled measures) does not result in substantial performance improvements.

Unsurprisingly, the similarity measures suggest a good representation of the reference data by both the benchmark and LPJmL model results.

Table 6.2: Distributional measures and scores for the global comparison of benchmark (Miami model) and LPJmL scenarios with MODIS NPP.

Metric	**Benchmark**	**Default scenario *D***	**Extensive grazing scenario G_D**
Mean	483 *(1.03)*	459 ***(0.98)***	527 *(1.12)*
SD	292 *(0.79)*	355 *(0.96)*	358 ***(0.97)***
Skewness	0.27 ***(0.31)***	0.07 *(0.08)*	0.14 *(0.16)*
Kurtosis	-1.02*(-1.28)*	-1.4 *(-1.76)*	-1.00 ***(-1.25)***
MB	13.4	**-10.6**	57.7
MAE	**167**	170	175
Centred MAE	173	**170**	197
Scaled MAE	167	162	**153**
RMSE	**237**	245	244
Centred RMSE	**237**	244	**237**
Scaled RMSE	237	234	**227**
r^2	0.590	0.598	**0.623**
d_r	**0.716**	0.708	0.699

Note: Values in brackets are ratios between for model and reference. Best scores are in bold.

Taylor diagram (Figure 6.5) and metrics (Table 6.2) show a good agreement of benchmark and LPJmL model results with MODIS NPP data on the global scale. In addition, the metrics do not imply a potential for substantial performance improvement by parameter adjustment. Differences between benchmark and LPJmL scores for the different metrics are generally negligible.

Case 2b: Regional comparison with MODIS data (2000-2005)

In order to account for varying orders of magnitude due to different areal extents and mean NPP values in the three considered regions, i.e. Argentina, Kazakhstan and Mongolia, metrics of the Taylor diagram (Figure 6.6) and deviance measures (Table 6.3) have been normalised. Furthermore, only the model-reference ratios of distributional measures are given in Table 6.3, to facilitate the comparison.

The Taylor diagram for the regional comparison of benchmark and LPJmL NPP model results with MODIS NPP reveals the following pattern (Figure 6.6): correlation coefficients of benchmark model and LPJmL scenarios are generally higher and more similar in Mongolia and Kazakhstan than in Argentina, where the benchmark model (B_ARG) outperforms the LPJmL scenarios (ARG1 and ARG2). Benchmark centred pattern RMSE scores are lower, i.e. perform better, than for the LPJmL scenarios in all three regions. In Argentina, LPJmL scenarios (ARG1 and ARG2) perform better than the benchmark (B_ARG) in terms of standard deviations. The opposite is true for Kazakhstan and Mongolia.

Measures of deviance in Table 6.3 do not indicate a trend of over- or underestimation of reference data by the benchmark and LPJmL model (MB vs. MAE scores) and imply that the variance in individual errors (MAE vs. RMSE scores) is generally higher in Argentina, both in the benchmark model and the LPJmL scenarios.

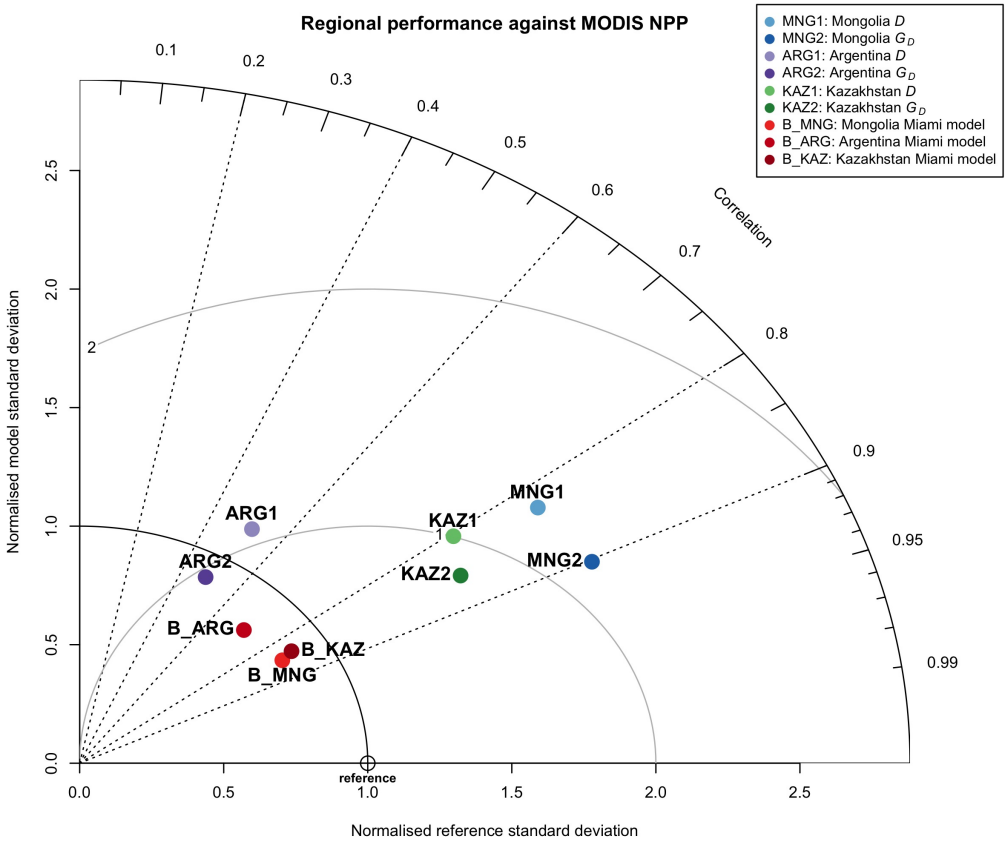


Figure 6.6: Normalised Taylor diagram showing statistics for benchmark (Miami model) and LPJmL model results in different regions compared to MODIS NPP data (reference point).

Ordinary MAE and RMSE scores, as well as centred RMSE scores of the benchmark are lower, i.e. better, than LPJmL scenarios in the three regions, with the exception of the ordinary MAE score for LPJmL G_D in Kazakhstan. The most significant improvements in performance can be achieved for the LPJmL scenarios in Kazakhstan and Mongolia, when the effect of scaling is removed (scaled MAE and RMSE scores). Therefore, adjustment of parameters in LPJmL G_D could potentially result in a better performance than the benchmark in Kazakhstan and Mongolia.

The coefficient of determination (r^2) is generally higher in Kazakhstan and Mongolia, while the index of agreement (d_r) scores do not show a conclusive pattern.

Table 6.3: Distributional measures and scores for the regional comparison of benchmark (Miami model) and LPJmL scenarios with MODIS NPP.

Metric	**Benchmark**	**Default scenario *D***	**Extensive grazing scenario G_D**
Mean ratio	1.12*/1.28†/1.15§	1.08*/0.90†/**0.97**§	1.17*/1.04†/1.21§
SD ratio	0.80*/0.87†/0.83§	1.15*/1.61†/1.92§	**0.90***/1.54†/1.97§
Skewness ratio	0.04*/0.60†/0.53§	0.12*/1.09†/1.43§	0.54*/1.10†/**1.06**§
Kurtosis ratio	-0.20*/0.29†/-1.31§	-0.27*/0.67†/3.83§	0.40*/**1.04**†/1.56§
MB	0.13*/0.28†/0.15§	0.09*/-0.10†/**-0.03**§	0.17*/0.04†/0.21§
MAE	0.35*/0.33†/**0.24**§	0.55*/0.40†/0.47§	0.46*/0.31†/0.45§
Centred MAE	0.44*/0.60†/0.34§	0.57*/0.49†/0.56§	0.56*/**0.32**†/0.46§
Scaled MAE	0.31*/0.19†/0.19§	0.38*/0.21†/0.21§	0.41*/0.18†/**0.16**§
RMSE	0.49*/0.38†/**0.29**§	0.73*/0.50†/0.59§	0.68*/0.42†/0.59§
Centred RMSE	0.48*/0.27†/**0.25**§	0.72*/0.50†/0.59§	0.65*/0.42†/0.55§
Scaled RMSE	0.48*/0.27†/0.25§	0.58*/0.29†/0.27§	0.59*/0.25†/**0.21**§
r^2	0.51*/0.71†/0.72§	0.27*/0.65†/0.69§	0.24*/0.74†/**0.81**§
d_r	0.68*/0.62†/**0.71**§	0.50*/0.51†/0.41§	0.56*/0.61†/0.44§

Note: Best scores are in bold. * = Argentina, † = Kazakhstan, § = Mongolia.

In the regional comparison with MODIS NPP, the benchmark model generally performs better than LPJmL *D* and G_D. Contrary to the findings in the global comparison with MODIS NPP (case 2a), the measures of deviance for the different regions (Table 6.3) indicate a potential for model improvement by parameter adjustment, especially in Kazakhstan and Mongolia.

6.3 Use-case 2: Improvements in grassland GPP

In order to demonstrate the use of Taylor diagrams for tracking model improvements, LPJmL monthly GPP results for the period 2003 - 2005 from a data fitting experiment are used. The experiment comprised the adjustment of eight different parameters that affect, amongst others, the simulation of primary production in LPJmL. These parameters were fitted to produce model results that match FluxNet GPP measurements at four different grassland sites as closely as possible, using an optimisation algorithm and a Markov chain Monte Carlo method (Soetaert and Petzoldt, 2010) to obtain, at each location, an optimal configuration of the eight parameters and parameter uncertainties.

The GPP model results were obtained by running LPJmL with default and site-specific optimal parameter configurations, amounting to eight runs in total (two runs at four sites). During each run, monthly GPP is only computed for the cell whose cell-centre coordinates are closest to the coordinates of the FluxNet site (Table 6.4).

Table 6.4: FluxNet sites used for parameter fitting (Oak Ridge National Laboratory Distributed Active Archive Center, 2015).

FluxNet site	**Coordinates (Lat,Lon)**	**Country**
Walnut Gulch Kendall Grasslands (US-Wkg)	31.73°, -109.94°	U.S.A
Freeman Ranch-Mesquite Juniper (US-FR2)	29.95°, -97.99°	U.S.A
Southern Great Plains control site (US-ARc)	35.55°, -98.04°	U.S.A
Monte Bondone (IT-MBo)	46.01°, 11.05°	Italy

6.3.1 Dataset preparation

Monthly GPP (gC m^{-2} $month^{-1}$) model results, for default and optimal parameter configurations, and site measurements for the period 2003 - 2005 are imported into *R*. No harmonisation is required and the data is partitioned into two list objects, one containing model results with default parameter configurations and corresponding FluxNet measurements, and one containing model results with optimal parameter configurations and corresponding FluxNet measurements. These objects are passed to the *taylorDiagram* function to produce an improvement-tracker Taylor diagram (see section 3.3.3). Code used to format data and produce results listed in Appendix B.

6.3.2 Results and discussion

Length and direction of the arrows in the improvement-tracker Taylor diagram (Figure 6.7) indicate amount and type of model improvement, or degradation, respectively. LPJmL model results of GPP at cell location 1 (corresponding to US-Wkg FluxNet station) exhibit a significant change in standard deviation and, to a lesser extent, in centred pattern RMSE, but only a modest change in correlation. While centred pattern RMSE and correlation coefficient improve with the optimal parameter configuration, the standard deviation in the model results actually becomes more different from the reference standard deviation. The same is true for cell location 2 (US-FR2): centred pattern RMSE improves and correlation with site measurements increases, while the differences in standard deviations becomes larger.
In contrast to that, model results at cell locations 3 and 4 exhibit a larger centred pattern RMSE, i.e. poorer performance, when the optimal parameter configuration is used. Differences between model and reference standard deviations become larger, while correlations remain approximately equal.

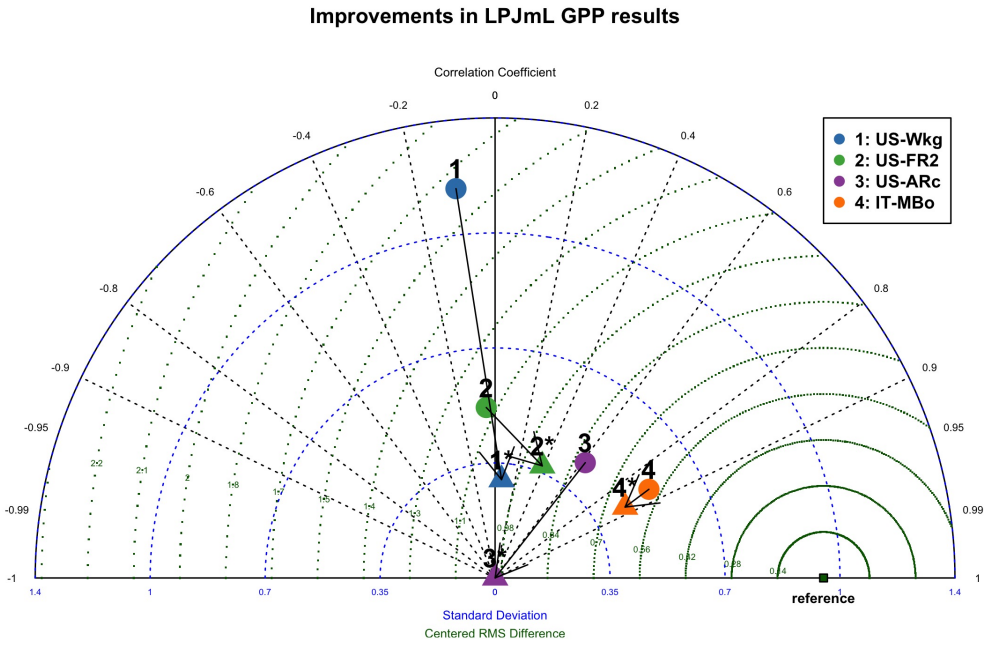


Figure 6.7: Normalised improvement-tracker Taylor with statistics for LPJmL model results with default (circles) and optimal (triangles) parameter configurations compared to FluxNet GPP data (reference point).

In summary, the improvement-tracker Taylor diagram revealed that an 'optimal' parameter configuration obtained through parameter fitting does not necessarily result in model improvements in all dimensions (locations 1 and 2), and can even lead to deterioration of the results with respect to the used reference data (locations 3 and 4).

Chapter 7

Conclusion and outlook

7.1 Conclusion

Vegetation models, such as DGVMs and LSMs, have been developed to increase the understanding of physical, chemical and dynamic processes connected to the global biosphere, and are an important component of climate models. More recent versions of vegetation models, like LPJmL (Bondeau et al., 2007), focus on the representation of agricultural processes, with the aim of understanding and evaluating the role of agricultural systems in the global carbon and water cycle.

Benchmarking is a valuable tool that can be utilised for target-aimed assessment and improvement of models. The prerequisite for effective and compatible community-wide benchmarking initiatives is a well-accepted, standardised benchmarking system, including collectively defined reference datasets, benchmarks and metrics for a scoring system (e.g. Luo et al., 2012; Kelley et al., 2013). Furthermore, since virtually all datasets used in the vegetation modelling community are of the spatial type, the benchmarking system would profit from built-in spatial data integration functionality, which is usually restricted to specialised applications, such as GIS.

In this thesis, the design and implementation of an accessible, user-friendly benchmarking system application has been described and demonstrated, with special focus on a readily understandable spatial data integration function, and tools for comprehensive model evaluation and improvement tracking. It has been shown that point and raster reference datasets from different sources can be easily imported, manipulated and harmonised with model data, using functions from the two customised packages **LandMark** and **PIKTools**, and from related libraries. Additionally, the built-in utility functions allow for straightforward creation and manipulation of spatial data objects, such as benchmark datasets.

In use-case 1, it has been demonstrated that Taylor diagrams facilitate a quick, preliminary evaluation of model performance relative to the benchmark, whereas the selected metrics provide a more comprehensive picture of the model's capacity to reproduce reference data. The possibility to conduct benchmark analyses at different spatial scales using built-in extraction tools is an important feature of the benchmarking system, as is evident from the comparison of the global and regional benchmark analysis with MODIS NPP reference data in use-case 1. Without this functionality, strong variations in performance or potential for improvement of the model in certain regions might be overlooked.

Lastly, use-case 2 illustrated the capability of a modified version of Taylor diagrams for model improvement tracking. These diagrams are not only useful for the evaluation of relative improvements in different model aspects, but help model developers to discriminate between different types of improvement: for example, changes in model parameters might result in smaller model errors, but simultaneously cause a lower correlation with the reference data.

The benchmarking system presented here enables model users and developers to comprehensively evaluate model performances and to identify weaknesses and potential improvements of the model on different scales.

7.2 Outlook

The benchmarking system presented here is a collection of *R*-packages that can be easily distributed and extended to satisfy the requirements of different modelling groups. In order to increase the versatility of the system, add-in packages similar to **PIKTools** can be created for different vegetation models. Furthermore, the current version of the system does not include metrics that facilitate the assessment of phase differences, i.e. time shifts, between model and reference data. An extension of the **LandMark** package by a function that includes measures such as cross-correlation or mean-phase difference (Kelley et al., 2013) would enable users to assess model performance more comprehensively. Another step that would contribute to the user-friendliness of the system, is to provide a selection of different metric ensembles for a range of applications or analysis purposes.

Bibliography

Abramowitz, G. (2005). Towards a benchmark for land surface models. *Geophysical Research Letters*, 32(22):L22702.

Abramowitz, G. (2012). Towards a public, standardized, diagnostic benchmarking system for land surface models. *Geoscientific Model Development*, 5(3):819–827.

Abramowitz, G., Leuning, R., Clark, M., and Pitman, A. (2008). Evaluating the performance of land surface models. *Journal of Climate*, 21(21):5468–5481.

Adams, B., White, A., and Lenton, T. (2004). An analysis of some diverse approaches to modelling terrestrial net primary productivity. *Ecological Modelling*, 177(3-4):353–391.

Bennett, N. D., Croke, B. F., Guariso, G., Guillaume, J. H., Hamilton, S. H., Jakeman, A. J., Marsili-Libelli, S., Newham, L. T., Norton, J. P., Perrin, C., Pierce, S. A., Robson, B., Seppelt, R., Voinov, A. A., Fath, B. D., and Andreassian, V. (2013). Characterising performance of environmental models. *Environmental Modelling & Software*, 40:1–20.

Best, M. J., Abramowitz, G., Johnson, H. R., Pitman, A. J., Balsamo, G., Boone, A., Cuntz, M., Decharme, B., Dirmeyer, P. A., Dong, J., Ek, M., Guo, Z., Haverd, V., van den Hurk, B. J. J., Nearing, G. S., Pak, B., Peters-Lidard, C., Santanello, J. A., Stevens, L., and Vuichard, N. (2015). The plumbing of land surface mod-

els: Benchmarking model performance. *Journal of Hydrometeorology*, 16(3):1425–1442.

Bivand, R., Pebesma, E., and Gomez-Rubio, V. (2013). *Applied spatial data analysis with R.* Springer, New York City, New York, U.S.A.

Blyth, E., Clark, D. B., Ellis, R., Huntingford, C., Los, S., Pryor, M., Best, M., and Sitch, S. (2011). A comprehensive set of benchmark tests for a land surface model of simultaneous fluxes of water and carbon at both the global and seasonal scale. *Geoscientific Model Development*, 4(2):255–269.

Blyth, E. M., Best, M., Cox, P., Essery, R., Boucher, O., Harding, R., Prentice, I. C., Vidale, P.-L., and Woodward, I. (2006). Jules: a new community land surface model. *IGBP newsletter*, 6(9–11).

Bondeau, A., Smith, P. C., Zaehle, S., Schaphoff, S., Lucht, W., Cramer, W., Gerten, D., Lotze-Campen, H., Mueller, C., Reichstein, M., and Smith, B. (2007). Modelling the role of agriculture for the 20th century global terrestrial carbon balance. *Global Change Biology*, 13(3):679–706.

Brovkin, V., Ganopolski, A., and Svirezhev, Y. (1997). A continuous climate-vegetation classification for use in climate-biosphere studies. *Ecological Modelling*, 101:251–261.

Camp, R. C. (1989). *Benchmarking: The Search for the Industry Best Practice that Leads to Superior Performance.* ASQC Quality Press, Milwaukee, Wisconsin, U.S.A.

Chai, T. and Draxler, R. R. (2014). "root mean square error (rmse) or mean absolute error (mae)?" arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250.

Chambers, J. M. (1998). *Programming with Data - A Guide to the S Language*. Springer, New York City, New York, U.S.A.

Collatz, G. J., Ball, J. T., Grivet, C., and Berry, G. A. (1991). Physiological and environmental regulation of stomatal conductance, photosynthesis and transpiration: a model that includes a laminar boundary layer. *Agricultural and Forest Meteorology*, 54:107–136.

Collatz, G. J., Ribas-Carbo, M., and Berry, J. A. (1992). A coupled photosynthesis - stomatal conductance model for leaves of c4 plants. *Australian Journal of Plant Physiology*, 19:519–538.

Cramer, W., Kicklighter, D. W., Bondeau, A., Moore, B., Churkina, G., Nemry, B., Ruimy, A., and Schloss, A. L. (1999). Comparing global models of terrestrial net primary productivity (NPP): overview and key results. *Global Change Biology*, 5:1–15.

Dai, A., Qian, T., Trenberth, K. E., and Milliman, J. D. (2009). Changes in continental freshwater discharge from 1948–2004. *Journal of Climate*, 22(2773 – 2791).

Döll, P. and Siebert, S. (1999). *A Digital Global Map of Irrigated Areas*. University of Kassel, Kassel, Germany.

Farquhar, G. and von Caemmerer, S. (1982). Modelling of photosynthetic response to environmental conditions. In Lange, O., Nobel, P., Osmond, C., and Ziegler, H., editors, *Physiological Plant Ecology II*, volume 12 / B of *Encyclopedia of Plant Physiology*, pages 549–587. Springer, Berlin, Germany.

Farquhar, G. D., von Caemmerer, S., and Berry, J. A. (1980). A biochemical model of photosynthetic CO_2 assimilation in leaves of C3 species. *Planta*, 149:78–90.

Fifer, R. M. (1988). *Benchmarking beating the competition: A practical guide to benchmarking*. Kaiser Associates, Vienna, Virginia, U.S.A.

Foley, J. A., Prentice, I. C., Ramankutty, N., Levis, S., Pollard, D., Sitch, S., and Haxeltine, A. (1996). An integrated biosphere model of land surface processes, terrestrial carbon balance, and vegetation dynamics. *Global Biogeochemical Cycles*, 10(4):603–628.

Friend, A. D., Stevens, A. K., Knox, R. G., and Cannell, M. G. R. (1997). A process-based terrestrial biosphere model of ecosystem dynamics. *Ecological Modelling*, 95:249–287.

Gent, P. R., Danabasoglu, G., Donner, L. J., Holland, M. M., Hunke, E. C., Jayne, S. R., Lawrence, D. M., Neale, R. B., Rasch, P. J., Vertenstein, M., Worley, P. H., Yang, Z.-L., and Zhang, M. (2011). The community climate system model version 4. *Journal of Climate*, 24(19):4973–4991.

Global Administrative Areas (2015). *GADM database of Global Administrative Areas Version 2.7, Web Page* / University of California, Davis, California, U.S.A. http://www.gadm.org. [Accessed: 2015-06-29].

Global Runoff Data Centre (2007). *Major River Basins of the World* / The Global Runoff Data Center (GRDC), Koblenz, Germany. http://www.gadm.org. [Accessed: 2015-06-13].

Harrell Jr, F. E., with contributions from Charles Dupont, and many others. (2015). *Hmisc: Harrell Miscellaneous*. R package version 3.16-0.

Harris, I., Jones, P., Osborn, T., and Lister, D. (2014). Updated high-resolution grids of monthly climatic observations – the CRU TS3.10 Dataset. *International Journal of Climatology*, 34(3):623–642.

Hartig, F., Dyke, J., Hickler, T., Higgins, S. I., O'Hara, R. B., Scheiter, S., and Huth, A. (2012). Connecting dynamic vegetation models to data – an inverse perspective. *Journal of Biogeography*, 39(12):2240–2252.

Hijmans, R. J. (2015). *raster: Geographic Data Analysis and Modeling*. R package version 2.3-40. http://CRAN.R-project.org/package=raster.

Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque, J. F., Large, W. G., Lawrence, D., Lindsay, K., Lipscomb, W. H., Long, M. C., Mahowald, N., Marsh, D. R., Neale, R. B., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., Collins, W. D., Hack, J. J., Kiehl, J., and Marshall, S. (2013). The community earth system model: A framework for collaborative research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360.

Intergovernmental Panel on Climate Change (2014a). Summary for policy makers. In Field, C. B., Barros, V. R., Dokken, D. J., Mach, K. J., Mastrandrea, M. D., Bilir, T. E., Chatterjee, M., Ebi, K. L., Estrada, Y. O., Genova, R. C., Girma, B., Kissel, E. S., Levy, A. N., MacCracken, S., Mastrandrea, P. R., and White, L. L., editors, *Climate Change 2014: Impacts, Adaptation, and Vulnerability. Part A: Global and Sectoral Aspects. Contribution of Working Group II to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, pages 1–32. Cambridge University Press, Cambridge, United Kingdom.

Intergovernmental Panel on Climate Change (2014b). Synthesis report. In Core Writing Team, R. P. and (eds.), L. M., editors, *Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva, Switzerland. 151 pp.

International Organization for Standardization [ISO] (2014). 19115-1:2014: Geographic information - metadata - part 1: Fundamentals. Technical report, International Organization for Standardization, Geneva, Switzerland.

Jachner, S. and v. d. Boogaart, K. G. (2007). Statistical methods for the qualitative

assessment of dynamic models with time delay (r package qualv). *Journal of Statistical Software*, 22(8):1–30.

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., and Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. *Biogeosciences*, 10(5):3313–3340.

Krinner, G., Viovy, N., de Noblet-Ducoudré, N., Ogée, J., Polcher, J., Friedlingstein, P., Ciais, P., Sitch, S., and Prentice, I. C. (2005). A dynamic global vegetation model for studies of the coupled atmosphere-biosphere system. *Global Biogeochemical Cycles*, 19(1):GB1015.

Lemon, J. (2006). Plotrix: a package in the red light district of r. *R News*, 6(4):8–12.

Lieth, H. (1972). Modeling the primary productivity of the world. *Nature and Resources*, 8(2):5–10.

Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W. (2010). Representing geography. In *Geographic Information Systems and Science*, chapter 3, pages 75–97. John Wiley & Sons, Ltd., Hoboken, New Jersey, U.S.A., 3. edition.

Luo, Y. Q., Randerson, J. T., Abramowitz, G., Bacour, C., Blyth, E., Carvalhais, N., Ciais, P., Dalmonech, D., Fisher, J. B., Fisher, R., Friedlingstein, P., Hibbard, K., Hoffman, F., Huntzinger, D., Jones, C. D., Koven, C., Lawrence, D., Li, D. J., Mahecha, M., Niu, S. L., Norby, R., Piao, S. L., Qi, X., Peylin, P., Prentice, I. C., Riley, W., Reichstein, M., Schwalm, C., Wang, Y. P., Xia, J. Y., Zaehle, S., and Zhou, X. H. (2012). A framework for benchmarking land models. *Biogeosciences*, 9(10):3857–3874.

Maguire, D. J. and Dangermond, J. (1991). The functionality of GIS. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographic Information Sys-*

tems: Principles and Applications, chapter 21. John Wiley & Sons, Ltd., Hoboken, U.S.A.

Manabe, S. (1969). Climate and the ocean circulation: 1, the atmospheric circulation and the hydrology of the earth's surface. *Monthly Weather Review*, 97:739–805.

Mohammadi, H., Rajabifard, A., and Williamson, I. P. (2010). Development of an interoperable tool to facilitate spatial data integration in the context of sdi. *International Journal of Geographical Information Science*, 24(4):487–505.

Monteith, J. (1981). Evaporation and environment. In Fogg, C., editor, *The State and Movement of Water in Living Organisms*, pages 205–234.

Monteith, J. L. (1972). Solar radiation and productivity in tropical ecosystems. *Jornal of Applied Ecology*, 9:747–766.

Monteith, J. L. (1995). Accommodation between transpiring vegetation and the convective boundary layer. *Journal of Hydrology*, 166(3–4):251–263.

Nash, J. E. and Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part i — a discussion of principles. *Journal of Hydrology*, 10(3):282–290.

National Aeronautics and Space Administration (2015). *Moderate Resolution Imaging Spectroradiometer (MODIS) Web Page* / National Aeronautics and Space Administration (NASA), Washington D.C., U.S.A. http://modis.gsfc.nasa.gov/data. [Accessed: 2015-10-15].

Neuwirth, E. (2014). *RColorBrewer: ColorBrewer Palettes*. R package version 1.1-2.

Oak Ridge National Laboratory (2010). *The International Land Model Benchmarking Project Web Page* / Oak Ridge National Laboratory (ORNL), Oak Ridge, Tennesse, U.S.A. http://www.ilamb.org/about/contacts.html. [Accessed: 2015-10-15].

Oak Ridge National Laboratory Distributed Active Archive Center (2015). *FLUXNET Web Page* / Oak Ridge National Laboratory Distributed Active Archive Center (ORNL-DAAC), Oak Ridge, Tennesse, U.S.A. http://fluxnet.ornl.gov. [Accessed: 2015-10-15].

Olson, R. J. (2001). *NPP Multi-Biome: NPP and Driver Data for Ecosystem Model-Data Intercomparison.* Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennesse, U.S.A.

Open Geospatial Consortium [OGC] (2011). *OGC Network Common Data Form (NetCDF) Core Encoding Standard version 1.0. OGC 10-090r3.* http://www.opengis.net/doc/IS/netcdf/1.0.

Pasek, J., with some assistance from Alex Tahk, some code modified from R-core; Additional contributions by Gene Culter, and Schwemmle., M. (2014). *weights: Weighting and Weighted Statistics.* R package version 0.80.

Pebesma, E. and Bivand, R. (2005). Classes and methods for spatial data in r. *R News*, 5(2):9–13.

Pierce, D. (2014a). *ncdf: Interface to Unidata netCDF data files.* R package version 1.6.8.

Pierce, D. (2014b). *ncdf4: Interface to Unidata netCDF (version 4 or earlier) format data files.* R package version 1.13.

Pitman, A. J. (2003). The evolution of, and revolution in, land surface schemes designed for climate models. *International Journal of Climatology*, 23(5):479–510.

Prentice, I. C., Liang, X., Medlyn, B. E., and Wang, Y.-P. (2015). Reliable, robust and realistic: the three r's of next-generation land-surface modelling. *Atmospheric Chemistry and Physics*, 15(10):5987–6005.

R Core Team (2015a). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

R Core Team (2015b). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria. https://cran.uni-muenster.de/doc/manuals/r-release/R-exts.pdf.

Ramankutty, N. and Foley, J. A. (1999). Estimating historical changes in global land cover; croplands from 1700 to 1992. *Global Biogeochmeical Cycles*, 13:997–1028.

Randerson, J. T., Hoffman, F. M., Thornton, P. E., Mahowald, N. M., Lindsay, K., Lee, Y.-H., Nevison, C. D., Doney, S. C., Bonan, G., Stoeckli, R., Covey, C., Running, S. W., and Fung, I. Y. (2009). Systematic assessment of terrestrial biogeochemistry in coupled climate-carbon models. *Global Change Biology*, 15(10):2462–2484.

Rost, S., Gerten, D., Bondeau, A., Lucht, W., Rohwer, J., and Schaphoff, S. (2008). Agricultural green and blue water consumption and its influence on the global water system. *Water resources Research*, 44:W09405.

Royce, W. W. (1987). The development of large software systems. In *ICSE '87 Proceedings of the 9th international conference on Software Engineering*, pages 328–338.

Rumbaugh, J., Jacobson, I., and Booch, G. (1999a). *The Unified Modeling Language Reference Manual*. Addison-Wesley Longman Publishing Co., Inc. Boston, Massachusetts, U.S.A.

Rumbaugh, J., Jacobson, I., and Booch, G. (1999b). *The Unified Software Development Process*. Addison-Wesley Longman Publishing Co., Inc. Boston, Massachusetts, U.S.A.

Running, S., R., Nemani, R., Heinsch, F. A., Zhao, M., Reeves, M., and Hashimoto, H. (2004). A continuous satellite-derived measure of global terrestrial primary production. *Bioscience*, 54(6):547–560.

Schaphoff, S., Heyder, U., Ostberg, S., Gerten, D., Heinke, J., and Lucht, W. (2013). Contribution of permafrost soils to the global carbon budget. *Environmental Research Letters*, 8(1):014026.

Simon, H., Baker, K. R., and Phillips, S. (2012). Compilation and interpretation of photochemical model performance statistics published between 2006 and 2012. *Atmospheric Environment*, 61:124–139.

Sitch, S., Smith, B., Prentice, I. C., Arneth, A., Bondeau, A., Cramer, W., Kaplan, J. O., Levis, S., Lucht, W., Sykes, M. T., Thonicke, K., and Venevsky, S. (2003). Evaluation of ecosystem dynamics, plant geography and terrestrial carbon cycling in the LPJ dynamic global vegetation model. *Global Change Biology*, 9(2):161–185.

Smith, P. (2014). Do grasslands act as a perpetual sink for carbon? *Global Change Biology*, 20(9):2708–2711.

Soetaert, K. and Petzoldt, T. (2010). Inverse modelling, sensitivity and monte carlo analysis in R using package FME. *Journal of Statistical Software*, 33(3):1–28.

Taylor, K. E. (2001). Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research*, 106(D7):7183 – 7192.

Thonicke, K., Spessa, A., Prentice, I. C., Harrison, S. P., Dong, L., and Carmona-Moreno, C. (2010). The influence of vegetation, fire spread and fire behaviour on biomass burning and trace gas emissions: results from a process-based model. *Biogeoscience*, 7(6):1991–2011.

Venetucci, R. (1992). Benchmarking: A reality check for strategy and performance objectives. *Production and Inventory Management Journal*, 33(4):32–36.

Waha, K., van Bussel, L. G. J., Müller, C., and Bondeau, A. (2012). Climate-driven simulation of global crop sowing dates. *Global Ecology and Biogeography*, 21(2):247–259.

White, F. (1983). The Vegetation of Africa; a descriptive memoir to accompany the Unesco/AETFAT/UNSO vegetation map of Africa. In *Natural Resources Research Series*, volume XX., page 356. UNESCO, Paris, France.

Willmott, C. J. (1981). On the validation of models. *Physical Geography*, 2:184–194.

Willmott, C. J., Ackleson, S. G., Davis, R. E., Feddema, J. J., Klink, K. M., Legates, D. R., O'Donnell, J., and Rowe, C. M. (1985). Statistics for the evaluation of model performance. *Journal*, 90(C5):8995–9005.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82.

Willmott, C. J., Robeson, S. M., and Matsuura, K. (2012). A refined index of model performance. *International Journal of Climatology*, 32(13):2088–2094.

Willmott, C. J., Robeson, S. M., Matsuura, K., and Ficklin, D. L. (2015). Assessment of three dimensionless measures of model performance. *Environmental Modelling & Software*, 73:167–174.

Zhao, M., Heinsch, F. A., Nemani, R. R., and Running, S. W. (2005). Improvements of the MODIS terrestrial gross and net primary production global data set. *Remote Sensing of the Environment*, 95:164–176.

Appendix A

Code documentation

Code of classes and selected functions is listed in the following order:

1. *.MetaData* data type (**LandMark**)
2. *LandGrid* class (**LandMark**)
3. *LandPoint* class (**LandMark**)
4. *harmonise* function (**LandMark**)
5. *compareSpatial* function (**LandMark**)
6. *getGeodata* function (**PIKTools**)
7. *computeMetrics* function (**LandMark**)
8. *metrics* function list (**LandMark**)
9. *taylorDiagram* function (**LandMark**)

MetaData data type

```
#.MetaData data type definition
#written by Jan Kowalewski
setClass(Class=".MetaData",
         slots=c(
           standard_name = "character",
           z_dim = "character",
           reference_time = "character",
           time_unit = "character",
           fill_value = "numeric",
           scale_factor = "numeric",
           institution = "character",
           source_name = "character",
           references = "character",
           comment = "list"),
         #define default values
         prototype=c(
           standard_name = "",
           z_dim = "",
           reference_time = "",
           time_unit = "",
           fill_value = Inf,
           scale_factor = 1,
           institution = "",
           source_name = "",
           references = "",
           comment = list())
)
```

code/metadata.R

LandGrid class

```
#RasterBrick class from raster package 2.3-40
#written by Robert J. Hijmans
setClass ('RasterBrick',
  contains = 'Raster',
  representation (
    file = '.RasterFile',
    data = '.MultipleRasterData',
    legend = '.RasterLegend'
  )
)

#LandGrid class definition in LandMark package
#contains-argument specifies inheritance
#written by Jan Kowalewski
setClass(Class="LandGrid",
         slots=c(metadata=".MetaData"),
         contains="RasterBrick")
```

code/landgrid_class_small.R

LandPoint class

```
#SpatialPointsDataFrame class definition in sp package version 1.1-0
#written by Edzer Pebesma
setClass("SpatialPointsDataFrame",
  contains = "SpatialPoints",
  representation(data = "data.frame", coords.nrs = "numeric"),
  prototype = list(bbox = matrix(NA), proj4string = CRS(as.character(NA)),
    coords = matrix(NA), data = data.frame(),
    coords.nrs = numeric(0)),
  validity = function(object) {
    if (nrow(object@coords) < 1)
      stop("no points set: too few rows")
    if (ncol(object@coords) <= 1)
      stop("no points set: too few columns")
    if (ncol(object@data) == 0)
      stop("data.frame is empty (possibly after stripping coordinate columns): use
          SpatialPoints() to create points-only object")
    if (nrow(object@data) != nrow(object@coords))
      stop("number of rows in data.frame and SpatialPoints don't match")
    n = length(object@coords.nrs)
    if (n > 0 && n != ncol(object@coords))
      stop("inconsistent coords.nrs slot")
    return(TRUE)
  }
)

#LandPoint class definition
#contains-argument specifies inheritance
#written by Jan Kowalewski
setClass(Class="LandPoint",
         slots=c(metadata=".MetaData", title="character", unit = "character",
         z = "list"),
         contains="SpatialPointsDataFrame")
```

code/landpoint_class_small.R

Harmonise LandGrid and LandPoint objects

Description

Harmonise reference and model data (LandGrid or LandPoint objects) in terms of output format and spatial region/location.

Usage

```
harmonise(x, y, ...)

## S4 method for signature 'LandGrid,LandGrid'
harmonise(x, y, dataset = NULL, FUN = NULL,
  ID = c(2, 3), ...)

## S4 method for signature 'LandPoint,LandGrid'
harmonise(x, y, dataset = NULL, FUN = NULL,
  ID = c(2, 3), ...)
```

Arguments

`x`	LandGrid or LandPoint object. Reference dataset to compare with model data
`y`	LandGrid object. Model results
`...`	Arguments to FUN argument
`dataset`	SpatialPolygonsDataFrame. Dataset used to extract cell values, cell areas and ratio of overlap with the polygon. Check the sp-package for more information on SpatialPolygonsDataFrame.
`FUN`	Some function that returns a SpatialPolygonsDataFrame object that can be used to extract cell values for the desired spatial unit.
`ID`	character or numeric. Names of the column(s) in the SpatialPolygonsDataFrame dataset that will be used to create an ID code. If 2 or more columns are chosen, the values will be separated by points (see ID argument in `extract`).

Value

A list containing 3 data frames (reference data, model, attributes) and metadata. Area values are in km^2.

See Also

`extract`

Examples

```
## Not run:
a <- harmonise(x=reference_data,y=model, dataset = "poly_deu", ID = c(2,3))
```

```
#all functions written by Jan Kowalewski
setGeneric("harmonise", function(x, y, ...)
  standardGeneric("harmonise"))

#harmonise function for two LandGrid objects: e.g.satellite data product and model
    results
setMethod('harmonise', signature(x='LandGrid', y='LandGrid'),
          function(x, y, dataset = NULL, FUN = NULL, ID = c(2,3), ...) {

            #compare raster properties (CRS, res, origin, etc.)
            #using compareSpatial function
            if(compareSpatial(x,y)){
              #crop data sets to same extent
              intersection <- raster::intersect(extent(x), extent(y))
              x <- crop(x, intersection)
              y <- crop(y, intersection)

              #get raster with area values from reference data set
              area_r <- raster::area(x[[1]],na.rm=F)

              #if no polygon is supplied, overlapping extent is used for extraction
              if (is.null(dataset) && is.null(FUN)){
                su <- raster::extent(x)
                reference <- raster::extract(x, su, cellnumbers=T, df=T, nl =
                    nlayers(x))
                #remove rows that contain only NAs
                find_na <- is.na(subset(reference, select = -c(cell,ID)))
                na_rows <- rowSums(find_na)
                keep <- which(na_rows != ncol(find_na))
                reference <- reference[keep,]
                reference <- data.frame(reference, weight=rep(1,nrow(reference)))
                rownames(reference) <- 1:nrow(reference)
              }
              #if a polygon is directly supplied, check if it's a valid object
              else if (!is.null(dataset)){
                stopifnot(class(dataset) == "SpatialPolygons" || class(dataset) ==
                    "SpatialPolygonsDataFrame")
                reference <- extract(x, dataset, ID=ID, cellnumbers=T, df=T, na.rm=
                    TRUE, weights=T)
                rownames(reference) <- 1:nrow(reference)
```

```
37              }
38              #if a polygon is supplied via a function, check if it's a valid
                    object
39              else if (is.null(dataset) && !is.null(FUN)){
40                su <- FUN(...)
41                stopifnot(class(su) == "SpatialPolygons" ||class(su) == "
                      SpatialPolygonsDataFrame")
42                reference <- extract(x, su, ID=ID, cellnumbers=T, df=T, na.rm=TRUE,
                       weights=T)
43                rownames(reference) <- 1:nrow(reference)
44              }
45
46              #extract values from other objects with cellnumbers from reference
                    dataset
47              model <- raster::extract(y, reference$cell)
48              area <- raster::extract(area_r, reference$cell)
49
50              #create separate data frames, first column is always cell number
51              attributes <- data.frame(cell=reference$cell, ID=reference$ID, area=
                    area, weight=reference$weight)
52              reference <- subset(reference, select = -c(ID,weight) )
53              model <- data.frame(cell=reference$cell,model)
54              #combine data frames in a list
55              res <- list(reference=reference, model=model, attributes=attributes)
56            }
57            return(res)
58          }
59 )
60
61 #harmonise function for a LandPoint and LandGrid object: e.g.site measurements and
       model results
62 setMethod('harmonise', signature(x='LandPoint', y='LandGrid'),
63           function(x, y, dataset = NULL, FUN = NULL, ID = c(2,3), ...) {
64             #get CRS of the two objects
65             px <- raster::projection(x, asText=FALSE)
66             py <- raster::projection(y, asText=FALSE)
67             #compare CRS
68             comp <- raster::compareCRS(py, px, unknown=TRUE)
69             #transform LandPoint CRS to LandGrid CRS automatically, if rgdal is
                   present
70             if (!comp) {
```

```
71              if (!raster:::.requireRgdal()) {#check if rgdal is installed ->
                    required for tranformstion
72                warning('CRS of LandPoint and LandGrid do not match')
73              } else {
74                warning('Transforming LandPoint to the CRS of LandGrid model data
                      object')
75                x <- sp::spTransform(x, py)
76              }
77            }
78            #transforming LandPoint to LandGrid, since subsequent computations are
                  faster this way
79            x <- raster::rasterize(x, y, fun=function(x,na.rm)mean(x,na.rm=T))
80            x <- raster::dropLayer(x, 1)
81            x <- landgrid(x)
82
83            #crop data sets to same extent
84            intersection <- raster::intersect(extent(x), extent(y))
85            x <- crop(x, intersection)
86            y <- crop(y, intersection)
87
88            #get raster with area values from reference data set
89            area_r <- raster::area(x[[1]],na.rm=F)
90
91            #if no polygon is supplied, overlapping extent is used for extraction
92            if (is.null(dataset) && is.null(FUN)){
93              su <- raster::extent(x)
94              reference <- raster::extract(x, su, cellnumbers=T, df=T, nl = nlayers
                    (x))
95              #remove rows that contain only NAs
96              find_na <- is.na(subset(reference, select = -c(cell,ID)))
97              na_rows <- rowSums(find_na)
98              keep <- which(na_rows != ncol(find_na))
99              reference <- reference[keep,]
100             reference <- data.frame(reference, weight=rep(1,nrow(reference)))
101             rownames(reference) <- 1:nrow(reference)
102           }
103           #if a polygon is directly supplied, check if it's a valid object
104           else if (!is.null(dataset)){
105             stopifnot(class(dataset) == "SpatialPolygonsDataFrame")
106             reference <- extract(x, dataset, ID=ID, cellnumbers=T, df=T, na.rm=
                    TRUE, weights=T)
```

```
107             rownames(reference) <- 1:nrow(reference)
108           }
109           #if a polygon is supplied via a function, check if it's a valid object
110           else if (is.null(dataset) && !is.null(FUN)){
111             su <- FUN(...)
112             stopifnot(class(su) == "SpatialPolygonsDataFrame")
113             reference <- extract(x, su, ID=ID, cellnumbers=T, df=T, na.rm=TRUE,
                    weights=T)
114             rownames(reference) <- 1:nrow(reference)
115           }
116
117           #extract values from other LandGrid objects with cellnumbers from
                reference
118           model <- raster::extract(y, reference$cell)
119           area <- raster::extract(area_r, reference$cell)
120
121           #create separate data frames, first column is always cell number
122           attributes <- data.frame(cell=reference$cell, ID=reference$ID, area=
                area, weight=reference$weight)
123           reference <- subset(reference, select = -c(ID,weight) )
124           model <- data.frame(cell=reference$cell,model)
125           #combine data frames in a list
126           res <- list(reference=reference, model=model, attributes=attributes)
127
128           return(res)
129         }
130 )
```

code/harmonise.R

```
#written by Jan Kowalewski
setGeneric("compareSpatial", function(x, y)
 standardGeneric("compareSpatial"))

 #compareSpatial checks if several spatial attributes of two LandGrid objects match
 #and suggests how to proceed if they do not match
setMethod('compareSpatial', signature(x='LandGrid', y='LandGrid'),
          function(x, y) {
            crs.c <- raster::compareCRS(x,y)
            resx.c <- raster::xres(x) == raster::xres(y)
            resy.c <- raster::yres(x) == raster::yres(y)
            origx.c <- raster::origin(x)[1] == raster::origin(y)[1]
            origy.c <- raster::origin(x)[2] == raster::origin(y)[2]
            if(!crs.c && (!resx.c | !resy.c)){
              stop("Reference system (CRS) and resolution of LandGrid objects is
                  not equal. Use projectRaster()
                   to transform reference dataset to World Geodetic Datum 1984 (
                       WGS84) and change resolution.")
            }
            if(!crs.c){
              stop("Reference system (CRS) of LandGrid objects is not equal. Use
                  projectRaster()
                   to transform reference dataset to World Geodetic Datum 1984 (
                       WGS84).")
            }
            if((!resx.c | !resy.c) | (!origx.c | !origy.c)){
              stop("Resolution and/or origin of LandGrid objects is not equal. Use
                  resample() or aggregate() first.")
            } else {
                return(TRUE)
            }
          }
)
```

code/compareSpatial.R

Get polygons of different spatial units

Description

Extract one or several polygons from the "geodata" data collection for usage in e.g. the harmonise or extract function in LandMark.

Usage

```
getGeodata(database, spatial_unit = "ALL")
```

Arguments

`database`	character. Name of the data source for the `spatial_unit` argument. For available databases see `print`
`spatial_unit`	character. CODE (see `geodata`) of the region(s) of interest within a certain database

Examples

```
## Not run:
#extract all polygons of the German Bundesländer
#(federal states) and use in harmonise
poly_deu <- getGeodata(database = "gadm_adm1", spatial_unit = "DEU")
ref <- landgrid()
model <- landgrid()

#add values
data <- runif(ncell(ref), 0.1, 0.9)
ref <- setVals(ref, data, layer=1)
model <- setVals(model, data + runif(ncell(model), 0.1, 0.4), layer=1)

df_deu <- harmonise(ref, model, dataset = poly_deu, ID = c("CODE", "PROVINCE"))

#this does the same
 df_deu <- harmonise(ref, model, ID = c("CODE", "PROVINCE"), FUN = getGeodata,
 database = "gadm_adm1", spatial_unit = "DEU")

## End(Not run)
```

```r
#Function to query and return polygons from the "geodata" database
#written by Jan Kowalewski
getGeodata <- function(database, spatial_unit = "ALL"){

  if (missing(database) && spatial_unit == 'ALL'){
    su <- geodata$global
  }
  if(database == 'gadm_adm0'){
    if(spatial_unit[1] == 'ALL'){
      su <- geodata$gadm_adm0
    } else {
      cc <- which(geodata$gadm_adm0$CODE %in% spatial_unit)
      su <- geodata$gadm_adm0[cc,]
    }
  }
  if(database == 'gadm_adm1'){
    if(spatial_unit[1] == 'ALL'){
      su <- geodata$gadm_adm1
    } else {
      cc <- which(geodata$gadm_adm1$CODE %in% spatial_unit)
      su <- geodata$gadm_adm1[cc,]
    }
  }
  if (database == 'grdc_basins'){
    if (spatial_unit[1] == 'ALL'){
      su <- geodata$grdc_basins
    } else {
      cc <- which(geodata$grdc_basins$CODE %in% spatial_unit)
      su <- geodata$grdc_basins[cc,]
    }
  }

  return(su)
}
```

code/getGeodata.R

Compute metrics.

Description

Compute a list of metrics for reference-model comparison. The list includes measures of deviation(absolute and normalized) and of similarity. Only the normalized deviation and similarity measures should be used for comparison of variables with different units. See "Details" section for formulae

Usage

```
computeMetrics(x, y, ...)

## S4 method for signature 'vector,vector'
computeMetrics(x, y, weights = NULL,
  weight.all = FALSE, type = c("all", "absolute", "normalised"), ...)

## S4 method for signature 'data.frame,data.frame'
computeMetrics(x, y, weights = NULL,
  weight.all = FALSE, type = c("all", "absolute", "normalised"), ...)

## S4 method for signature 'list,missing'
computeMetrics(x, use.weights = FALSE,
  weight.all = FALSE, type = c("all", "absolute", "normalised"),
  labels = NULL, ...)
```

Arguments

`x`	vector, data.frame, list. Reference data set or a list containing data frames of the reference and model data sets (and optionally weights). It can also be a list of lists containing model and reference data sets. See details
`y`	vector, data.frame, list, missing. Model data set or a list containing data frames of the reference and model data sets (and optionally weights).
`...`	Additional arguments (none implemented)
`weights`	vector, data.frame. Weights used to calculate weighted metrics.
`weight.all`	logical. If TRUE, measures of deviance and coefficient of determination are also weighted. If FALSE, only distributional measures are weighted.
`type`	character. "absolute" returns only absolute measures of deviance, "normalised" retruns only normalised measures of deviance, "all" returns both types.
`use.weights`	logical. Should the weights, contained in the list(s), be applied, when computing the metrics?
`labels`	character. Labels used for lists, if more than one list is returned.

Details

If the data sets are supplied as vectors or data frames, `x` is always the reference and y the model data set. Weights have to be supplied as a vector or data frame, having the same length (vector) or number of rows (data.frame) as the reference/model data sets.

If the data sets are supplied to as a list (e.g. the return value of `harmonise`) to `x`, and `y` is missing, the list has to contain a data frame for the reference, model and (optionally) weights data. Additionally, it is possible to supply a list of lists to `x`, each of which contains 2 or 3 data frames (reference, model and possibly weights). This approach should be used, when the relative performance of different models or variables is assessed.

Following distributional measures (`?metrics`) are implemented:

-Mean, standard deviation, Skewness (`skewness`), Kurtosis (`kurtosis`)

Following error metrics (`?metrics`) are implemented:

-Mean bias (`mb`)

-Mean absolute error (`mae`), centred mean absolute error (`cmae`), scaled mean absolute error (`smae`)

-Root mean squared error (`rmse`), root centred mean squared error (`rcmse`), root scaled mean squared error (`rsmse`)

-Normalised error metrics (all of the above) - normalised by the mean of the reference data.

Following similarity measures (`?metrics`) are implemented:

-Ceofficient of determination (`r2`)

-Willmott's refined index of agreement 'dr' (`willmott`)

Value

List of metrics (or a list of lists of metrics).

See Also

`harmonise`, `extract`, `metrics`

Examples

```
## Not run:
#create empty LandGrid with 3 layers
ref <- landgrid()
model <- landgrid()

#add values
ref <- setVals(ref, runif(ncell(ref), 0.1, 0.9),
layer=1)
model <- setVals(model, runif(ncell(model), 0.1, 0.9),
layer=1)
```

```
#harmoise values in data sets
data_DEU <- harmonise(x = ref, y = model,
dataset="gadm_adm1", spatial_unit="DEU")

DEU_metrics <- computeMetrics(data_DEU)


## End(Not run)
```

```
#all functions written by Jan Kowalewski

setGeneric("computeMetrics", function(x, y, ...)
  standardGeneric("computeMetrics"))

#compute metrics base function: two supplied vectors, i.e. one for reference data,
    one for model results
setMethod('computeMetrics', signature(x='vector',y='vector'),
          function(x, y, weights=NULL, weight.all = FALSE, type = c("all","absolute
              ", "normalised"), ...) {

            #stopifnot(type %in% c("absolute", "normalised", "all"))
            type <- match.arg(type)

            if (length(x) == 0 || length(y) == 0)
              stop("vector of length zero")
            if (length(x) != length(y) && length(y) != 1)
              stop("incompatible dimensions")

            #remove value pairs with NA
            find_na <- is.na(cbind(x,y))
            na_rows <- rowSums(find_na)
            keep <- which(na_rows == 0)
            x <- x[keep]
            y <- y[keep]
            weights <- weights[keep]


            if(!is.null(weights)) {
              weighted <- TRUE
            } else {
              weighted <- FALSE
            }

            #calculate distributional measures
            mean_ref <- wtd.mean2(x, weights)
            mean_mod <- wtd.mean2(y, weights)

            sd_ref <- sqrt(wtd.var2(x, weights))
            sd_mod <- sqrt(wtd.var2(y, weights))
```

```
39
40            skewness_ref <- skewness(x, weights)
41            skewness_mod <- skewness(y, weights)
42
43            kurtosis_ref <- kurtosis(x, weights)
44            kurtosis_mod <- kurtosis(y, weights)
45
46            ref <- c(mean_ref, sd_ref, skewness_ref, kurtosis_ref)
47            mod <- c(mean_mod, sd_mod, skewness_mod, kurtosis_mod)
48            distribution <- data.frame(cbind(ref, mod), row.names = c("mean", "sd",
                   "skewness", "kurtosis"))
49
50            #reset weights to NULL, if only distributional values should be
                  weighted
51            if (!weight.all){
52              weights <- NULL
53            }
54
55            #calculate measures of similarity
56            coefs <- r2(ref = x, mod = y, weights = weights)
57            willmott <- willmott(ref = x, mod = y)
58
59            similarity = list(coefs = coefs, willmott = willmott)
60
61            #calculate measures of deviance, depending on selected type(s)
62            if (type == "absolute"){
63              absolute <- .absoluteMetrics(x, y, weights)
64
65              out <- list(weighted = weighted,
66                          distribution = distribution,
67                          absolute = absolute,
68                          similarity = similarity)
69            }
70            else if (type == "normalised"){
71              normalised <- .normalisedMetrics(x, y, weights)
72
73              out <- list(weighted=weighted,
74                          distribution = distribution,
75                          normalised = normalised,
76                          similarity = similarity)
77            }
```

```
78            else if (type == "all"){
79              absolute <- .absoluteMetrics(x, y, weights)
80              normalised <- .normalisedMetrics(x, y, weights)
81
82              out <- list(weighted=weighted,
83                          distribution = distribution,
84                          absolute = absolute,
85                          normalised = normalised,
86                          similarity = similarity)
87            }
88            #assign class "compareMetrics" (S3-style) to output list for print
                  method
89            class(out) <- "compareMetrics"
90
91            return(out)
92          }
93 )
94
95 #compute metrics function for two supplied data frames, i.e. one for reference data
       , one for model results
96 setMethod('computeMetrics', signature(x='data.frame',y='data.frame'),
97           function(x, y, weights=NULL, weight.all = FALSE, type = c("all", "
                absolute", "normalised"), ...) {
98
99            #stopifnot(type %in% c("absolute", "normalised", "all"))
100           type <- match.arg(type)
101
102           if (any(names(x) == "cell")){
103             x <- subset(x, select = -cell)
104           }
105           if (any(names(y) == "cell")){
106             y <- subset(y, select = -cell)
107           }
108           if (is.data.frame(weights) && any(names(weights) == "area") && any(
                  names(weights) == "weight")){
109             weights <- weights[,"area"] * weights[,"weight"]
110           }
111
112           ref   <- as.matrix(x)
113           model <- as.matrix(y)
114
```

```
115             if (is.data.frame(weights)){
116               if (nrow(weights) == nrow(ref) && ncol(weights) == ncol(ref)){
117                 weights <- as.matrix(weights)
118               } else {
119                 stop("Check dimensions of weights data frame.")
120               }
121             }
122             if (is.vector(weights)){
123               if (length(weights) == nrow(ref)){
124                 weights <- matrix(rep(weights, ncol(ref)), nrow = nrow(ref), byrow
                      = T)
125               }
126               else if (length(weights) != (nrow(ref) * ncol(ref))){
127                 stop("Check length of weights vector.")
128               }
129             }
130
131             ref <- as.vector(ref)
132             model <- as.vector(model)
133             weights <- as.vector(weights)
134
135             return(computeMetrics(x = ref, y = model, weights = weights, weight.all
                   = weight.all, type = type))
136
137           }
138 )
139
140 #compute metrics function for one list, e.g. output from harmonise function
141 setMethod('computeMetrics', signature(x='list',y='missing'),
142           function(x, use.weights = FALSE, weight.all = FALSE, type = c("all" ,"
                absolute", "normalised"), labels = NULL, ...) {
143
144             #stopifnot(type %in% c("absolute", "normalised", "all"))
145             type <- match.arg(type)
146
147             weights <- NULL
148             ll <- length(x)
149             if (.testList(x)){
150               out_list <- rep(list(list()), ll)
151               temp_list <- x[[1]]
152               if (length(temp_list) == 2){
```

```
153                 ref <- temp_list[[1]]
154                 model <- temp_list[[2]]
155                 out_list[[1]] <- computeMetrics(x = ref, y = model, weights =
                        weights, weight.all = weight.all, type = type)
156               }
157               if (length(temp_list) == 3){
158                 ref <- temp_list[[1]]
159                 model <- temp_list[[2]]
160                 if (use.weights){
161                   weights <- temp_list[[3]]
162                 }
163                 out_list[[1]] <- computeMetrics(x = ref, y = model, weights =
                        weights, weight.all = weight.all, type=type)
164               }
165               if (ll > 1){
166                 for (i in 2:ll){
167                   temp_list <- x[[i]]
168                   if (length(temp_list) == 2){
169                     ref <- temp_list[[1]]
170                     model <- temp_list[[2]]
171                     out_list[[i]] <- computeMetrics(x = ref, y = model, weights =
                            NULL, type=type)
172                   }
173                   if (length(temp_list) == 3){
174                     ref <- temp_list[[1]]
175                     model <- temp_list[[2]]
176                     if (use.weights){
177                       weights <- temp_list[[3]]
178                     }
179                     out_list[[i]] <- computeMetrics(x = ref, y = model, weights =
                            weights, weight.all = weight.all, type=type)
180                   }
181                 }
182               }
183               if (is.null(labels) || (length(labels) != ll)){
184                 labels <- names(x)
185                 if (is.null(labels)){
186                   labels <- paste("X",seq(1,ll),sep="")
187                 }
188                 names(out_list) <- labels
189               } else {
```

```
190                 names(out_list) <- labels
191               }
192
193               return(out_list)
194             }
195             else if (.testDataFrames(x)){
196               temp_list <- x
197               if (length(temp_list) == 2){
198                 ref <- temp_list[[1]]
199                 model <- temp_list[[2]]
200                 return(computeMetrics(x = ref, y = model, weights = NULL, type=type
                        ))
201               }
202               if (length(temp_list) == 3){
203                 ref <- temp_list[[1]]
204                 model <- temp_list[[2]]
205                 if (use.weights){
206                   weights <- temp_list[[3]]
207                 }
208                 return(computeMetrics(x = ref, y = model, weights = weights, weight
                        .all = weight.all, type=type))
209               }
210             } else {
211               stop("See 'computeMetrics' help for format of list argument.")
212             }
213           }
214 )
```

code/computeMetrics.R

Functions for several metrics.

Description

Several distributional, error and similarity measures. The list includes measures of deviation(absolute and normalized) and of similarity. Only the normalized deviation and similarity measures should be used for comparison of variables with different units. See "Details" section for formulae

Usage

```
mb(ref, mod, weights = NULL)

nmb(ref, mod, weights = NULL)

mae(ref, mod, weights = NULL)

cmae(ref, mod, weights = NULL)

smae(ref, mod, weights = NULL)

nmae(ref, mod, weights = NULL)

ncmae(ref, mod, weights = NULL)

nsmae(ref, mod, weights = NULL)

rmse(ref, mod, weights = NULL)

rcmse(ref, mod, weights = NULL)

rsmse(ref, mod, weights = NULL)

nrmse(ref, mod, weights = NULL)

nrcmse(ref, mod, weights = NULL)

nrsmse(ref, mod, weights = NULL)

r2(ref, mod, weights = NULL)

willmott(ref, mod)

skewness(x, weights = NULL)

kurtosis(x, weights = NULL)
```

Arguments

`ref`	vector. Reference data
`mod`	vector. Modeled data
`weights`	vector. If weights are supplied, weighted metrics will be calculated. Has to be of the same length as the other supplied data sets.
`x`	vector. Reference or model data

Details

In the following equations: N = sample size

Following metrics are implemented:

################################

Distributional measures

Skewness (`skewness`): *(1/N * ∑(x-mean(x))^3) / (sd(x))^3*

Kurtosis (`kurtosis`): *((1/N * ∑(x-mean(x))^4) / (sd(x))^4) - 3*

################################

Error metrics

#In the following equations: N = sample size

Mean bias (`mb`): *(∑(mod - ref)) / N*

Mean absolute error (`mae`): *(∑ | mod - ref |) / N*

Centred mean absolute error (`cmae`): *(∑ | mod - ref - median(ref - mod) |) / N*

Scaled mean absolute error (`smae`): *(∑ | r |) / N*, where r = residuals of ref and mod

Root mean squared error (`rmse`): *√(∑((mod - ref)^2) / N)*

Centred root mean squared error (`rcmse`): *√(∑((mod - ref - mean(ref - mod))^2) / (N-1))*

Scaled root mean squared error (`rsmse`): *√(∑(r^2) / (N-2))*, where r = residuals of ref and mod

#For interpretation of the above error metrics, refer to:

www.jstatsoft.org/v22/i08/paper

Normalised error metrics (`nmb`, `nmae`, `ncmae`, `nsmae`, `nrmse`, `nrcmse`, `nrsmse`): all normalised by the mean of the reference data

#Mean bias and mean absolute error have varying names in the literature, so please compare equations

################################

Measures of similarity

Ceofficient of determination (r2): Squared Pearson's correlation coefficient

Willmott's refined index of agreement 'dr' (`willmott`):

A = $\sum$ *| mod - ref |*

B = *2 * $\sum$ | ref - mean(ref) |*

A <= B: *dr = 1 - (A / B)*

A > B: *dr = (B / A) - 1*

See Also

`computeMetrics`

```
#all functions written by Jan Kowalewski
#### Error measures ####
#Mean bias
mb <- function(ref, mod, weights = NULL ){
  if (is.null(weights)){

    return(sum(mod - ref) / length(ref))

  } else {

    return(sum(weights * (mod - ref)) / sum(weights))

  }
}

#Normalised mean bias: -100 to +inf
#normalised by mean of reference data
nmb <- function(ref, mod, weights = NULL ){
  if (is.null(weights)){
    mean_o <- mean(ref)
    mb <- mb(ref = ref, mod = mod)
    return(mb / mean_o)

  } else {
    w_mean_o <- wtd.mean2(ref, weights)
    w_mb <- mb(ref = ref, mod = mod, weights = weights)
    return(w_mb / w_mean_o)

  }
}

#### Mean Absolute Error ####
#Mean absolute error
mae <- function(ref, mod, weights = NULL ){

  if (is.null(weights)){

    return(sum(abs(ref - mod)) / length(ref))

  } else {
```

```
41
42      return(sum(weights * abs(ref - mod)) / sum(weights))
43
44    }
45  }
46
47  #centred mean absolute error
48  cmae <- function(ref, mod, weights = NULL ){
49
50    if (is.null(weights)){
51
52      return(sum(abs(mod - ref - median(ref - mod)) / length(ref)))
53
54    } else {
55      w_cmae <- sum(weights * abs(mod - ref - Hmisc::wtd.quantile(ref - mod, weights,
            probs = .5))) / sum(weights)
56      return(w_cmae)
57
58    }
59  }
60
61  #scaled mean absolute error
62  smae <- function(ref, mod, weights = NULL ){
63    res <- residuals(lm(ref ~ mod))
64    if (is.null(weights)){
65
66      return(sum(abs(res)) / length(res))
67
68    } else {
69
70      return(sum(weights * abs(res)) / sum(weights))
71
72    }
73  }
74
75  ##normalised mean absolute error
76  #normalised by mean of reference data
77  nmae <- function(ref, mod, weights = NULL ){
78    if (is.null(weights)){
79      mean_o <- mean(ref)
80      mae <- mae(ref = ref, mod = mod)
```

```
81      return(mae / mean_o)
82
83    } else {
84      w_mean_o <- wtd.mean2(ref, weights)
85      w_mae <- mae(ref = ref, mod = mod, weights = weights)
86      return(w_mae / w_mean_o)
87
88    }
89  }
90
91  ##normalised centred mean absolute error
92  #normalised by mean of reference data
93  ncmae <- function(ref, mod, weights = NULL ){
94    if (is.null(weights)){
95      mean_o <- mean(ref)
96      cmae <- cmae(ref = ref, mod = mod)
97      return(cmae / mean_o)
98
99    } else {
100     w_mean_o <- wtd.mean2(ref, weights)
101     w_cmae <- cmae(ref = ref, mod = mod, weights = weights)
102     return(w_cmae / w_mean_o)
103
104   }
105 }
106
107 #normalised scaled (i.e. use residuals) mean absolute error
108 #normalised by mean of reference data
109 nsmae <- function(ref, mod, weights = NULL ){
110
111   if (is.null(weights)){
112     mean_o <- mean(ref)
113     smae <- smae(ref = ref, mod = mod)
114     return(smae / mean_o)
115
116   } else {
117     w_mean_o <- wtd.mean2(ref, weights)
118     w_smae <- smae(ref = ref, mod = mod, weights = weights)
119     return(w_smae / w_mean_o)
120
121   }
```

```
}

#### Root Mean Squared Error ####
#Root mean squared error
rmse <- function(ref, mod, weights = NULL ){

  if (is.null(weights)){

    return(sqrt(sum((ref - mod)^2) / length(ref)))

  } else {

    return(sqrt(sum(weights * ((ref - mod)^2)) / sum(weights)))

  }
}

#Root centered mean squared error
rcmse <- function(ref, mod, weights = NULL ){

  if (is.null(weights)){

    return(sqrt(sum((ref - mod - mean(ref - mod))^2) / (length(ref) - 1)))

  } else {

    return(sqrt(sum(weights * ((ref - mod - wtd.mean2(ref - mod, weights))^2)) / (
        sum(weights) - 1)))

  }
}

#Root scaled mean squared error
rsmse <- function(ref, mod, weights = NULL ){
  res <- residuals(lm(ref ~ mod))
  if (is.null(weights)){

    return(sqrt(sum((res)^2) / (length(res) - 2)))

  } else {

```

```
162     return(sqrt(sum(weights * (res^2)) / (sum(weights) - 2)))
163
164   }
165 }
166
167 #normalised root mean squared error
168 #normalised by mean of reference data
169 nrmse <- function(ref, mod, weights = NULL ){
170
171   if (is.null(weights)){
172     mean_o <- mean(ref)
173     rmse <- rmse(ref = ref, mod = mod)
174     return(rmse / mean_o)
175
176   } else {
177     w_mean_o <- wtd.mean2(ref, weights)
178     w_rmse <- rmse(ref = ref, mod = mod, weights = weights)
179     return(w_rmse / w_mean_o)
180
181   }
182 }
183
184 #normalised root centered mean squared error
185 #normalised by mean of reference data
186 nrcmse <- function(ref, mod, weights = NULL ){
187
188   if (is.null(weights)){
189     mean_o <- mean(ref)
190     rcmse <- rcmse(ref = ref, mod = mod)
191     return(rcmse / mean_o)
192
193   } else {
194     w_mean_o <- wtd.mean2(ref, weights)
195     w_rcmse <- rcmse(ref = ref, mod = mod, weights = weights)
196     return(w_rcmse / w_mean_o)
197
198   }
199 }
200
201 #normalised root scaled mean squared error
202 #normalised by mean of reference data
```

```
203 nrsmse <- function(ref, mod, weights = NULL ){
204
205   if (is.null(weights)){
206     mean_o <- mean(ref)
207     rsmse <- rsmse(ref = ref, mod = mod)
208     return(rsmse / mean_o)
209
210   } else {
211     w_mean_o <- wtd.mean2(ref, weights)
212     w_rsmse <- rsmse(ref = ref, mod = mod, weights = weights)
213     return(w_rsmse / w_mean_o)
214
215   }
216 }
217
218 #### Similarity measures ####
219 #Coefficient of Determination
220 r2 <- function(ref, mod, weights = NULL ){
221   if (is.null(weights)){
222     r <- wtd.cor2(ref, mod)
223     r2 <- (r[1])^2
224     out <- data.frame(cbind(r2 = r2, r = r[1], std.err = r[2], t.value = r[3], p.
          value = r[4]))
225     return(out)
226
227   } else {
228     r <- wtd.cor2(ref, mod, weights)
229     r2 <- (r[1])^2
230     out <- data.frame(cbind(r2 = r2, r = r[1], std.err = r[2], t.value = r[3], p.
          value = r[4]))
231     return(out)
232
233   }
234 }
235
236 #Willmott's refined index of model performance
237 #Willmott et al. 2012
238 willmott <- function(ref, mod){
239   o_bar <- mean(ref)
240   N <- sum(abs(mod - ref))
241   D <- 2 * sum(abs(ref - o_bar))
```

```
242   if (N <= D){
243     return(1 - (N/D))
244   } else {
245     return((D/N) - 1)
246   }
247 }
248 
249 #### Distributional measures ####
250 #Skewness
251 #measures asymmentry in value distribution
252 skewness <- function(x, weights = NULL){
253   if (is.null(weights)){
254     x_bar <- mean(x)
255     x_sd <- sd(x)
256   } else {
257     x_bar <- wtd.mean2(x, weights)
258     x_sd <- sqrt(wtd.var2(x, weights))
259   }
260 
261   return(sum((x - x_bar)^3) / (length(x) * (x_sd)^3))
262 }
263 
264 #Kurtosis
265 #measures pointedness of value distribution
266 kurtosis <- function(x, weights = NULL){
267   if (is.null(weights)){
268     x_bar <- mean(x)
269     x_sd <- sd(x)
270   } else {
271     x_bar <- wtd.mean2(x, weights)
272     x_sd <- sqrt(wtd.var2(x, weights))
273   }
274 
275   return((sum((x - x_bar)^4) / (length(x) * (x_sd)^4)) - 3)
276 }
```

code/metrics.R

Plot a Taylor Diagram.

Description

Plot a taylor diagram, showing the correlation, centered root-mean-square difference and standard deviation between a benchmark and model data set in one diagram.

Usage

```
taylorDiagram(x, y, ...)

## S4 method for signature 'vector,vector'
taylorDiagram(x, y, weights = NULL, add = FALSE,
  normalize = FALSE, label = NULL, pos = 3, col = "red", pch = 19,
  pos.cor = TRUE, xlab = "", ylab = "", main = "Taylor Diagram",
  show.gamma = TRUE, ngamma = 4, gamma.col = 8, sd.arcs = 0,
  ref.sd = FALSE, grad.corr.lines = c(0.2, 0.4, 0.6, 0.8, 0.9), pcex = 1,
  cex.axis = 1, mar = c(5, 4, 6, 6), coords = FALSE, ...)

## S4 method for signature 'data.frame,data.frame'
taylorDiagram(x, y, weights = NULL,
  match = FALSE, label = NULL, ...)

## S4 method for signature 'list,missing'
taylorDiagram(x, use.weights = FALSE,
  col.pal = NULL, add = FALSE, labels = NULL, pos = 3, ...)

## S4 method for signature 'list,list'
taylorDiagram(x, y, use.weights = FALSE,
  col.pal = NULL, add = FALSE, arrows = TRUE, labels = NULL, ...)
```

Arguments

`x`	vector, data.frame, list. A vector or data.frame of the benchmark data set or a list containing data frames of the benchmark and model data sets (and optionally weights). It can also be a list of lists containing model and benchmark data sets. See details
`y`	vector, data.frame, list, missing. A vector or data.frame of the benchmark data set or a list containing data frames of the benchmark and model data sets (and optionally weights). It can also be a list of lists containing model and benchmark data sets. See details
`...`	Additional arguments passed to `plot`
`weights`	vector, data.frame. Weights used to calculate weighted correlation and standard deviation.
`add`	logical. Add a point to an existing taylor diagram (TRUE) or plot new diagram (FALSE).
`normalize`	logical. Normalize models so that reference has a standard deviation of 1.
`label`	character. Text to label plotted points.
`pos`	numeric. Position of labels relative to point: 1 - below, 2 - left, 3 - above, 4 - right.
`col`	character, numeric. Color for displayed points
`pch`	numeric. Type of point to display
`pos.cor`	logical. Display only positive (TRUE) or all correlation values (FALSE)

xlab	character. Label for x-axis
ylab	character. Label for y-axis
main	character. Plot title
show.gamma	logical. Display standard deviation arcs around the reference point (only for pos.cor = TRUE)
ngamma	numeric. Number of gamma arcs to display
gamma.col	numeric. Color of gamma arcs (only for pos.cor = TRUE)
sd.arcs	numeric. Display arcs along standard deviation axes
ref.sd	logical. Display arc representing the reference standard deviation
grad.corr.lines	numeric. Values for the radial lines of correlation values
pcex	numeric. Character expansion for plotted points
cex.axis	numeric. Character expansion for axis text
mar	numeric. Margin values (only for pos.cor = TRUE)
coords	logical. Return polar coordinates of a plotted point.
match	logical. Should supplied data.frames be matched for dimensions and names.
use.weights	logical. Should the weights, contained in the list(s), be applied, when computing the metrics for the Taylor Diagram
col.pal	color palette. Color palette from e.g. RColorBrewer package. Length of color palette must match number of points to be plotted or longer.
labels	character, list. Character vector or list of 2 character vectors (for signature x = "list" , y="list") with same length as points to be plotted.
arrows	logical. Plot arrows between points of different model runs

Details

If the data sets are supplied as vectors or data frames, `x` is always the benchmark and y the model data set. Weights have to be supplied as a vector or data frame, having the same length (vector) or number of rows (data.frame) as the benchmark/model data sets.

If the data sets are supplied to as a list (e.g. the return value of `harmonise`) to `x`, and `y` is missing, the list has to contain a data frame for the benchmark, model and (optionally) weights data. Additionally, it is possible to supply a list of lists, each of which contains 2 or 3 data frames (benchmark, model and possibly weights). This approach should be used, when the relative performance of different models or variables is assessed.

If data sets are supplied in a list (i.e. as explained above) to `x` and `y`, arrows can be plotted between the respective data points in `x` and `y`. The arrow head always points from the data point in `x` to the data point in `y`.

Value

Polar coordinates of point (if coords = TRUE).

See Also

`harmonise`, `extract` and `plot`

Examples

```
## Not run:
#create empty LandGrid with 3 layers
ref <- landgrid()
```

```
model <- landgrid()

#add values
data <- runif(ncell(ref), 0.1, 0.9)
ref <- setVals(ref, data, layer=1)
model <- setVals(model, data + runif(ncell(model), 0.1, 0.4), layer=1)

#harmoise values in data sets
data_DEU <- harmonise(x = ref, y = model, dataset="gadm_adm1", spatial_unit="DEU")

taylorDiagram(data_DEU, label = "data_deu")

## End(Not run)
```

```
setGeneric("taylorDiagram", function(x,y,...)
  standardGeneric("taylorDiagram"))
#taylorDiagram base function for two supplied vectors
#original from plotrix-package: by Olivier Eterradossi with modifications by Jim Lemon
#further modifications (e.g. adding weighting factors): Christoph Mueller, PIK
#further modifications: Jan Kowalewski
setMethod('taylorDiagram', signature(x='vector',y='vector'),
          function(x, y, weights=NULL, add = FALSE, normalize = FALSE, label = NULL,
                   pos = 3, col = "red", pch = 19, pos.cor = TRUE, xlab = "", ylab = "",
                   main = "Taylor Diagram", show.gamma = TRUE, ngamma = 4, gamma.col = 8,
                   sd.arcs = 0, ref.sd = FALSE, grad.corr.lines = c(0.2, 0.4, 0.6, 0.8, 0.9),
                   pcex = 1, cex.axis = 1, mar = c(5, 4, 6, 6), coords = FALSE, ...) {

            grad.corr.full <- c(0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.99, 1)

            if (!is.null(weights) && !is.vector(weights)){
              stop("Weights have to be supplied as vector.")
            }

            R <- wtd.cor2(x, y, weights)[1]

            #harmonize NA in both sets
            subs <- as.logical(x)
            subs[] <- F
            subs[which(is.finite(x) & is.finite(y))] <- T

            ref <- subset(x, subs)
            model <- subset(y, subs)
            ww <- subset(weights, subs)

            sd.r <- sqrt(wtd.var2(ref,ww,na.rm=T))
            sd.f <- sqrt(wtd.var2(model,ww,na.rm=T))

            if (normalize) {
```

```
35              sd.f <- sd.f/sd.r
36              sd.r <- 1
37            }
38
39            ####### graphical parameters #######
40            maxsd <- 1.5 * max(sd.f, sd.r)
41            oldpar <- par("mar", "xpd", "xaxs", "yaxs")
42            if (!add) {
43              #pos.cor=TRUE shows only positive values of correlation
44              if (pos.cor) {
45                if (nchar(ylab) == 0)
46                  ylab = "Standard deviation"
47                par(mar = mar)
48                plot(0, xlim = c(0, maxsd), ylim = c(0, maxsd), xaxs = "i",
49                     yaxs = "i", axes = FALSE, main = main, xlab = xlab,
50                     ylab = ylab, type = "n", cex = cex.axis, ...)
51                if (grad.corr.lines[1]) {
52                  for (gcl in grad.corr.lines) lines(c(0, maxsd *
53                                                       gcl), c(0, maxsd * sqrt(1
                                                           - gcl^2)), lty = 3)
54                }
55                segments(c(0, 0), c(0, 0), c(0, maxsd), c(maxsd,
56                                                          0))
57                axis.ticks <- pretty(c(0, maxsd))
58                axis.ticks <- axis.ticks[axis.ticks <= maxsd]
59                axis(1, at = axis.ticks, cex.axis = cex.axis)
60                axis(2, at = axis.ticks, cex.axis = cex.axis)
61                if (sd.arcs[1]) {
62                  if (length(sd.arcs) == 1)
63                    sd.arcs <- axis.ticks
64                  for (sdarc in sd.arcs) {
65                    xcurve <- cos(seq(0, pi/2, by = 0.03)) * sdarc
66                    ycurve <- sin(seq(0, pi/2, by = 0.03)) * sdarc
67                    lines(xcurve, ycurve, col = "blue", lty = 3)
68                  }
69                }
70                if (show.gamma[1]) {
71                  if (length(show.gamma) > 1)
72                    gamma <- show.gamma
73                  else gamma <- pretty(c(0, maxsd), n = ngamma)[-1]
74                  if (gamma[length(gamma)] > maxsd)
```

```
75                    gamma <- gamma[-length(gamma)]
76                  labelpos <- seq(45, 70, length.out = length(gamma))
77                  for (gindex in 1:length(gamma)) {
78                    xcurve <- cos(seq(0, pi, by = 0.03)) * gamma[gindex] +
79                      sd.r
80                    endcurve <- which(xcurve < 0)
81                    endcurve <- ifelse(length(endcurve), min(endcurve) -
82                                         1, 105)
83                    ycurve <- sin(seq(0, pi, by = 0.03)) * gamma[gindex]
84                    maxcurve <- xcurve * xcurve + ycurve * ycurve
85                    startcurve <- which(maxcurve > maxsd * maxsd)
86                    startcurve <- ifelse(length(startcurve), max(startcurve) +
87                                           1, 0)
88                    lines(xcurve[startcurve:endcurve], ycurve[startcurve:endcurve],
89                          col = gamma.col)
90                    if (xcurve[labelpos[gindex]] > 0)
91                      boxed.labels2(xcurve[labelpos[gindex]], ycurve[labelpos[
                          gindex]],
92                                    gamma[gindex], border = FALSE)
93                  }
94                }
95                xcurve <- cos(seq(0, pi/2, by = 0.01)) * maxsd
96                ycurve <- sin(seq(0, pi/2, by = 0.01)) * maxsd
97                lines(xcurve, ycurve)
98                bigtickangles <- acos(seq(0.1, 0.9, by = 0.1))
99                medtickangles <- acos(seq(0.05, 0.95, by = 0.1))
100               smltickangles <- acos(seq(0.91, 0.99, by = 0.01))
101               segments(cos(bigtickangles) * maxsd, sin(bigtickangles) *
102                          maxsd, cos(bigtickangles) * 0.97 * maxsd, sin(
                               bigtickangles) *
103                          0.97 * maxsd)
104               par(xpd = TRUE)
105               if (ref.sd) {
106                 xcurve <- cos(seq(0, pi/2, by = 0.01)) * sd.r
107                 ycurve <- sin(seq(0, pi/2, by = 0.01)) * sd.r
108                 lines(xcurve, ycurve)
109               }
110               points(sd.r, 0, cex = pcex)
111               text(cos(c(bigtickangles, acos(c(0.95, 0.99)))) *
112                      1.05 * maxsd, sin(c(bigtickangles, acos(c(0.95,
113                                                          0.99)))) * 1.05 *
```

```
                                                                maxsd, c(seq
                                                                (0.1, 0.9, by
                                                                = 0.1),
114                                                                         0.95,
                                                                                0.9
                                                                                )
115             text(maxsd * 0.8, maxsd * 0.8, "Correlation", srt = 315)
116             segments(cos(medtickangles) * maxsd, sin(medtickangles) *
117                         maxsd, cos(medtickangles) * 0.98 * maxsd, sin(
                                medtickangles) *
118                         0.98 * maxsd)
119             segments(cos(smltickangles) * maxsd, sin(smltickangles) *
120                         maxsd, cos(smltickangles) * 0.99 * maxsd, sin(
                                smltickangles) *
121                         0.99 * maxsd)
122           }
123           else {
124             R <- wtd.cor2(x, y, ww)[1]
125             if (add == FALSE) {
126               #for consistent legend positioning
127               maxray <- 1.4
128               plot(c(-maxray, maxray), c(0, maxray), type = "n",
129                    asp = 1, bty = "n", xaxt = "n", yaxt = "n",
130                    xlab = xlab, ylab = ylab, main = main, cex = cex.axis)
131               discrete <- seq(180, 0, by = -1)
132               listepoints <- NULL
133               for (i in discrete) {
134                 listepoints <- cbind(listepoints, maxray *
135                                        cos(i * pi/180), maxray * sin(i * pi/
                                            180))
136               }
137               listepoints <- matrix(listepoints, 2, length(listepoints)/2)
138               listepoints <- t(listepoints)
139               lines(listepoints[, 1], listepoints[, 2])
140               lines(c(-maxray, maxray), c(0, 0))
141               lines(c(0, 0), c(0, maxray))
142               for (i in grad.corr.lines) {
143                 lines(c(0, maxray * i), c(0, maxray * sqrt(1 - i^2)), lty = 3)
144                 lines(c(0, -maxray * i), c(0, maxray * sqrt(1 - i^2)), lty = 3)
```

```
145
146                 }
147                 for (i in grad.corr.full) {
148                   text(1.05 * maxray * i, 1.05 * maxray * sqrt(1 -
149                                                                i^2), i, cex =
                                                                    0.6)
150                   text(-1.05 * maxray * i, 1.05 * maxray * sqrt(1 -
151                                                                 i^2), -i, cex =
                                                                      0.6)
152                 }
153                 seq.sd <- seq.int(0, 2 * maxray, by = (maxray/10))[-1]
154                 for (i in seq.sd) {
155                   xcircle <- sd.r + (cos(discrete * pi/180) *
156                                        i)
157                   ycircle <- sin(discrete * pi/180) * i
158                   for (j in 1:length(xcircle)) {
159                     if ((xcircle[j]^2 + ycircle[j]^2) < (maxray^2)) {
160                       points(xcircle[j], ycircle[j], col = "darkgreen",
161                              pch = ".")
162                       if (j == 10)
163                         text(xcircle[j], ycircle[j], signif(i,
164                                                             2), cex = 0.5, col =
                                                                  "darkgreen")
165                     }
166                   }
167                 }
168                 seq.sd <- seq.int(0, maxray, length.out = 5)
169                 for (i in seq.sd) {
170                   xcircle <- (cos(discrete * pi/180) * i)
171                   ycircle <- sin(discrete * pi/180) * i
172                   if (i)
173                     lines(xcircle, ycircle, lty = 3, col = "blue")
174                   text(min(xcircle), -0.03 * maxray, signif(i,
175                                                             2), cex = 0.5, col =
                                                                  "blue")
176                   text(max(xcircle), -0.03 * maxray, signif(i,
177                                                             2), cex = 0.5, col =
                                                                  "blue")
178                 }
179                 text(0, -0.08 * maxray, "Standard Deviation",
180                      cex = 0.7, col = "blue")
```

```
181                   text(0, -0.12 * maxray, "Centered RMS Difference",
182                        cex = 0.7, col = "darkgreen")
183                   points(sd.r, 0, pch = 22, bg = "darkgreen", cex = 1.1)
184                   text(0, 1.1 * maxray, "Correlation Coefficient",
185                        cex = 0.7)
186                 }
187                 S <- (2 * (1 + R))/(sd.f + (1/sd.f))^2
188               }
189             }
190             radius <- sd.f * R
191             angle <- sd.f * sin(acos(R))
192             points(radius, angle, pch = pch, col = col,
193                    cex = pcex)
194             cat("correlation",R,"SD",sd.f,"\n")
195             invisible(oldpar)
196             # do not return old par but the max SD to locate the legend
197             invisible(maxsd)
198             if (!is.null(label)){
199               text(radius, angle, labels=label, cex = 0.6 * pcex, pos = pos, font =
                      2)
200             }
201             if (coords){
202               return(matrix(data = c(radius, angle), ncol = 2))
203             }
204           }
205 )
206 #taylorDiagram function for two supplied data frames, i.e. reference data and model
      results
207 #written by Jan Kowalewski
208 setMethod('taylorDiagram', signature(x='data.frame',y='data.frame'),
209           function(x, y, weights=NULL, match = FALSE, label = NULL, ...) {
210
211             if (any(names(x) == "cell")){
212               x <- subset(x, select = -cell)
213             }
214             if (any(names(y) == "cell")){
215               y <- subset(y, select = -cell)
216             }
217             if (is.data.frame(weights) && any(names(weights) == "area") && any(
                    names(weights) == "weight")){
218               weights <- weights[,"area"] * weights[,"weight"]
```

```
219             }
220
221             if (match){
222               xdims <- dim(x)
223               ydims <- dim(y)
224               match_dims <- xdims == ydims
225               if (sum(match_dims) != length(xdims)){
226                 stop("Dimensions of data frames do not match.")
227               }
228               xnames <- names(x)
229               ynames <- names(y)
230               if (!is.null(xnames) && !is.null(ynames)){
231                 match_names <- xnames == ynames
232                 if(sum(match_names) != length(xnames)){
233                   stop("Names of data entries do not match.")
234                 }
235               } else {
236                 warning("Could not match data entry names -> NULL.")
237               }
238             }
239
240             ref   <- as.matrix(x)
241             model <- as.matrix(y)
242
243             if (is.data.frame(weights)){
244               if (nrow(weights) == nrow(ref) && ncol(weights) == ncol(ref)){
245                 weights <- as.matrix(weights)
246               } else {
247                 stop("Check dimensions of weights data frame.")
248               }
249             }
250             if (is.vector(weights)){
251               if (length(weights) == nrow(ref)){
252                 weights <- matrix(rep(weights, ncol(ref)), nrow = nrow(ref), byrow
                      = T)
253               }
254               else if (length(weights) != (nrow(ref) * ncol(ref))){
255                 stop("Check length of weights vector.")
256               }
257             }
258
```

```
259             ref <- as.vector(ref)
260             model <- as.vector(model)
261             weights <- as.vector(weights)
262
263             tD <- taylorDiagram(x = ref, y = model, weights = weights, label =
                    label, ...)
264             invisible(tD)
265           }
266 )
267 #taylorDiagram function for a supplied list, e.g. output from harmonise function
268 #written by Jan Kowalewski
269 setMethod('taylorDiagram', signature(x='list', y='missing'),
270           function(x, use.weights = FALSE, col.pal = NULL, add = FALSE, labels =
                NULL, pos = 3, ...) {
271             weights <- NULL
272             ll <- length(x)
273             if (is.null(col.pal)){
274               col.pal <- c(RColorBrewer::brewer.pal(8,"Accent"),RColorBrewer::
                    brewer.pal(12,"Paired"),RColorBrewer::brewer.pal(12,"Set3"),
                    RColorBrewer::brewer.pal(8,"Dark2"))
275             }
276             if (.testList(x)){
277               if (length(pos) == ll){
278                 stopifnot(pos %in% c(1,2,3,4))
279               } else {
280                 stopifnot(pos %in% c(1,2,3,4))
281                 pos <- rep(pos, ll)
282               }
283               if (is.vector(labels)){
284                 if (length(labels) != ll){
285                   warning("Labels do not match number of points and have been
                        replaced.")
286                   labels <- paste("X", seq(1,ll), sep="")
287                 }
288               }
289               else if (!is.null(labels) && !is.vector(labels)){
290                 warning("Labels were not supplied as character vector and have been
                      replaced.")
291                 labels <- paste("X", seq(1,ll), sep="")
292               }
293               temp_list <- x[[1]]
```

```
294             if (length(temp_list) == 2){
295               ref <- temp_list[[1]]
296               model <- temp_list[[2]]
297               taylorDiagram(x = ref, y = model, add = add, col = col.pal[1],
                      label = labels[1], pos = pos[1], ...)
298             }
299             if (length(temp_list) == 3){
300               ref <- temp_list[[1]]
301               model <- temp_list[[2]]
302               if (use.weights){
303                 weights <- temp_list[[3]]
304               }
305               taylorDiagram(x = ref, y = model, weights = weights, add = add, col
                    = col.pal[1], label = labels[1] , pos = pos[1], ...)
306             }
307             if (ll > 1){
308               for (i in 2:ll){
309                 temp_list <- x[[i]]
310                 if (length(temp_list) == 2){
311                   ref <- temp_list[[1]]
312                   model <- temp_list[[2]]
313                   taylorDiagram(x = ref, y = model, add = TRUE, col = col.pal[i],
                        label = labels[i], pos = pos[i], ...)
314                 }
315                 if (length(temp_list) == 3){
316                   ref <- temp_list[[1]]
317                   model <- temp_list[[2]]
318                   if (use.weights){
319                     weights <- temp_list[[3]]
320                   }
321                   taylorDiagram(x = ref, y = model, weights = weights, add = TRUE
                        , col = col.pal[i], label = labels[i], pos = pos[i], ...)
322                 }
323               }
324             }
325           }
326           else if (.testDataFrames(x)){
327             temp_list <- x
328             if (length(pos) == 1){
329               stopifnot(pos %in% c(1,2,3,4))
330             } else {
```

```
331                 pos <- pos[1]
332                 stopifnot(pos %in% c(1,2,3,4))
333               }
334               if (is.vector(labels)){
335                 if (length(labels) > 1){
336                   warning("Labels vector is longer than number of points. Only
                          first label is used.")
337                   labels <- labels[1]
338                 }
339               }
340               else if (!is.null(labels) && !is.vector(labels)){
341                 warning("Labels were not supplied as character vector and have been
                        replaced.")
342                 labels <- "X1"
343               }
344               if (length(temp_list) == 2){
345                 ref <- temp_list[[1]]
346                 model <- temp_list[[2]]
347                 taylorDiagram(x = ref, y = model, add = add, label = labels, pos =
                      pos, ...)
348               }
349               if (length(temp_list) == 3){
350                 ref <- temp_list[[1]]
351                 model <- temp_list[[2]]
352                 if (use.weights){
353                   weights <- temp_list[[3]]
354                 }
355                 taylorDiagram(x = ref, y = model, weights = weights, add = add,
                      label = labels, pos = pos, ...)
356               }
357             } else {
358               stop("See function help for format of list argument.")
359             }
360           }
361 )
362
363 #taylorDiagram function for two supplied lists, e.g. two outputs from harmonise
      function
364 #this function can be used two plot a Taylor diagram with arrows for improvement
      tracking
365 #written by Jan Kowalewski
```

```
366 setMethod('taylorDiagram', signature(x='list', y='list'),
367           function(x, y, use.weights = FALSE, col.pal = NULL, add = FALSE, arrows =
                  TRUE, labels = NULL, ...) {
368             weights <- NULL
369             weights2 <- NULL
370             xlength <- length(x)
371             ylength <- length(y)
372             if (is.null(col.pal)){
373               col.pal <- c(RColorBrewer::brewer.pal(8,"Accent"),RColorBrewer::
                      brewer.pal(12,"Paired"),
374                            RColorBrewer::brewer.pal(12,"Set3"),RColorBrewer::brewer
                                .pal(8,"Dark2"))
375             }
376             if (xlength == ylength){
377               if (.testList(x) && .testList(y)){
378                 if (is.list(labels)){
379                   if (length(labels) == 2){
380                     l1 <- length(labels[[1]])
381                     l2 <- length(labels[[2]])
382                     if (l1 != xlength || l2 != ylength){
383                       warning("At least one label vector in labels list does not
                              match number of points. Labels have been replaced.")
384                       labels1 <- paste("X", seq(1,xlength), sep="")
385                       labels2 <- paste("Y", seq(1,xlength), sep="")
386                       labels <- list(labels1, labels2)
387                     }
388                   } else {
389                     warning("List of label vectors has wrong length and been
                            replaced.")
390                     labels1 <- paste("X", seq(1,xlength), sep="")
391                     labels2 <- paste("Y", seq(1,xlength), sep="")
392                     labels <- list(labels1, labels2)
393                   }
394                 }
395                 else if (is.vector(labels)){
396                   if (length(labels) != (xlength + ylength)){
397                     warning("Length of label vector does not match number of points
                            . Labels have been replaced.")
398                     labels1 <- paste("X", seq(1,xlength), sep="")
399                     labels2 <- paste("Y", seq(1,xlength), sep="")
400                     labels <- list(labels1, labels2)
```

```
401                 } else {
402                   labels1 <- labels[1:xlength]
403                   labels2 <- labels[xlength+1 : xlength+ylength]
404                   labels <- list(labels1, labels2)
405                 }
406               }
407               else if (!is.null(labels)) {
408                 warning("Labels should be supplied as character vector or list of
                        character vectors.")
409                 labels <- list(labels1 = paste("X", seq(1,xlength), sep=""),
                        labels2 = paste("Y", seq(1,xlength), sep=""))
410               }
411
412               xlabels <- labels[[1]]
413               temp_list <- x[[1]]
414               if (length(temp_list) == 2){
415                 ref <- temp_list[[1]]
416                 model <- temp_list[[2]]
417                 x1 <- taylorDiagram(x = ref, y = model, add = add, col = col.pal
                        [1], coords = TRUE, label = xlabels[1], ...)
418               }
419               if (length(temp_list) == 3){
420                 ref <- temp_list[[1]]
421                 model <- temp_list[[2]]
422                 if (use.weights){
423                   weights <- temp_list[[3]]
424                 }
425                 x1 <- taylorDiagram(x = ref, y = model, weights = weights, add =
                        add, col = col.pal[1], coords = TRUE, label = xlabels[1],
                        ...)
426               }
427               if (xlength > 1){
428                 xcoords <- matrix(rep(NA,xlength*2),nrow = xlength)
429                 xcoords[1, ] <- x1
430                 ycoords <- matrix(rep(NA,ylength*2),nrow = ylength)
431                 for (i in 2:xlength){
432                   temp_list <- x[[i]]
433                   if (length(temp_list) == 2){
434                     ref <- temp_list[[1]]
435                     model <- temp_list[[2]]
436                     xc <- taylorDiagram(x = ref, y = model, add = TRUE, col = col
```

```
                .pal[i], coords = TRUE, label = xlabels[i], ...)
437               xcoords[i, ] <- xc
438             }
439             if (length(temp_list) == 3){
440               ref <- temp_list[[1]]
441               model <- temp_list[[2]]
442               if (use.weights){
443                 weights <- temp_list[[3]]
444               }
445               xc <- taylorDiagram(x = ref, y = model, weights = weights,
                      add = TRUE, col = col.pal[i], coords = TRUE, label =
                      xlabels[i], ...)
446               xcoords[i, ] <- xc
447             }
448           }
449           ylabels <- labels[[2]]
450           for (j in 1:ylength){
451             temp_list2 <- y[[j]]
452             if (length(temp_list2) == 2){
453               ref2 <- temp_list2[[1]]
454               model2 <- temp_list2[[2]]
455               yc <- taylorDiagram(x = ref2, y = model2, add = TRUE, col =
                      col.pal[j], coords = TRUE, label = ylabels[j], pch = 17,
                      ...)
456               ycoords[j, ] <- yc
457               if (arrows){
458                 arrows(xcoords[j, 1], xcoords[j, 2], ycoords[j, 1], ycoords
                        [j, 2], code = 2)
459               }
460             }
461             if (length(temp_list2) == 3){
462               ref2 <- temp_list2[[1]]
463               model2 <- temp_list2[[2]]
464               if (use.weights){
465                 weights2 <- temp_list[[3]]
466               }
467               yc <- taylorDiagram(x = ref2, y = model2, weights = weights2,
                      add = TRUE, col = col.pal[j], coords = TRUE, label =
                      ylabels[j], pch = 17, ...)
468               ycoords[j, ] <- yc
469               if (arrows){
```

```
470                     arrows(xcoords[j, 1], xcoords[j, 2], ycoords[j, 1], ycoords
                            [j, 2], code = 2)
471                   }
472                 }
473               }
474             }
475           }
476           else if (.testDataFrames(x) && .testDataFrames(y)){
477             if (is.list(labels)){
478               if (length(labels) == 2){
479                 l1 <- length(labels[[1]])
480                 l2 <- length(labels[[2]])
481                 if (l1 != 1 || l2 != 1){
482                   warning("At least one label vector in labels list does not
                          match number of points. Labels have been replaced.")
483                   labels <- list("X1", "Y1")
484                 }
485               } else {
486                 warning("List of label vectors has wrong length and been
                        replaced.")
487                 labels <- list("X1", "Y1")
488               }
489             }
490             else if (is.vector(labels)){
491               if (length(labels) != 2){
492                 warning("Length of label vector does not match number of points
                        . Labels have been replaced.")
493                 labels <- list("X1", "Y1")
494               } else {
495                 labels <- list(labels[1], labels[2])
496               }
497             }
498             else if (!is.null(labels)) {
499               warning("Labels should be supplied as character vector or list of
                      character vectors.")
500               labels <- list("X1", "Y1")
501             }
502
503             temp_list <- x
504             if (length(temp_list) == 2){
505               ref <- temp_list[[1]]
```

```
506                   model <- temp_list[[2]]
507                   xc <- taylorDiagram(x = ref, y = model, add = add, coords = TRUE,
                          label = labels[[1]], ...)
508                 }
509                 if (length(temp_list) == 3){
510                   ref <- temp_list[[1]]
511                   model <- temp_list[[2]]
512                   if (use.weights){
513                     weights <- temp_list[[3]]
514                   }
515                   xc <- taylorDiagram(x = ref, y = model, weights = weights, add =
                        add, coords = TRUE, label = labels[[1]], ...)
516                 }
517                 temp_list2 <- y
518                 if (length(temp_list2) == 2){
519                   ref2 <- temp_list2[[1]]
520                   model2 <- temp_list2[[2]]
521                   yc <- taylorDiagram(x = ref2, y = model2, add = TRUE, coords =
                        TRUE, label = labels[[2]], pch = 17, ...)
522                   if (arrows){
523                     arrows(xc[, 1], xc[, 2], yc[, 1], yc[, 2], code = 2)
524                   }
525                 }
526                 if (length(temp_list2) == 3){
527                   ref2 <- temp_list2[[1]]
528                   model2 <- temp_list2[[2]]
529                   if (use.weights){
530                     weights2 <- temp_list[[3]]
531                   }
532                   yc <- taylorDiagram(x = ref2, y = model2, weights = weights2, add
                        = TRUE, coords = TRUE, label = labels[[2]], pch = 17, ...)
533                   if (arrows){
534                     arrows(xc[, 1], xc[, 2], yc[, 1], yc[, 2], code = 2)
535                   }
536                 }
537               }
538             }
539           }
540 )
```

code/taylorDiagram.R

Appendix B

Use-case documentation

Code is listed in the following order:

1. LPJmL and Miami model preparation
2. EMDI and MODIS NPP reference data preparation
3. Use-case 1-1: Comparison with EMDI NPP
4. Use-case 1-1: Comparison with MODIS NPP
5. Use-case 2: GPP improvement tracking

LPJmL and Miami model data preparation

```
####Example script of LPJmL and Miami model dataset preparation####
#examples are shown for the period 2000 - 2005
#dataset preparation process for the period 1931 - 1996 (comprison with EMDI data)
#is equivalent

ff_lu <- ".../Landuse/"

#load grassland masks (annual grassland landuse data)
fn_lu_annual <- paste(ff_lu,"mask_grassland_per_year_2000_2005.nc",sep="")
grassland_per_year <- readLandMark(file_name = fn_lu_annual)

#load grassland mask for sum of years 2000 - 2005
fn_lu_average <- paste(ff,"mask_grassland_2000_2005.nc",sep="")
grassland <- readLandMark(file_name = fn_lu_average)

######## LPJML OUTPUT PREPARATION ########

ff_in <- ".../Daten/"
ff_out <- ".../Daten/lpjml_npp_mean/"
fn_lpjml_npp <- "/mnpp.bin"
years <- 2000:2005
nyears <- length(years)

### read data, calculate mean values, mask data, write mean npp
#X: scenario identifier - can be replaced by 0,1,etc.
file_name <- paste(ff_in,"output_grass_X",fn, sep="")
npp_X <- readLPJBin(file_name = file_name, wyears = years, years = 109,
                    time_unit = "months", z_dim = "time", ncells = 67420)
f <- 1
t <- 12
#number of months
nm <- 12

layer_list <- list()

for (i in 1:nyears){
  lyrs <- f:t
  layer_list[[i]] <- sum(npp_0[[lyrs]])
  f <- f+nm
```

```
  t <- t+nm
}
default_npp <- brick(layer_list) * grassland_per_year
mean_npp_X <- mean(default_npp, na.rm = TRUE)
mean_npp_X <- mean_npp_X * grassland
mean_npp_X <- landgrid(mean_npp_X,
                       title = "Weighted mean NPP - default",
                       standard_name = "npp_default_interannual_mean_2000_2005",
                       unit = "g C m^-2",
                       z_dim = "category",
                       z_names = "mean_2000_2005")

writeLandMark(mean_npp_X, file_name = paste(ff_out,"grassX_npp_interannual_mean_
    2000_2005", sep=""))

######## MIAMI MODEL DATA PREPARATION ########
ff_out <- ".../Daten/miami_model/"
#read climate data
ff_clim <- ".../Daten/Climate/"
fn <- paste(ff_clim,"prec_2000_2005.nc",sep="")
prec <- readLandMark(file_name = fn)
fn <- paste(ff_clim,"temp_2000_2005.nc",sep="")
temp <- readLandMark(file_name = fn)

#function to calculate miami model data (Lieth, 1972)
miami_model <- function(temp, prec, factor = 0.45){
  NPPT <- 3000 / (1 + exp(1.315 - 0.119 * temp))
  NPPP <- 3000 * (1 - exp(-0.000664 * prec))
  NPPT_vals <- values(NPPT)
  NPPP_vals <- values(NPPP)
  NPP_vals <- matrix(nrow=nrow(NPPT_vals),ncol=ncol(NPPT_vals))
  for (i in 1:ncol(NPPT_vals)){
    NPP_vals[,i] <- pmin(NPPT_vals[,i], NPPP_vals[,i])
  }
  NPP <- landgrid(nl=ncol(NPP_vals))
  NPP <- setValues(NPP, NPP_vals)

  return(NPP * factor)
}

miami_npp_annual <- miami_model(temp, prec)
```

```
miami_npp_annual <- miami_npp_annual * grassland_per_year
miami_npp_annual_mean <- mean(miami_npp_annual_mean, na.rm = TRUE)
miami_npp_annual_mean <- miami_npp_annual_mean * grassland
miami_npp_annual_mean <- landgrid(miami_npp_annual_mean,
                          title = "Miami NPP mean annual total",
                          standard_name = "mean_npp_miami_model_2000_2005",
                          unit = "g C yr-1 m-2",
                          z_dim = "category",
                          z_names = "mean_2000_2005")

writeLandMark(miami_npp_annual_mean, file_name = "miami_npp_interannual_mean_2000_
    2005.nc", file_folder = ff_out)
```

code/lpjml_miami_preparation.R

EMDI and MODIS reference data preparation

```
####Example script of EMDI and MODIS NPP reference dataset preparation####

####### EMDI DATASET PREPARATION #########
ff_in <- ".../Daten/EMDI/NPP_tables/"
ff_out <- ".../Daten/EMDI/"
fn_emdi <- paste(ff_in,"emdi_grassland_npp.csv",sep="")
emdi_grassland_npp <- read.csv(fn)
emdi_npp_annual_mean <- landpoint(subset(emdi_grassland_npp, select = c(x,y,npp)))
emdi_npp_annual_mean <- setMetadata(emdi_npp_annual_mean, title = "NPP annual total
    ",
                         unit = "g C yr-1 m-2",
                         standard_name = "site_npp_annual_1931_1996",
                         source_name = "EMDI")

writeLandMark(mean_npp_X, file_name = paste(ff_out,"emdi_npp_interannual_mean_1931_
    1996", sep=""))

####### MODIS DATASET PREPARATION #########
ff <- ".../Daten/modis_gpp_npp/"
years <- 2000:2005
image_list <- list()
i <- 1
for(year in years){
  image <- raster(paste(ff,"MOD17A3_Science_NPP_",year,".tif",sep=""))
  image <- image * 0.1
  image <- clamp(image, -1000, 6000, useValues = FALSE)
  image_list[[i]] <- aggregate(image, fact = 60, fun = mean, na.rm=TRUE)
  i <- i+1
}

npp_modis <- brick(image_list[[1]], image_list[[2]], image_list[[3]], image_list
    [[4]], image_list[[5]], image_list[[6]])
mask <- crop(lpjmlinfo$full_raster, npp_modis)
npp_modis <- mask(npp_modis, mask)
modis_npp_annual <- landgrid(npp_modis)
modis_npp_annual <- setMetadata(modis_npp_annual,
                        unit = "g C m^-2",
                        standard_name = "annual_net_primary_production_total",
                        title = "Annual NPP",
```

```
                        reference_time = "2000-01-01",
                        time_unit = "years",
                        z_dim = "time",
                        scale_factor = 10,
                        source_name = "MODIS NPP - University of Montana",
                        references = "http://www.ntsg.umt.edu/project/mod17")


ff_lu <- ".../Landuse/"

#load grassland masks (annual grassland landuse data)
fn_lu_annual <- paste(ff_lu,"mask_grassland_per_year_2000_2005.nc",sep="")
grassland_per_year <- readLandMark(file_name = fn_lu_annual)

#load grassland mask for sum of years 2000 - 2005
fn_lu_average <- paste(ff,"mask_grassland_2000_2005.nc",sep="")
grassland <- readLandMark(file_name = fn_lu_average)

modis_npp_annual <- modis_npp_annual * grassland_per_year
modis_npp_annual_mean <- mean(modis_npp_annual_mean, na.rm=TRUE)
modis_npp_annual_mean <- modis_npp_annual_mean * grassland

modis_npp_annual_mean <- landgrid(modis_npp_annual_mean, values = TRUE,
                              title = "MODIS Grassland NPP - annual",
                              standard_name = "npp_weighted_annual_grassland_2000_
                                  2005",
                              unit = "g C m^-2",
                              z_dim = "category",
                              z_names = "mean_2000_2005")

writeLandMark(modis_annual_mean, file_name = "modis_npp_grassland_annual_mean",
    file_folder = ff)
```

code/reference_preparation.R

Use-case 1-1: Comparison with EMDI NPP

```
##### CASE 1: comparison with EMDI NPP #####
#set paths
ff_miami <- ".../Daten/miami_model/"
ff_lpjml <- ".../Daten/lpjml_npp_mean/"
ff_emdi <- ".../Daten/EMDI/"
ff_out <- ".../benchmarking_results/EMDI/"

#model and benchmark data
miami_npp <- readLandMark(paste(ff_miami,"miami_npp_interannual_mean_1931_1996.nc",
     sep=""))
grass0_npp <- readLandMark(paste(ff_lpjml,"grass0_npp_interannual_mean_1931_1996.nc
    ", sep=""))
grass2_npp <- readLandMark(paste(ff_lpjml,"grass2_npp_interannual_mean_1931_1996.nc
    ", sep=""))
#refernce data
emdi_npp <- readLandMark(paste(ff_emdi,"emdi_npp_interannual_mean_1931_1996.nc",sep
    =""))

#harmonise datasets
miami_emdi_npp <- harmonise(x = emdi_npp, y = miami_npp)
grass0_emdi_npp <- harmonise(x = emdi_npp, y = grass0_npp)
grass2_emdi_npp <- harmonise(x = emdi_npp, y = grass2_npp)

emdi_sites <- list(grass0_emdi_npp,
                   grass2_emdi_npp,
                   miami_emdi_npp)

#compute metrics for comparison with EMDI
metrics_emdi <- computeMetrics(emdi_sites,
                               use.weights=TRUE,
                               weight.all = TRUE,
                               type = "absolute",
                               label = c("default",
                                         "dgrazing",
                                         "miami"))

#plot taylor diagram for comparison with EMDI
#color of points
colors <- c(RColorBrewer::brewer.pal(5,"Set1")[2:3], "red")
```

```
#point labels
labels <- c("1","2","B")
#label positions
pos <- c(4,2,3)
fn <- paste(ff_out,"taylor_EMDI_sites.jpeg",sep="")
jpeg(fn, height=7*300, width=8*300, res=300, pointsize=9)
taylorDiagram(scenarios_sites,
              normalize=FALSE,
              use.weights = TRUE,
              labels = labels,
              pos = pos,
              ref.sd = TRUE,
              col.pal=colors,
              xlab = expression('NPP (gC m'^'-2'*'year'^'-1'*')'),
              pcex = 2,
              main = "Performance against EMDI class A sites")
# add text and legend
text(138, -8, labels = "reference", font = 2, cex = 0.8)
legend(250, 350, legend=c(expression("1: LPJmL "*italic(D)),
                          expression("2: LPJmL "*italic(G[D])),
                          "B: Miami model"),
       pch=19, col = colors, cex = 1.2, pt.cex = 1.2, text.font = 2)
dev.off()
```

code/emdi_comparison.R

Use-case 1-2: Comparison with MODIS NPP

```
##### CASE 2: comparison with MODIS NPP #####
#set paths
ff_miami <- ".../Daten/miami_model/"
ff_lpjml <- ".../Daten/lpjml_npp_mean/"
ff_modis <- ".../Daten/modis_gpp_npp/"
ff_out <- ".../benchmarking_results/MODIS/"

#model and benchmark data
miami_npp <-  readLandMark(paste(ff_miami,"miami_npp_interannual_mean_2000_2005.nc"
    , sep =""))
grass0_npp <- readLandMark(paste(ff_models,"grass0_npp_interannual_mean_2000_2005.
    nc", sep=""))
grass2_npp <- readLandMark(paste(ff_models,"grass2_npp_interannual_mean_2000_2005.
    nc", sep=""))
#reference data
modis_npp <- readLandMark(paste(ff_modis,"modis_npp_grassland_annual_mean.nc",sep="
    "))

##### CASE 2a: global comparison with MODIS NPP #####

#harmonise datasets - global scale
miami_modis_npp <- harmonise(x = modis_npp, y = miami_npp)
grass0_modis_npp <- harmonise(x = modis_npp, y = grass0_npp)
grass2_modis_npp <- harmonise(x = modis_npp, y = grass2_npp)

modis_global <- list(grass0_modis_npp, grass2_modis_npp, miami_modis_npp)

#compute metrics for global comparison with MODIS
metrics_modis_global <- computeMetrics(modis_global,
                                       use.weights=TRUE,
                                       weight.all = TRUE,
                                       type = "absolute",
                                       label = c("default", "dgrazing", "miami"))

#plot taylor diagram for global comparison with MODIS
#color of points
colors <- c(RColorBrewer::brewer.pal(5,"Set1")[2:3],"red")
#point labels
labels <- c("1", "2", "B")
```

```
#label positions
pos <- c(3,3,3)
fn <- paste(ff_out,"taylor_modis_global.jpeg",sep="")
jpeg(fn, height=7*300, width=8*300, res=300, pointsize=9)
taylorDiagram(modis_global,
              normalize=FALSE,
              use.weights = TRUE,
              labels = labels,
              pos = pos,
              xlab = expression('NPP (gC m'^'-2'*'year'^'-1'*')'),
              col.pal=colors,
              ref.sd = TRUE,
              pcex = 2,
              main = "Global performance against MODIS NPP")
# add text and legend
text(370, -15, labels = "reference", font = 2, cex = 0.8)
legend(410, 570, legend=c(expression("1: LPJmL "*italic(D)),
                          expression("2: LPJmL "*italic(G[D])),
                          "B: Miami model"),
       pch=19, col = colors, cex = 1.2, pt.cex = 1.2, text.font = 2) #, bty = "n")
dev.off()

##### CASE 2b: regional comparison with MODIS NPP #####

#harmonise datasets for Mongolia, Argentina, Kazakhstan, Australia
miami_modis_mng <- harmonise(x = modis_npp, y = miami_npp, FUN=getGeodata, database
     = "gadm_adm0", spatial_unit = "MNG")
miami_modis_arg <- harmonise(x = modis_npp, y = miami_npp, FUN=getGeodata, database
     = "gadm_adm0", spatial_unit = "ARG")
miami_modis_kaz <- harmonise(x = modis_npp, y = miami_npp, FUN=getGeodata, database
     = "gadm_adm0", spatial_unit = "KAZ")

grass0_modis_mng <- harmonise(x = modis_npp, y = grass0_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "MNG")
grass0_modis_arg <- harmonise(x = modis_npp, y = grass0_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "ARG")
grass0_modis_kaz <- harmonise(x = modis_npp, y = grass0_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "KAZ")

grass2_modis_mng <- harmonise(x = modis_npp, y = grass2_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "MNG")
```

```
grass2_modis_arg <- harmonise(x = modis_npp, y = grass2_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "ARG")
grass2_modis_kaz <- harmonise(x = modis_npp, y = grass2_npp, FUN=getGeodata,
    database = "gadm_adm0", spatial_unit = "KAZ")

modis_regional <- list(grass0_modis_mng, grass2_modis_mng,
                       grass0_modis_arg, grass2_modis_arg,
                       grass0_modis_kaz, grass2_modis_kaz,
                       miami_modis_mng, miami_modis_arg, miami_modis_kaz)

#compute metrics for global comparison with MODIS
metrics_modis_regional <- computeMetrics(modis_regional,
                                         use.weights=TRUE,
                                         weight.all = TRUE,
                                         type = "all",
                                         label = c( "default-mng", "dgrazing-mng",
                                              "default-arg", "dgrazing-arg",
                                              "default-kaz", "dgrazing-kaz",
                                              "miami-mng","miami-arg","miami-kaz"))

#plot taylor diagram for regional comparison with MODIS
#colors of points
col_miami <- RColorBrewer::brewer.pal(9,"Reds")[c(6,7,8)]
col_mng <- RColorBrewer::brewer.pal(9,"Blues")[c(5,7)]
col_arg <- RColorBrewer::brewer.pal(9,"Purples")[c(5,7)]
col_kaz <- RColorBrewer::brewer.pal(9,"Greens")[c(5,7)]
colors <- c(col_mng, col_arg, col_kaz, col_miami)
#point labels
labels <- c( "MNG1", "MNG2",
             "ARG1", "ARG2",
             "KAZ1", "KAZ2",
             "B_MNG", "B_ARG", "B_KAZ"
)
#label positions
pos <- c(3, 2,
         3, 3,
         3, 2,
         1, 2, 4)

fn <- paste(ff_out,"taylor_modis_regional.jpeg",sep="")
jpeg(fn, height=7*300, width=8*300, res=300, pointsize=9)
```

```
109 taylorDiagram(scenarios_modis_regions,
110               normalize=TRUE,
111               labels = labels,
112               use.weights = TRUE,
113               pos = pos,
114               ref.sd = TRUE,
115               xlab = "Normalised reference standard deviation",
116               ylab = "Normalised model standard deviation",
117               col.pal=colors,
118               pcex = 2,
119               main = "Regional performance against MODIS NPP")
120 text(1, -0.06, labels = "reference", font = 2, cex = 0.8)
121 # add text and legend
122 legend(2.35, 3.2, legend=c(expression("MNG1: Mongolia "*italic(D)),expression("MNG2
        : Mongolia "*italic(G[D])),
123                            expression("ARG1: Argentina "*italic(D)), expression("
                                   ARG2: Argentina "*italic(G[D])),
124                            expression("KAZ1: Kazakhstan "*italic(D)), expression("
                                   KAZ2: Kazakhstan "*italic(G[D])),
125                            "B_MNG: Mongolia Miami model", "B_ARG: Argentina Miami
                                   model", "B_KAZ: Kazakhstan Miami model"),
126        pch=19, col = colors, cex = 0.9, pt.cex = 1.2, text.font = 2)
127 dev.off()
```

code/modis_comparison.R

Use-case 2: GPP improvement tracking

```
1  ##### USE-CASE 2: GPP improvement tracking #####
2  ff_out <- ".../Daten/benchmarking_results/improvement_tracking/"
3  setwd(".../Daten/mcmc_parameter_optimization/")
4
5  #GPP results from standard and optimal parameter set simulations are available in .
       RData databases for each FluxNet station separately
6  #Data for each model run and station is loaded into R, and converted to a list that
        resembles an output list from harmonise
7  #Taylor diagram with arrows are plotted to investigate performance improvements
8
9  #######   IT-MBo optimal (Bestrun..) and standard paramter (Standardrun...) runs
       #######
10 load("hfree/Bestrun_IT-MBo.RData")
11 bestrun_mbo <- cost_best #observations (FluxNet) and results for run with optimal
       parameter set
12 gpp_id <- which(bestrun_mbo$residuals[,1] == "GPP_f") #select only GPP results
13 o <- order(bestrun_mbo$residuals[gpp_id,2]) #put in order of time
14 obs_gpp_best_mbo <- bestrun_mbo$residuals[gpp_id,c(2,3)] #store observations in
       separate variable
15 obs_gpp_best_mbo <- t(obs_gpp_best_mbo[o,]$obs) #transpose matrix
16 mod_gpp_best_mbo <- bestrun_mbo$residuals[gpp_id,c(2,4)] #store model results in
       separate variable
17 mod_gpp_best_mbo <- t(mod_gpp_best_mbo[o,]$mod) #transpose matrix
18 #convert to list, similar to harmonise output
19 bestrun_mbo <- list(reference=as.data.frame(obs_gpp_best_mbo), model=as.data.frame(
       mod_gpp_best_mbo))
20
21 load("hfree/Standardrun_IT-MBo.RData")
22 stand_mbo <- cost_stan #observations (FluxNet) and results for run with standard
       parameter set
23 gpp_id <- which(stand_mbo$residuals[,1] == "GPP_f") #as above
24 o <- order(stand_mbo$residuals[gpp_id,2])
25 obs_gpp_stand_mbo <- stand_mbo$residuals[gpp_id,c(2,3)]
26 obs_gpp_stand_mbo <- t(obs_gpp_stand_mbo[o,]$obs)
27 mod_gpp_stand_mbo <- stand_mbo$residuals[gpp_id,c(2,4)]
28 mod_gpp_stand_mbo <- t(mod_gpp_stand_mbo[o,]$mod)
29 stand_mbo <- list(reference=as.data.frame(obs_gpp_stand_mbo), model=as.data.frame(
       mod_gpp_stand_mbo))
30
```

```
31 ######   US-ARc optimal (Bestrun..) and standard paramter (Standardrun...) runs
       #######
32 load("hfree/Bestrun_US-ARc.RData")
33 bestrun_arc <- cost_best
34 gpp_id <- which(bestrun_arc$residuals[,1] == "GPP_f")
35 o <- order(bestrun_arc$residuals[gpp_id,2])
36 obs_gpp_best_arc <- bestrun_arc$residuals[gpp_id,c(2,3)]
37 obs_gpp_best_arc <- t(obs_gpp_best_arc[o,]$obs)
38 mod_gpp_best_arc <- bestrun_arc$residuals[gpp_id,c(2,4)]
39 mod_gpp_best_arc <- t(mod_gpp_best_arc[o,]$mod)
40 bestrun_arc <- list(reference=as.data.frame(obs_gpp_best_arc), model=as.data.frame(
       mod_gpp_best_arc))
41
42 load("hfree/Standardrun_US-ARc.RData")
43 stand_arc <- cost_stan
44 gpp_id <- which(stand_arc$residuals[,1] == "GPP_f")
45 o <- order(stand_arc$residuals[gpp_id,2])
46 obs_gpp_stand_arc <- stand_arc$residuals[gpp_id,c(2,3)]
47 obs_gpp_stand_arc <- t(obs_gpp_stand_arc[o,]$obs)
48 mod_gpp_stand_arc <- stand_arc$residuals[gpp_id,c(2,4)]
49 mod_gpp_stand_arc <- t(mod_gpp_stand_arc[o,]$mod)
50 stand_arc <- list(reference=as.data.frame(obs_gpp_stand_arc), model=as.data.frame(
       mod_gpp_stand_arc))
51
52 #######   US-FR2 optimal (Bestrun..) and standard paramter (Standardrun...) runs
       #######
53 load("hfree/Bestrun_US-FR2.RData")
54 bestrun_fr2 <- cost_best
55 gpp_id <- which(bestrun_fr2$residuals[,1] == "GPP_f")
56 o <- order(bestrun_fr2$residuals[gpp_id,2])
57 obs_gpp_best_fr2 <- bestrun_fr2$residuals[gpp_id,c(2,3)]
58 obs_gpp_best_fr2 <- t(obs_gpp_best_fr2[o,]$obs)
59 mod_gpp_best_fr2 <- bestrun_fr2$residuals[gpp_id,c(2,4)]
60 mod_gpp_best_fr2 <- t(mod_gpp_best_fr2[o,]$mod)
61 bestrun_fr2 <- list(reference=as.data.frame(obs_gpp_best_fr2), model=as.data.frame(
       mod_gpp_best_fr2))
62
63 rm(cost_best, gpp_id, o, obs_gpp_best_fr2, mod_gpp_best_fr2)
64
65 load("hfree/Standardrun_US-FR2.RData")
66 stand_fr2 <- cost_stan
```

```
67 gpp_id <- which(stand_fr2$residuals[,1] == "GPP_f")
68 o <- order(stand_fr2$residuals[gpp_id,2])
69 obs_gpp_stand_fr2 <- stand_fr2$residuals[gpp_id,c(2,3)]
70 obs_gpp_stand_fr2 <- t(obs_gpp_stand_fr2[o,]$obs)
71 mod_gpp_stand_fr2 <- stand_fr2$residuals[gpp_id,c(2,4)]
72 mod_gpp_stand_fr2 <- t(mod_gpp_stand_fr2[o,]$mod)
73 stand_fr2 <- list(reference=as.data.frame(obs_gpp_stand_fr2), model=as.data.frame(
       mod_gpp_stand_fr2))
74
75 #######   US-Wkg optimal (Bestrun..) and standard paramter (Standardrun...) runs
       #######
76 load("hfree/Bestrun_US-Wkg.RData")
77 bestrun_wkg <- cost_best
78 gpp_id <- which(bestrun_wkg$residuals[,1] == "GPP_f")
79 o <- order(bestrun_wkg$residuals[gpp_id,2])
80 obs_gpp_best_wkg <- bestrun_wkg$residuals[gpp_id,c(2,3)]
81 obs_gpp_best_wkg <- t(obs_gpp_best_wkg[o,]$obs)
82 mod_gpp_best_wkg <- bestrun_wkg$residuals[gpp_id,c(2,4)]
83 mod_gpp_best_wkg <- t(mod_gpp_best_wkg[o,]$mod)
84 bestrun_wkg <- list(reference=as.data.frame(obs_gpp_best_wkg), model=as.data.frame(
       mod_gpp_best_wkg))
85
86 load("hfree/Standardrun_US-Wkg.RData")
87 stand_wkg <- cost_stan
88 gpp_id <- which(stand_wkg$residuals[,1] == "GPP_f")
89 o <- order(stand_wkg$residuals[gpp_id,2])
90 obs_gpp_stand_wkg <- stand_wkg$residuals[gpp_id,c(2,3)]
91 obs_gpp_stand_wkg <- t(obs_gpp_stand_wkg[o,]$obs)
92 mod_gpp_stand_wkg <- stand_wkg$residuals[gpp_id,c(2,4)]
93 mod_gpp_stand_wkg <- t(mod_gpp_stand_wkg[o,]$mod)
94 stand_wkg <- list(reference=as.data.frame(obs_gpp_stand_wkg), model=as.data.frame(
       mod_gpp_stand_wkg))
95
96 ###create lists
97 standard <- list(stand_wkg, stand_fr2, stand_arc, stand_mbo)
98 bestrun <- list(bestrun_wkg, bestrun_fr2, bestrun_arc, bestrun_mbo)
99
100 #plot taylor diagram for max modis
101 #color of points
102 colors <- c(RColorBrewer::brewer.pal(5,"Set1")[2:5])
103 #point labels
```

```
104 labels <- list(c("1","2","3","4"),c("1*","2*","3*","4*"))
105 #label positions
106 pos <- c(2, 2, 3, 3, 2, 2, 3, 3)
107 fn <- paste(fout,"taylor_mcmc_improvement.jpeg",sep="")
108 jpeg(fn, height=5.5*300, width=8*300, res=300, pointsize=9)
109 taylorDiagram(standard, bestrun,
110               normalize=TRUE,
111               labels = labels,
112               show.gamma = FALSE,
113               pos.cor=FALSE,
114               ref.sd = TRUE,
115               col.pal=colors,
116               pcex = 2.5,
117               main = "Improvements in LPJmL GPP results")
118 # add text and legend
119 text(1, -0.06, labels = "reference", font = 2, cex = 0.8)
120 legend(1, 1.4, legend=c("1: US-Wkg",
121                         "2: US-FR2",
122                         "3: US-ARc",
123                         "4: IT-MBo"),
124        pch=19, col = colors, cex = 1, pt.cex = 1.2, text.font = 2)
125 dev.off()
```

code/improvement_tracking.R