Chemically Aware Model Builder (camb): An R package for property and bioactivity modelling of small molecules

Daniel S. Murrell[1,†], Isidro Cortes-Ciriano[2,†], Gerard J.P. van Westen[3], Ian P. Stott[4], Andreas Bender[1,*], Thérèse E. Malliavin[2,*], Robert C. Glen[1,*]

1 Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom.

2 Unite de Bioinformatique Structurale, Institut Pasteur and CNRS UMR 3825, Structural Biology and Chemistry Department, 25-28, rue Dr. Roux, 75 724 Paris, France.

3 European Molecular Biology Laboratory European Bioinformatics Institute Wellcome Trust Genome Campus, Hinxton, United Kingdom.

4 Unilever Research, Bebington, UK

† Equal contributors

∗ E-mail: Therese Malliavin: terez@pasteur.fr, Andreas Bender: ab454@cam.ac.uk, Robert Glen: rcg28@cam.ac.uk

Abstract

In silico predictive models have proved valuable for the optimisation of compound potency, selectivity and safety profiles. *camb* is an R package that provides an environment for the rapid generation of quantitative Structure-Property and Structure-Activity models for small molecules (QSAR, QSPR, QSAM, PCM). It is aimed at both advanced and beginner R users. Its capabilities include standardised chemical structure representation, computation of 905 two-dimensional and 14 fingerprint type descriptors for small molecules, 8 types of amino acid descriptors, 13 whole protein sequence descriptors, filter methods for feature selection, generation of predictive models (using an interface to the R package *caret*), as well as techniques to ensemble these models (using techniques from the R package *caretEnsemble*). Results can be visualised through high-quality, customisable plots (R package *ggplot2*). Overall, *camb* constitutes an open-source framework to perform the following steps: (i) compound standardisation, (ii) molecular and protein descriptor calculation, (iii) descriptor preprocessing and model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules. This will speed model generation, provide reproducibility and tests of robustness.

Introduction

The advent of high-throughtput technologies over the last two decades has led to a vast increase in the number of compound and bioactivity databases (1; 2; 3). This increase in the amount of chemical and biological information has been exploited by developing fields in drug discovery such as modern quantitative structure activity relationships (QSAR), quantitative structure property relationships (QSPR), quantitative sequence-activity modelling (QSAM), or proteochemometric modelling (PCM) (4; 5).

The R programming environment provides a flexible platform for statistical analyses (6). R is most extensively used in genomics (7), and the availability of R packages for cheminformatics and medicinal chemistry is small in comparison. Nonetheless, R currently constitutes the most frequent choice in the medicinal chemistry literature for compound bioactivity and property modelling (8). In general, these studies share a common algorithmic structure, which can be summarised in 4 model generation steps: (i) compound standardisation, (ii) descriptor calculation, (iii) preprocessing, feature selection, model training and validation, and (iv) bioactivity/property prediction for new molecules. Figure 1 illustrates the functionalities of *camb* for the common steps of a predictive bioactivity / property modelling.

Currently available R packages provide the capability for only subsets of the above mentioned steps. For instance, the R packages *chemmineR* (9) and *rcdk* (10) enable the manipulation of SDF and SMILES

files, the calculation of physicochemical descriptors, the clustering of molecules, and the retrieval of compounds from PubChem (3). On the machine learning side, the *caret* package provides a unified platform for the training of machine learning models (11).

While it is possible to use a combination of these packages to set up your own workflow, going from start to finish requires a reasonable understanding of model building in *caret*. No unified R package exists which caters to all the steps required to go from molecular structures to a trained model ready to use on unstandardised molecules in a manner that is easily followed in a tutorial containing only a few lines of code. Here, we present the R package *camb*: **C**hemically **A**ware **M**odel **B**uilder, which aims to address the current lack of an R framework encompassing the four steps mentioned above. Specifically, the *camb* package makes it extremely easy to feed new molecules that have no previous standardisation through a single function, and acquire new predictions. The package has been conceived in a way that users with minimal programming skills are able to generate competitive predictive models and high-quality plots under default operation. It must be noted that *camb* does limit practitioners to a limited but easily used workflow to begin with, however, each function can be used with non-default parameters to fulfil the more versatile needs of more experienced users. In fact, experienced users are encouraged to use the full capacity of the (11) package as these authors recommend it highly.

Overall, *camb* enables the generation of predictive models, such as Quantitative Structure-Activity Relationships (QSAR), Quantitative Structure-Property Relationships (QSPR), Quantitative Sequence-Activity Modelling (QSAM), or Proteochemometric Modelling (PCM), starting with: chemical structure files, protein sequences (if required), and the associated properties or bioactivities. Moreover, *camb* is the first R package that enables the manipulation of chemical structures utilising Indigo's C API (12), and the calculation of: (i) molecular fingerprints and 1- and 2-dimensional topological descriptors calculated using the PaDEL-Descriptor Java library (13), (ii) hashed and unhashed Morgan fingerprints (14), and (iii) 8 types of amino acid descriptors. Two case studies illustrating the application of *camb* for QSPR modelling (solubility prediction) and PCM are available in the online Supplementary Information.

Design and Implementation

This section describes the tools provided by *camb* for (i) compound standardisation, (ii) descriptor calculation, (iii) preprocessing and feature selection, model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules.

0.1 Compound stardardization

In order to represent all molecules in the same way (compound standardisation), *camb* provides the function *StandardiseMolecules* which utilises Indigo's C API (12). SDF and SMILES formats are provided as molecule input options. Any molecules that Indigo fails to parse are removed during the standardisation step. The maximum number of each halogen atom that a compound can possess in order to pass the standardisation process can be defined by the user. Additional arguments of this function include the removal of inorganic molecules or those compounds with a molecular mass above or below a defined threshold. *camb* makes use of Indigo's InChI (15) plugin to represent all tautomers in canonical SMILES by converting molecules to InChI, discarding tautomeric information, and converting back to SMILES.

0.2 Descriptor calculation

Currently, *camb* supports the calculation of compound descriptors and fingeprints via PaDEL-Descriptor (13), and Morgan circular fingerprints (14) as implemented in RDkit (16). The function *GeneratePadelDescriptors* permits the calculation of 905 1- and 2-dimensional descriptors and 10 PaDEL fingerprints, namely: CDK fingerprints, CDK extended fingerprints, Kier-Hall E-state fragments (17), CDK graph only

fingerprints, MACCS fingerprints, Pubchem fingerprints, Substructure fingerprints (18), and Klekota-Roth fingerprints (19).

Morgan fingerprints can be computed with the function *MorganFPs* through the python library RDkit (16). Hashed fingerprints can be generated as *binary*, recording the presence or absence of each substructure, or *count based*, recording the number of occurrences of each substructure. Additionally, the *MorganFPs* function also computes unhashed (keyed) fingerprints, where each substructure in the dataset is assigned a unique position in a binary fingerprint of length equal to the number of substructures existing in the dataset. Since the positions of substructures in the unhashed fingerprint depend on the dataset, the function *MorganFPs* allows calculation of unhashed fingerprints for new compounds using a basis defined by the substructures present in the training dataset. This ensures that substructures in new compounds map to the same locations on the fingerprint and allows enhanced model interpretation by noting which exact substructures are deemed important by the learning algorithm.

The function *SeqDescs* enables the calculation of 13 types of whole protein sequence descriptors from UniProt identifiers or from amino acid sequences (20), namely: Amino Acid Composition (AAC), Dipeptide Composition (DC), Tripeptide Composition (TC), Normalized Moreau-Broto Autocorrelation (MoreauBroto), Moran Autocorrelation (Moran), Geary Autocorrelation (Geary) , CTD (Composition/-Transition/Distribution) (CTD), Conjoint Traid (CTriad), Sequence Order Coupling Number (SOCN), Quasi-sequence Order Descriptors (QSO), Pseudo Amino Acid Composition (PACC), Amphiphilic Pseudo Amino Acid Composition (APAAC).

In addition, *camb* permits the calculation of 8 types of amino acid descriptors, namely: 3 and 5 Z-scales (Z3 and Z5), T-Scales (TScales), ST-Scales (STScales), Principal Components Score Vectors of Hydrophobic, Steric, and Electronic properties (VHSE), BLOSUM62 Substitution Matrix (BLOSUM), FASGAI (FASGAI), MSWHIM (MSWHIM), and ProtFP PCA8 (ProtFP8). Amino acid descriptors can be used for the modelling of the activity of small peptides or for the description of protein binding sites (5; 21; 22; 23). Further details about these descriptors and their predictive signal for predictive bioactivity modelling can be found in two recent publications (21; 24).

0.3 Model training and validation

Prior to model training, descriptors often need to be statistically preprocessed (25). To this end, several functions (see package documentation and tutorials) are provided. These functions include the removal of non-informative descriptors (function *RemoveNearZeroVarianceFeatures*) or highly correlated descriptors (function *RemoveNearZeroVarianceFeatures*), the imputation of missing descriptor values (function *ImputeFeatures*), and descriptor centering and scaling to unit variance (function *PreProcess*) among others.

camb invokes the R package *caret* to set up cross validation frameworks and train machine learning models. *caret* provides a common interface to the most popular machine learning packages that exist in R. These include learning methods in Bagging, Bayesian Methods, Boosting, Boosted Trees, Elastic Net, MARS, Gaussian Processes, K Nearest Neighbour, Principal Component Regression, Radial Basis Function Networks, Random Forests, Relevance Vector Machines, and Support Vector Machines among others. Additionally, two ensemble modelling approaches, namely greedy and stacking optimisation, have been integrated from the R package *caretEnsemble* (26), which allows the combination of models to form ensemble models, which have proven to be less error prone (23).

In greedy optimization (27), the cross-validated RMSE is optimized using a linear combination of input model predictions. The input models are all trained using an identical fold composition. Each model is assigned a weight in the following manner. Initially, all models have their weight set to zero. The weight for a given model is repeatedly incremented by 1 if the subsequent normalized weight vector results in a closer match between the weighted combination of cross-validated predictions and the observed values (i.e. lower RMSE of the linear combination). This repetition is carried out n times, by default $n = 1,000$. The resulting weight vector is then normalized to obtain a final weight vector.

In the case of model stacking (23), the predictions of the input models serve as training data points for a meta-model. This meta-model can be linear, *e.g.* Partial Least Squares (28), or non-linear, *e.g.* Random Forest (29). If the selected algorithm permits to determine the importance of each descriptor, in this case each descriptors corresponds to a single model, such as Elastic Net (30), a vector of model weights can be defined.

These model ensembles can be applied to a test set (which was not used when building the ensembles), and the error metric (e.g. RMSE) compared to that of the single models on the holdout set.

In the general case, prior to model training, the dataset is divided into a training set, comprising *e.g.* 70% of the data, and a test set, which comprises the remaining data. The test set is used to assess the predictive power of the models on new data points not considered in the training phase. In the training phase, the values of the model parameters (hyper-parameters) are optimized by grid search and *k*-fold cross validation (CV) (31). A grid of plausible hyper-parameter values covering an exponential range is defined (function *expGrid*). Next, the training set is split into *k* folds by, *e.g.* stratified or random sampling of the bioactivity / property values. For each combination of hyper-parameters, a model is trained on $k-1$ folds, and the values for the remaining fold are then predicted. This procedure is repeated *k* times, each time holding out a different fold. The values of the hyper-parameters exhibiting the lowest average RMSE (or another metric such as *e.g.* R^2) value along the *k* folds are considered optimal. A model is then trained on the whole training set using the optimal hyper-parameter values, and the predictive power of this model is assessed on the test set. The final model, trained on the whole dataset after having optimized the hyper-parameter values by CV, can be applied on an external chemical library.

Statistical metrics for model validation have also been included, namely:

Internal validation (cross-validation):

$$q^2_{int} \text{ or } R^2_{int} = 1 - \frac{\sum_{i=1}^{N_{tr}} (y_i - \widetilde{y}_i)^2}{\sum_{i=1}^{N_{tr}} (y_i - \bar{y}_{tr})^2} \tag{1}$$

$$RMSE_{int} = \frac{\sqrt{(y_i - \widetilde{y}_i)^2}}{N} \tag{2}$$

where N_{tr}, y_i, $\widetilde{y}_i$ and $\bar{y}_{tr}$ represent the size of the training set, the observed, the predicted and the averaged values of the dependent variable for those datapoints included in the training set. The *i*th position within the training set is defined by *i*.

External validation (test set):

$$Q^2_{1\ test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{tr})^2} \tag{3}$$

$$Q^2_{2\ test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2} \tag{4}$$

$$Q^2_{3\ test} = 1 - \frac{[\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}_j)^2]/N_{test}}{[\sum_{j=1}^{N_{tr}} (y_j - \bar{y}_{tr})^2]/N_{tr}} \tag{5}$$

$$RMSE_{test} = \frac{\sqrt{(y_j - \widetilde{y}_j)^2}}{N} \tag{6}$$

$$R_{test} = \frac{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})(\widetilde{y}_j - \bar{\widetilde{y}}_{test})}{\sqrt{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2 \sum (\widetilde{y}_j - \bar{\widetilde{y}}_{test})^2}} \quad (7)$$

$$R^2_{0\ test} = 1 - \frac{\sum_{j=1}^{N_{test}} (y_j - \widetilde{y}^{r0}_j)^2}{\sum_{j=1}^{N_{test}} (y_j - \bar{y}_{test})^2} \quad (8)$$

where N_{tr}, N_{test}, y_j, $\widetilde{y}_j$, and $\bar{y}_{test}$ represent the size of the training and test sets, the observed, the predicted, and the averaged values of the dependent variable for those datapoints comprising the test set, respectively. $\bar{y}_{tr}$ represents the averaged values of the dependent variable for those datapoints comprising the training set. The jth position within the training set is defined by j.

$R^2_{0\ test}$ is the square of the coefficient of determination through the origin, being $\widetilde{y}^{r0}_j = k\widetilde{y}_j$ the regression through the origin (observed versus predicted) and k its slope. The reader is referred to ref. (32) for a detailed discussion of both the evaluation of model predictive ability through the test set and about the three different formulations for Q^2_{test}, namely $Q^2_{1\ test}$, $Q^2_{2\ test}$, and $Q^2_{3\ test}$. The value of these metrics permits the assessment of model performance according to the criteria proposed by Tropsha and Golbraikh (33; 34), namely: $q^2_{int} > 0.5$, $R^2_{test} > 0.6$, $\frac{(R^2_{test} - R^2_{0\ test})}{R^2_{test}} < 0.1$, and $0.85 \leq k \leq 1.15$. Although these values might change depending on the dataset modelled, as well as on the application context, *e.g.* higher errors might be tolrated in hit identification in comparison to lead optimization, these criteria can serve as general guidelines to assess model predictive ability. The function *Validation* permits the calculation of all these metrics.

In cases where information about the experimental error of the data is available, the values for the statistical metrics on the test set can be compared to the theoretical maximum and minimum achievable values given (i) the uncertainty of the experimental measurements, (ii) the size of the training and test sets, and (iii) the distribution of the dependent variable (35). The distribution of maximum and minimum $R^2_{0\ test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values can be computed with the functions *MaxPerf* and *MinPerf*. The distributions of maximum model performance are calculated in the following way. A sample, S, of size equal to the test set is randomly drawn from the dependent variable, *e.g.* IC_{50} values. Next, the experimental uncertainty is added to S, which defines the sample S_{noise}. The $R^2_{0\ test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values for S against S_{noise} are then calculated. These steps are repeated n times, by default 1,000, to calculate the distributions of $R^2_{0\ test}$, R_{test}, Q^2_{test}, and $\text{RMSE}_{\text{test}}$ values. To calculate the distributions of minimum model performance, the same steps are followed, with the exception that S is randomly permuted before calculating the values for the statistical metrics.

Visualization functionality for model performance and for exploratory analyses of the data is provided. All plots are generated using the R package *ggplot2* (36). Default options for plotting functions allow the generation of high-quality plots, and in addition, the layer-based structure of ggplot objects allows for further optimisation by the addition of customisation layers. These visualization tools include correlation plots (*CorrelationPlot*), bar plots with error bars (*ErrorBarplot*), and Principal Component Analysis (PCA) (*PCA* and *PCAPlot*) among others. For instance, the *camb* function *PCA* performs a Principal Component Analysis (PCA) on compound and/or protein descriptors. The output can be directly sent to the function *PCAPlot*, which will depict the two fist principal components, with the shape and color of a user-defined factor *e.g.* compound class or protein isoform (Figure 2).

Visual depiction of compounds is also possible with the function *PlotMolecules*, utilising Indigo's C API. Visualization functions are exemplified in the tutorials provided in the Supplementary Information and with the package documentation.

0.4 Predictions for new molecules

One of the major benefits of having all the tools available in one framework is that it is straightforward to perform exactly the same processing on new molecules as that used on the training set, *e.g.* standardisation of molecules and centering and scaling of descriptors. The *camb* function *PredictExternal* allows the user to read an external set of molecules together with a trained model, and outputs predictions on this external set. This *camb* functionality ensures that the same standardization options and descriptor types are used when a model is applied to make predictions for new molecules. An example of this is shown in the QSPR tutorial.

Results

Two tutorials demonstrating property and bioactivity modelling are available in the Supplementary Information and with the package documentation. We encourage *camb* users to visit the package repository (https://github.com/cambDI/camb) for future updated versions of the tutorials. In the following subsections, we show the results obtained for the two case studies presented in the tutorials, namely: (i) QSPR: prediction of compound aqueous solubility (logS), and (ii) PCM: modelling of the inhibition of 11 mammalian cyclooxygenases (COX) by small molecules. The datasets are available in the *examples/PCM* directory of the package. Further details about the PCM dataset be found in ref. (23).

0.5 Case Study 1: QSPR

To illustrate the functionalities of *camb* for compound property modelling, we downloaded the aqueous solubility values for 1,708 small molecules. Aqueous solubility values were expressed as logS, where S corresponds to the solubility at a temperature of 20-25 °C in mol/L. We subjected compound structures to a common representation using the function *StandardiseMolecules* with the default parameters. Thus, all molecules were kept irrespective of the numbers of fluorines, iodines, chlorine, and bromines present in their strucuture, or of their molecular mass. Molecules were represented with implicit hydrogens, dearomatized, and passed through the InChI format to ensure that tautomers are represented by the same SMILES. Next, we calculated 905 one and two-dimensional topological and physicochemical descriptors with the the function *GeneratePadelDescriptors* provided by the PaDEL-Descriptor (13) Java library built into the camb package. We imputed missing descriptor values with the function *ImputeFeatures*, and removed (i) highly-correlated descriptors (function *RemoveHighlyCorrelatedFeatures* using a cut-off value of 0.95), which provide redundant predictive signal, and descriptors with a variance close to zero (the function *RemoveNearZeroVarianceFeatures* using a cut-off value of 30/1), which do not contain any predictive signal. Prior to model training, all descriptors were centered to zero mean and scaled to unit variance with the function *PreProcess*. After applying these steps the dataset consisted of 1,606 molecules encoded with 211 descriptors.

We trained three machine learning models on 80% of the data (training set), namely: (i) Support Vector Machine (SVM) with a squared exponential kernel, (ii) Random Forest (RF), and (iii) Gradient Boosting Machines (GBM). 5-fold cross-validation was used to optimize the value of the hyperparameters. The values for the internal and external validation for these three models are summarized in Table 1. Overall, the three algorithms displayed high performace on the test set, with RMSE / R_0^2 values of: GBM: 0.52/0.93; RF: 0.59/0.91; and SVM: 0.60/0.91 (Table 1 and Figure 3A). We next evaluated whether the combination of these three models into model ensembles leads to improved predictive ability. We explored the two ensemble modelling techniques supported by *camb*, namely: greedy optimization and model stacking. First, we trained a greedy ensemble using 1,000 iterations with the function *caretEnsemble*. The greedy ensemble picks a linear combination of model outputs that is a local minimum in the RMSE landscape.

Secondly, we created a linear and a non-linear stacking ensembles. In model stacking, the cross-validated predictions of a library of models are used as descriptors, on which a meta-model (ensemble model) is trained. This meta model can be a linear model, *e.g.* SVM with a linear kernel, or non linear, such as Random Forest. The application of ensemble modelling led to a decrease by 10-15% of $RMSE_{test}$ values (Table 1). The highest predictive power was obtained with the greedy and the Linear Stacking ensembles, with $R^2_{0\ test}$ and $RMSE_{test}$ of 0.93 and 0.51, respectively. Taken together, these results indicate that higher predictive power can be obtained by combining different single QSPR models with both greedy optimisation and model stacking.

0.6 Case Study 2: Proteochemometrics

In the second case study we illustrate the functionalities of *camb* for Proteochemoemtric Modelling. The tutorial "PCM with *camb*" reports the complete modelling pipeline for this dataset (23). We extracted the bioactivity data for 11 mammalian COX from ChEMBL 16 (2; 23) (Table 2). We only kept the data satisfying the following criteria: (i) assay score confidence higher than 8, (ii) activity relationship equal to '=', (iii) activity type equal to "IC50", and (iv) activity unit equal to 'nM'. The mean IC_{50} value was taken for duplicated compound-COX combinations. The final dataset comprised 3,228 distinct compounds, 11 mammalian COX, and a total number of 4,937 datapoints (13.9% matrix completeness) (23).

All compound structures were subjected to a common representation with the function *StandardiseMolecules* using the default parameters. Next, we calculated (i) PaDEL descriptors (13) with the function *GeneratePadelDescriptors*, (ii) and Morgan fingerprints with the function *MorganFPs*. We considered substructures with a maximal diameter of 4 bonds, whereas the length of the fingerprints was set to 512. To describe the target space, we derived the binding site amino acid descriptors from the crystallographic structure of ovine COX-1 complexed with celecoxib (PDB ID: 3KK6 (37)) by selecting those residues within a sphere of radius equal to 10 Å centered in the ligand. Subsequently, we performed multiple sequence alignment to determine the corresponding residues for the other 10 COX, and calculated 5 *Z*-scales for these residues with the function *AADescs*.

Prior to model training, missing descriptor values were imputed (function *ImputeFeatures*). Subsequently, we removed highly-correlated descriptors (function *RemoveHighlyCorrelatedFeatures* using a cut-off value of 0.95), and those exhibiting a variance close to zero (function *RemoveNearZeroVarianceFeatures* using a cut-off value of 30/1). All descriptors were then centered to zero mean and scaled to unit variance (*z*-scores) (function *PreProcess*). These steps led to a final selection of 356 descriptors: 242 bits from the Morgan fingerprints, 99 physicochemical descriptors, and 15 *Z*-scales. The dataset was split into a training set, comprising 80% of the data, and a test set with the function *SplitSet*. We trained three single PCM models, namely: GBM, RF, and SVM with a squared exponential kernel (Table 3), with 5-fold cross-validation.

Next, we combined these models into model ensembles using (i) greedy optimisation (1,000 iterations), and (ii) model stacking (Table 3). The function *Validation* served to calculate the values for the statistical metrics on the test set. The observed against the predicted values on the test set were reported with the function *CorrelationPlot* (Figure 3B). All model ensembles displayed higher predictive power on the test set than single PCM models (Table 3). The lowest RMSE value on the test set, namely 0.72 was obtained with the Elastic Network (EN) Stacking model (Table 3), whereas the higest R^2_0 value, namely 0.63, was obatined with the greedy, the Linear Stacking and the SVM Radial Stacking ensembles. As in the previous case study, these data indicate that higher predictive power can be obtained by combining single PCM models in more predictive model ensembles, although this improvement might be sometimes marginal. Overall, this case study illustrates the versatility of *camb* to train and validate PCM models from amino acid sequences and compound structures in an integrated and seamless modelling pipeline.

Availability and Future Directions

camb is coded in R, C++, Python and Java and is available open source at https://github.com/cambDI/camb. Pre-compiled versions are available for OSX, Linux and Windows. We plan to include further functionality based on the C++ Indigo API, and to implement new error estimation methods for regression and classification models. Additionally, we plan to further integrate the python library RDkit with *camb*. The package is fully documented and includes the usage examples and details of the R functions implemented in *camb*.

In silico predictive models have proved valuable for the optimisation of compound potency, selectivity and safety profiles. In this context, *camb* provides an open framework to (i) compound standardisation, (ii) molecular and protein descriptor calculation, (iii) preprocessing and feature selection, model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules. This will speed model generation, provide reproducibility and tests of robustness. *camb* functions have been designed to meet the needs of both expert and amateur users. Therefore, *camb* can serve as an education platform for undergraduate, graduate, and post-doctoral students, while providing versatile functionalities for predictive bioactivity / property modelling in more advanced settings.

Author Contributions

Conceived and coded the package: ICC and DM. Wrote the tutorials: ICC and DM. Provided analytical tools for amino acid descriptor calculation: GvW. Wrote the paper: DM, ICC, GvW, IS, AB, TM and RG.

Acknowledgements

ICC thanks the Paris-Pasteur International PhD Programme and Institut Pasteur for funding. TM thanks CNRS and Institut Pasteur for funding. DSM and RCG thanks Unilever for funding. GvW thanks EMBL (EIPOD) and Marie Curie (COFUND) for funding. AB thanks Unilever and the European Research Commission (Starting Grant ERC-2013-StG 336159 MIXTURE) for funding.

References

1. Bender A (2010) Databases: Compound bioactivities go public. Nat Chem Biol 6: 309–309.

2. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, et al. (2011) ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic Acids Res 40: D1100–D1107.

3. Wang Y, Xiao J, Suzek TO, Zhang J, Wang J, et al. (2012) PubChem's BioAssay database. Nucleic Acids Res 40: D400–412.

4. van Westen GJP, Wegner JK, IJzerman AP, van Vlijmen HWT, Bender A (2011) Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. Med Chem Comm 2: 16-30.

5. Cortes Ciriano I, Ain QU, Subramanian V, Lenselink EB, Mendez Lucio O, et al. (2014) Polypharmacology modelling using proteochemometrics: Recent developments and future prospects. Med Chem Comm Accepted .

6. R Core Team (2013) R: A language and environment for statistical computing .

7. Gentleman RC, Carey VJ, Bates DM, others (2004) Bioconductor: Open software development for computational biology and bioinformatics. Genome Biol 5: R80.

8. Mente S, Kuhn M (2012) The use of the R language for medicinal chemistry applications. Curr Top Med Chem 12: 1957–1964.

9. Cao Y, Charisi A, Cheng LC, Jiang T, Girke T (2008) ChemmineR: a compound mining framework for R. Bioinformatics 24: 1733-1734.

10. Guha R (2007) Chemical informatics functionality in R. J Stat Softw 18.

11. Kuhn M (2008) Building predictive models in r using the caret package. J Stat Softw 28: 1–26.

12. Indigo (2013) Indigo cheminformatics library .

13. Yap CW (2011) PaDEL-Descriptor: an open source software to calculate molecular descriptors and fingerprints (v2.16). J Comput Chem 32: 1466–1474.

14. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. J Chem Inf Model 50: 742–754.

15. InChI (2013) IUPAC - international union of pure and applied chemistry: The IUPAC international chemical identifier (InChI) .

16. Landrum G (2006). Rdkit: Open-source cheminformatics. URL `http://www.rdkit.org`.

17. Hall LH, Kier LB (1995) Electrotopological state indices for atom types: A novel combination of electronic, topological, and valence state information. J Chem Inf Comput Sci 35: 1039–1045.

18. O'Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, et al. (2011) Open Babel: An open chemical toolbox. J Cheminf 3: 33.

19. Klekota J, Roth FP (2008) Chemical substructures that enrich for biological activity. Bioinformatics 24: 2518–2525.

20. Xiao N, Xu Q (2014) protr: Protein sequence descriptor calculation and similarity computation with R .

21. van Westen GJ, Swier RF, Cortes-Ciriano I, Wegner JK, Overington JP, et al. (2013) Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): Modeling performance of 13 amino acid descriptor sets. J Cheminf 5: 42.

22. van Westen GJP, van den Hoven OO, van der Pijl R, Mulder-Krieger T, de Vries H, et al. (2012) Identifying novel adenosine receptor ligands by simultaneous proteochemometric modeling of rat and human bioactivity data. J Med Chem 55: 7010–7020.

23. Cortes-Ciriano I, Murrell DS, van Westen GJ, Bender A, Malliavin T (2014) Ensemble modeling of cyclooxygenase inhibitors. Manuscript in Preparation .

24. van Westen G, Swier R, Wegner JK, IJzerman AP, van Vlijmen HW, et al. (2013) Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets. J Cheminf 5: 41.

25. Andersson CR, Gustafsson MG, Strmbergsson H (2011) Quantitative chemogenomics: machine-learning models of protein-ligand interaction. Curr Top Med Chem 11: 1978–1993.

26. Mayer Z (2013) caretEnsemble: Framework for combining caret models into ensembles. [R package version 1.0] .

27. Caruana R, Niculescu-Mizil A, Crew G, Ksikes A (2004) Ensemble selection from libraries of models. In: Proceedings of the Twenty-first International Conference on Machine Learning. New York, NY, USA: ACM, ICML '04, p. 18.

28. Wold S, Sjstrm M, Eriksson L (2001) PLS-regression: a basic tool of chemometrics. Chemometr Intell Lab 58: 109-130.

29. Breiman L (2001) Random forests. Machine Learning 45: 5–32.

30. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. iJ R Stat Soc 67: 301–320.

31. Hawkins DM, Basak SC, Mills D (2003) Assessing Model Fit by Cross-Validation. J Chem Inf Comput Sci 43: 579–586.

32. Consonni V, Ballabio D, Todeschini R (2010) Evaluation of model predictive ability by external validation techniques. J Chemometrics 24: 194-201.

33. Golbraikh A, Tropsha A (2002) Beware of q2! J Mol Graphics Modell 20: 269–276.

34. Tropsha A, Gramatica P, Gombar VK (2003) The Importance of Being Earnest: Validation is the Absolute Essential for Successful Application and Interpretation of QSPR Models. QSAR Comb Sci 22: 69-77.

35. Cortes Ciriano I, van Westen GJ, Lenselink EB, Murrell DS, Bender A, et al. (2014) Proteochemometric modeling in a Bayesian framework. J Cheminf 6: 35.

36. Wickham H (2009) ggplot2: elegant graphics for data analysis .

37. Rimon G, Sidhu RS, Lauver DA, Lee JY, Sharma NP, et al. (2010) Coxibs interfere with the action of aspirin by binding tightly to one monomer of cyclooxygenase-1. Proc Natl Acad Sci USA 107: 28-33.

38. Kruger FA, Overington JP (2012) Global Analysis of Small Molecule Binding to Related Protein Targets. PLoS Comput Biol 8: e1002333.

Figure Legends

Figure 1: Overwiew of *camb* functionalities for bioactivity / property modelling (QSAR, QSPR, QSAM and PCM). *camb* provides an open and seamless framework to (i) compound standardisation, (ii) molecular and protein descriptor calculation, (iii) preprocessing and feature selection, model training, visualisation and validation, and (iv) bioactivity/property prediction for new molecules. In the first instance, compound structures are subjected to a common representation with the function *StandardiseMolecules*. Proteins are encoded with 8 types of amino acid and / or 13 types of full protein sequence descriptors, whereas *camb* enables the calculation of 905 2D physicochemical descriptors for small molecules, and 14 types of fingerprints, such as Morgan or Klekota fingerprints. Molecular descriptors are statistically preprocessed, *e.g.* by centering their values to zero mean and scaling them to unit variance. Subsequently, single or ensemble machine learning models can be trained, visualized and validated. Finally, the *camb* function *PredictExternal* allows the user (i) to read an external set of molecules with a trained model, (ii) to apply the same processing to these new molecules, and (iii) to output predictions for this external set. This ensures that the same standardization options and descriptor types are used when a model is applied to make predictions for new molecules.

Figure 2: PCA analysis of the binding site amino acid descriptors corresponding to the 11 mammalian cyclooxygenases considered in the second case study (Proteochemometrics). Binding site amino acid descriptors (5 Z-scales) were input to the function *PCA*. The first two principal components (PCs) explained more than 80% of the variance. This indicates that there are mainly two sources of variability in the data. To generate the plot, we used the function *PCAPlot* using the default options. Cyclooxigenases cluster into two distant groups, which correpond to the isoenzyme type, *i.e. COX-1 and COX-2*. Given that small molecules tend to display similar binding profiles within orthologues (38), we hypothesized that merging bioactivity data from paralogues and orthologues will lead to more predictive PCM models (23).

Figure 3: Observed against predicted values on the test set corresponding to (A) the compound solubility (LogS) dataset (case study 1: QSPR), and (B) the cyclooxygenase (COX) inhibition dataset (case study 2: PCM). Both A and B were generated with the function *CorrelationPlot*. The area defined by the blue lines comprises 1 LogS units (A) and 1 pIC_{50} units (B). Both plots were generated using the predictions on the test set calculated with the Linear Stacking ensembles (Table 1 and 3). Overall, high predictive power is attained on the test set for both datasets, with respective RMSE/R_0^2 values of 0.51/0.93 (A), and 0.73/0.63 (B). Taken together, these data indicate that ensemble modelling leads to higher predictive power, although this increase might be marginal for some datasets (B).

Tables

	Algorithm	R^2_{int}	RMSE_{int}	$R^2_{0\ test}$	$\text{RMSE}_{\text{test}}$
A	GBM	0.90	0.59	0.93	0.52
	RF	0.89	0.62	0.91	0.59
	SVM Radial	0.88	0.63	0.91	0.60
B	Greedy	-	0.57	0.93	0.51
	Linear Stacking	0.90	0.57	0.93	0.51
	RF Stacking	0.89	0.62	0.92	0.55

Table 1: Internal and external validation metrics for the single and ensemble QSPR models trained on the compound solubility dataset. The lowest RMSE value on the test set, namely 0.51, was obtained with the greedy and with the linear stacking ensembles.
Abbreviations. GBM: Gradient Boosting Machine; RF: Random Forest; RMSE: root mean square error in prediction; SVM: Support Vector Machines.

UniProt ID	Isoenzyme	Organism	Number of Datapoints
P23219	1	*Homo sapiens*	1,346
O62664	1	*Box taurus*	48
P22437	1	*Mus musculus*	50
O97554	1	*Oryctolagus cuniculus*	11
P05979	1	*Ovis aries*	442
Q63921	1	*Rattus Norvegicus*	23
P35354	2	*Homo sapiens*	2,311
O62698	2	*Bos taurus*	21
Q05769	2	*Mus musculus*	305
P79208	2	*Ovis aries*	341
P35355	2	*Rattus Norvegicus*	39

Table 2: Cyclooxygenase inhibition dataset (case study 2: PCM). We extracted the bioactivity data for 11 mammalian cyclooxigenases from ChEMBL 16 (**?**). The final bioactivity selection comprised 3,228 distinct compounds.

	Algorithm	R^2_{int}	$RMSE_{int}$	$R^2_{0\ test}$	$RMSE_{test}$
A	GBM	0.59	0.77	0.60	0.76
	RF	0.60	0.78	0.61	0.79
	SVM	0.61	0.75	0.60	0.76
B	Greedy Ensemble	-	0.73	0.63	0.73
	Linear Stacking	0.63	0.73	0.63	0.73
	EN Stacking	0.63	0.72	0.62	0.72
	SVM Linear Stacking	0.63	0.73	0.62	0.73
	SVM Radial Stacking	0.63	0.73	0.63	0.73
	RF Stacking	0.61	0.76	0.58	0.77

Table 3: Internal and external validation metrics for the single and ensemble PCM models trained on the COX dataset. Combining single models trained with different algorithms in model ensembles allows to increase model predictive ability. We obtained the highest $R^2_{0\ test}$ and $RMSE_{test}$ values namely, 0.63 and 0.73 pIC_{50} unit respectively, with the greedy ensemble, and with the following model stacking techniques: (i) linear, and (ii) SVM radial.
Abbreviations. EN: Elastic Net; GBM: Gradient Boosting Machine; RF: Random Forest; RMSE: root mean square error in prediction; SVM: Support Vector Machines.