Machine learning with R in the Life Sciences

17.03. + 18.03.2025
Konstantin Pelz, Prof. Dr. Markus List
Data Science in Systems Biology

This beginner's course will introduce you to the fundamentals of machine learning using R. You will be guided through the entire process, from data selection to model evaluation and improvement. The course emphasizes best practices and general knowledge, and requires a basic understanding of R programming. To simplify the learning experience, we will use the tidyverse and tidymodels packages, more on that later.

By the end of this course, you will be able to:

- Understand the basic concepts of machine learning.
- Select and split data meaningfully.
- Choose appropriate machine learning methods for various problems.
- Implement basic machine learning methods in R.

This two-day course is divided into:

- Day 1: Theoretical input covering fundamental concepts.
- Day 2: Practical sessions in smaller groups to apply the learned concepts.

Content

Data

Data is the foundation of machine learning (garbage in, garbage out). The challenging aspects are selecting the right data and splitting it intelligently. We will mostly be working with tabular data that is either numeric or categorical.

Selection

The quality of your model depends on the quality of the data used to train it. Therefore, selecting sufficient, high-quality data is crucial. The data should have the following properties:

- **Tidy**: Well-structured.
- **Balanced**: All classes appear approximately equally.
- **Normalized**: Variables have the same mean and standard deviation.
- **Confounder Corrected**: No unwanted confounders, such as measurement batch effects.
- **Complete**: Some models can handle missing data, while others cannot.
- **Correct data type**: Some models require numeric data, so you might need a numeric encoding of categorical variables.

If your data has issues, consider the following solutions:

- **Imbalanced Data**: Use downsampling/subsampling or upsampling/supersampling. One example would be to create synthetic samples using the upsampling method SMOTE[1].
- **Missing Data**: Perform imputation to make your data more complete, though this may introduce some noise.

The amount of data needed depends on the model used. Simpler models like logistic regression may be stable with 20-50 events per variable, while more complex models like random forests, support vector machines, or neural networks may require over 200 events per variable[2]. Generally, choose your model based on the available data and compare different models.

[1] *SMOTE: Synthetic Minority Over-sampling Technique* by Chawla et. al. at https://doi.org/10.48550/arXiv.1106.1813

[2] *Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints* from van der Ploeg et. al. at https://doi.org/10.1186/1471-2288-14-137

Splitting

Data is provided to the model in multiple steps, typically split into Training, Validation, and Test sets:

- **Training:** Data shown to the model to learn underlying patterns.
- **Validation:** An intermediate test set used to optimize hyperparameters.
- **Test:** The final set used to evaluate the performance of the trained and optimized model. It should only be used once, and no parameters should be changed after evaluation.

Ensure that the sets do not overlap or have similarities to avoid **data leakage**, which can lead to artificially improved performance that does not translate to real-world examples.

To improve robustness, use **k-fold cross-validation**. This method systematically splits the data into k mini-datasets, each used as a test set once while the rest is used for training. Other types of cross-validation include:

- **Repeated Cross-Validation:** To mitigate noise in the results of a single cross-validation, it can be repeated and averaged.
- **Nested Cross-Validation:** Not only the testing, but also the hyperparameter tuning is cross-validated.
- **Leave-One-Out Cross-Validation:** Used for small datasets. Every data point is used as the test set once, while all other data points are used for training.
- **Bootstrapping**: In bootstrapping, mini-datasets get created by sampling data from our original dataset by random sampling with replacement. This way, the distribution is different, and not all data might be used, which shows how robust the model is.

Simple k-fold cross-validation, image https://www.baeldung.com/cs/cross-validation-k-fold-loo

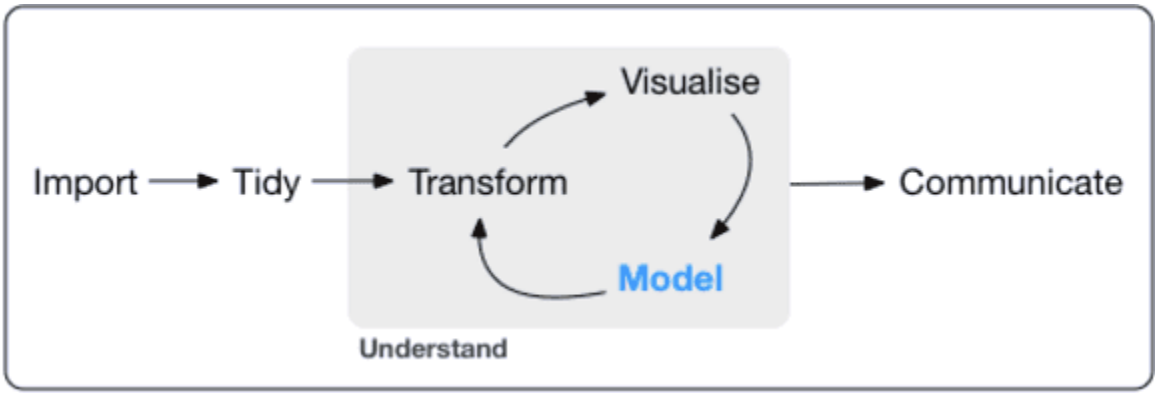
Software

This course is not an introduction to R programming. For those interested in learning the basics of R, please refer to the graduate school resources. Below are some links and background information for further self-study.

We will be working with the tidyverse. But what is the tidyverse?

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

These curated packages are designed to work seamlessly together, with consistent function signatures. For example, the first formal argument in tidyverse functions is always a data frame that serves as the function's input. The main philosophy involves the following workflow: importing data, tidying it, and then understanding it through visualization and modification, ultimately leading to effective communication of results.



https://www.kdnuggets.com/2017/10/tidyverse-powerful-r-toolbox.html

For more information, you can explore the following resources:

- https://www.tidyverse.org/learn/
- https://www.tidymodels.org/learn/
- https://gagneurlab.github.io/dataviz/tidy-data-and-combining-tables.html
- https://moderndive.com/index.html
- https://rstudio.github.io/cheatsheets/data-transformation.pdf
- https://www.tmwr.org/

The main packages we will be working with will be ggplot2, dplyr, tibble, and tidy models.

Models

Unsupervised

Unsupervised models are excellent for discovering trends and hidden patterns in data without any labels or categories. They are often used for clustering problems. Unsupervised models do not use any prior knowledge, i.e. they work with unlabeled data.

Exemplary Functions:

- Visualize complex data to provide an overview.
- Identify structures in data.
- Identify groups of samples with high similarity or short distance.
- Identify similar features (e.g., co-expressed genes).

Commonly used methods:

- **Principal Component Analysis (PCA):** Linearly transforms data so the principal components capture the largest variation.
- **Hierarchical Clustering:** Clusters data points based on distance, creating a tree that can be cut at any height to form clusters.
- **Partitioning (K-means Clustering):** Iteratively creates groups based on local similarity.
- **Self-Organizing Maps:** Neural model that preserves topological structure.

Semi-supervised

Semi-supervised learning combines labeled and unlabeled data, useful when unlabeled data is easy to obtain but labeled data is scarce or expensive.

Exemplary functions:

- **Self-Training:** Predict labels for unlabeled data, then retrain on the combination of predictions and original data.
- **Co-Training:** Ensemble method using multiple models on subsets of features or data, labeling with the highest agreement.

Commonly used methods:

- **Label Propagation:** Creates labels for unlabeled data based on the closest labeled data points.
- **Gaussian Mixture Models:** Soft clustering technique to determine the probability that a data point belongs to a cluster.

For more information, visit IBM's Semi-Supervised Learning page:
https://www.ibm.com/think/topics/semi-supervised-learning

Supervised

Supervised learning models use a labeled training set to estimate labels for new data or relationships in the data, often used for classification or regression problems.

Exemplary functions:

- **Classifying Samples:** Assign labels to unseen data.
- **Regression:** Identify trends in data and predict additional data points.

Commonly used methods:

- **Regression (Linear, Logistic, Regularized):** Estimates relationships between a dependent variable (label) and one or more independent variables (attributes/features).
- **Decision Tree-Based Methods:** Includes decision trees and random forests.
- **Nearest-Neighbor Methods:** Uses distance to find points that have a label and are close by.
- **Support Vector Machines (SVMs):** Finds a hyperplane that splits your data.
- **Neural Networks:** Inspired by human biology, using a network of weights and biases to propagate an input signal.
- **Language Models:** Often predict the most likely next word.

Model selection

Selecting the right model for your use-case is crucial, as there is often a trade-off between complexity and performance. The appropriate model helps you optimize the properties that are important to your problem. In general, it makes sense to start with a simple model to get a baseline.

Important Properties to Consider:

- **Accuracy**: How accurately the model predicts the data.
- **Interpretability**: How well a human can follow the model's decision-making process.
- **Robustness**: How well the model deals with missing or noisy data.
- **Data Requirements**: Whether the model can handle your data modalities, such as images or text.
- **Resource Requirements / Scalability**: The computational power needed for the model, both for the current amount of data and scaled up for the future.

Model Types and Their Pros and Cons[3]:

- **Regression Models (e.g., Linear, Logistic Regression):**
  - Pros: High interpretability, low resource usage.
  - Cons: May not handle complex relationships well.
- **Decision Trees, Random Forest:**
  - Pros: Good interpretability, good scalability, lower resource usage.
  - Cons: Can be prone to overfitting (especially decision trees).

[3] *https://robots.net/fintech/how-to-decide-which-model-to-use-in-machine-learning/* accessed on 04.03.2025

- **Neural Networks:**
  - Pros: Great accuracy.
  - Cons: Low interpretability, high resource usage, needs lots of data.
- **Support Vector Machines (SVMs):**
  - Pros: Good scalability.
  - Cons: Can be resource-intensive and less interpretable.
- **Models with Regularization (e.g., Lasso, Ridge):**
  - Pros: Robust to overfitting.
  - Cons: May require careful tuning of regularization parameters.

Result evaluation

Evaluating the performance of your models is essential when tuning parameters or reporting results.

		Predicted condition		Sources: [8][9][10][11][12][13][14][15] view · talk · edit	
	Total population $= P + N$	**Predicted positive**	**Predicted negative**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)** [a]	**True positive** (TP), hit[b]	**False negative** (FN), miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate type II error [c] $= \frac{FN}{P} = 1 - TPR$
	Negative (N)[d]	**False positive** (FP), false alarm, overestimation	**True negative** (TN), correct rejection[e]	False positive rate (FPR), probability of false alarm, fall-out type I error [f] $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{TP + FP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{TN + FN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{TP + FP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{TN + FN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2\,PPV \times TPR}{PPV + TPR} = \frac{2\,TP}{2\,TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

a. ^ the number of real positive cases in the data
b. ^ A test result that correctly indicates the presence of a condition or characteristic
c. ^ Type II error: A test result which wrongly indicates that a particular condition or attribute is absent
d. ^ the number of real negative cases in the data
e. ^ A test result that correctly indicates the absence of a condition or characteristic
f. ^ Type I error: A test result which wrongly indicates that a particular condition or attribute is present

https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Basic_concept

Basic terms:

- True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)Specificity / True Negative Rate (TNR): Fraction of the negative classifications among all **truly negative** = TN / TN + FP
- Recall / Sensitivity / True Positive Rate (TPR): Fraction of the true positive classifications among all **truly positive**
  = TP / TP + FN
- Precision: Fraction of the true positive classifications among all **classified as positive** = TP / TP + FP
- F1 score: widely used as it scores the balance between precision and recall
  = 2 * Precision * Recall / (Precision + Recall) = 2 * TP / (2 * TP + FP + FN)
- Type I Error: False positive (classified as positive but truly negative)
- Type II Error: False negative (classified as negative but truly positive)
- Accuracy and Balanced Accuracy: Fraction of correctly classified data points. The balanced accuracy takes the performance of both classes into account and is calculated with the formula $\frac{sensitivity+specificity}{2}$.
- Correlation: Statistical relationship between two variables. Important: Correlation is not causation! One example of a coefficient would be the concordance correlation coefficient.
- Mean Squared Error (MSE): $\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \dot{Y}_i\right)^2$, where n is the number of observations, Y_i is the true value of the i^{th} observation, and $\dot{Y}_i$ is the predicted value of the i^{th} observation.
- Receiver operating characteristic (ROC): Visual representation of TPR and FPR at different thresholds, usually a curve
- Precision-Recall curve: Visual representation of the precision and recall at different cutoffs.
- Area under the (receiver operating characteristic) curve (AUC / AUROC): Number representing the area under the curve, where 1.0 (a perfect square) is a perfect model and 0.5 is random.
- AUPRC: Same as AUROC but under the precision-recall curve.

There are some things to consider when performing an evaluation:

- **Unbalanced dataset**: When your dataset is very unbalanced, containing mostly observations of one class, your model might perform well by just predicting the majority class. This is not transferable to real-world data.

- **Batch effects**: Biological data is often measured in smaller batches, which can cause technical variations between them, known as batch effects. These artifacts can skew results and confound the signal of the underlying data.
- **Data leakage**: To get true performance metrics, ensure your model has never seen the test data before. This also applies to data very similar to your training data or containing properties that allow shortcuts in prediction. This paper[4] about data leakage provides more insight by formulating guiding questions. Your data might also contain additional information that gives the model a shortcut for prediction. For example, if cancer patients are measured in the morning and non-cancer patients in the afternoon, the model might predict based on measurement time rather than the actual data.
- **Project Management Triangle**: This concept refers to the trade-off between time, money, and quality. In machine learning, there is often a trade-off between computing resources and the quality of the result. This paper[5] for example, built their own version of AlphaFold 3 that was a little worse, but much faster and thus could be used for an initial assessment of a big dataset.
- **Robust results**: To ensure stable and robust results, use methods like cross-validation or bootstrapping.
- **Other pitfalls**: This paper[6] gives insight into additional pitfalls with the applied context of genomics. Some of them were already mentioned in this script.

When evaluating the performance of a clustering model, different methods of evaluation are required:

- Silhouette score: This score gets calculated for each point and describes how well a point fits in its own cluster compared to the others. The formula is $s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$, where s_i is the silhouette score of point i, a_i refers to the average distance between point i and all other points in its cluster, whereas b_i describes the lowest average distance between point i and all points in any other cluster (so the lowest distance to any other cluster). This score ranges from –1 to 1, where 1 describes a well clustered point and –1 refers to a point that is likely misclassified.

[4] *Guiding questions to avoid data leakage in biological machine learning applications* by Bernett et. al. at https://doi.org/10.1038/s41592-024-02362-y
[5] *Computing the Human Interactome* by Zhang et. al. at https://doi.org/10.1101/2024.10.01.615885
[6] *Navigating the pitfalls of applying machine learning in genomics* by Whalen et. al. at https://doi.org/10.1038/s41576-021-00434-9

- Rand index: This index compares the similarity of two different clusterings by looking at all points in a pairwise fashion. The formula is $R = \frac{a+b}{n}$, where a refers to number of times any pair of two points belongs to the same cluster in both clusterings, b describes the number of times any pair of two points belongs to two different clusters in both clusterings, and n is the total number of pairs.
- Adjusted rand index: The rand index has a bias towards clusterings with a big number of clusters, so Yeung and Ruzzo proposed an adjusted version in 2001. This takes this into account and adjusts by the expected index.

Model improvement

The first model is usually not the final model. There are various strategies to enhance the performance of your model.

General Considerations

The first question to answer is, "What does improvement mean?" While this may seem simple, it is crucial to clarify your priorities and optimization goals. For example, if you are creating a screening test, you want high sensitivity to avoid missing any cases. Conversely, for a diagnostic test, you want high specificity to avoid false diagnoses. Additionally, over-optimizing performance metrics can lead to overfitting, where the model is too tailored to your data and does not generalize well to new data points.

Data-Focused Strategies

Improving your data can significantly enhance model performance. The two main strategies are improving data quality and increasing data quantity.

- **Data Quality:** Improve data quality by using other methods or machine learning models. For instance, if you are predicting based on a diagnosis hidden in a lengthy description, you could use one model to extract diagnoses from the descriptions and another model for the prediction. This way, you don't need one model to do both tasks.
- **Data Quantity:** Increase your data by sourcing from more places, such as web scraping. This may result in "dirty data," which can be inaccurate, incomplete, inconsistent, or erroneous. However, increasing your dataset size with dirty data can sometimes improve performance. For sensitive data like health data, additional data points may be inaccessible due to data protection. Using federated learning,

where training occurs locally and only model weights are shared, can lead to more usable data.

Model-Focused Strategies

The standard way to improve a model is by tuning its hyperparameters. This can be done manually or through systematic searches like grid search, where all parameter combinations get tried and evaluated. It is important to use a validation set, not the final test set, for performance evaluation.

Additionally, ensemble methods combine multiple models to improve performance. These methods rely on the concept that multiple weak models can form a strong learner[7]. For example:

- **Bagging:** Trains multiple models independently and averages their predictions.
- **Boosting:** Trains multiple models sequentially, with each model correcting the errors of the previous one.

[7] *The strength of weak learnability* by Robert E. Schapire at https://doi.org/10.1007/BF00116037