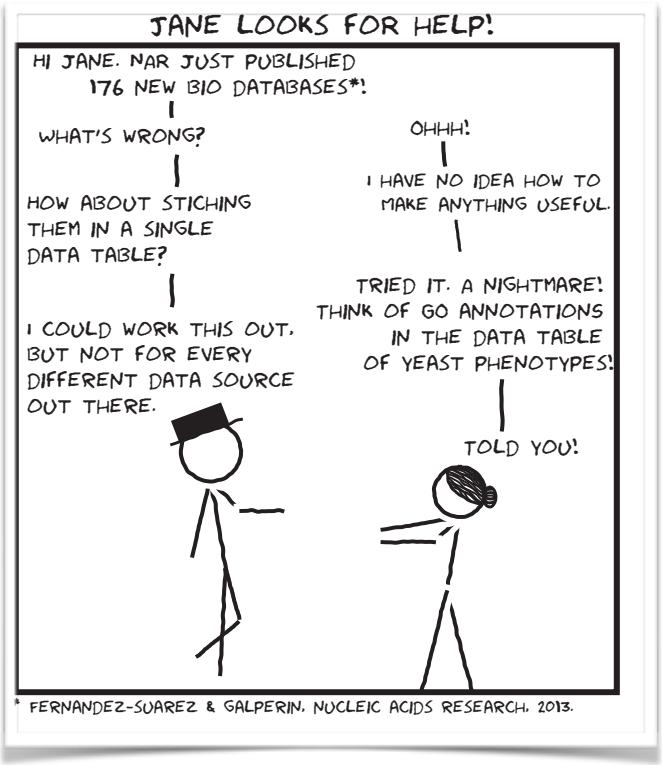
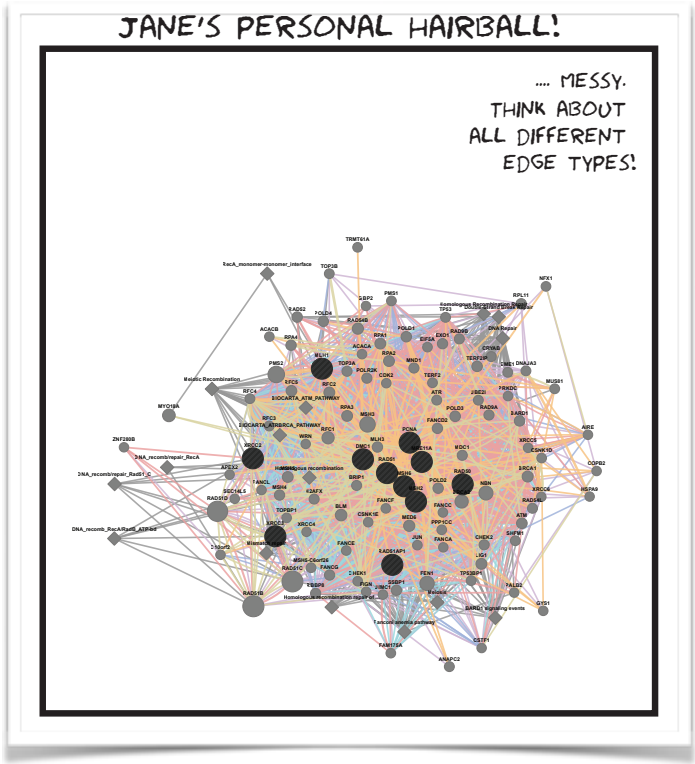




Large-scale data fusion

by collective matrix factorization

Tutorial at the Basel Computational Biology Conference,

Basel, Switzerland, 2015

These notes include introduction to integrative data analysis with examples from collaborative filtering and systems biology, and Orange workflows that we will construct during the tutorial.

Tutorial instructors:
Marinka Zitnik and Blaz Zupan, with the help from members of Bioinformatics Lab, Ljubljana.

Welcome to the hands-on Data Fusion Tutorial! This tutorial is designed for data mining researchers and biologists with interest in data analysis and large-scale data integration. We will explore latent factor models, a popular class of approaches that have in recent years seen many successful applications in integrative data analysis. We will describe the intuition behind matrix factorization and explain why factorization approaches are suitable when collectively analyzing many heterogeneous data sets. To practice data fusion, we will construct visual data fusion workflows using Orange and its Data Fusion Add-on.

If you haven't already installed Orange, please follow the installation guide at http://biolab.github.io/datafusion-installation-guide.

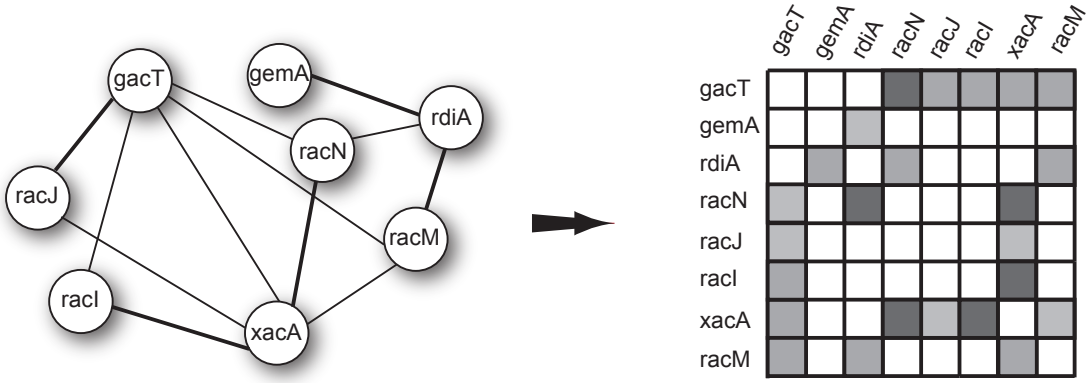
* See http://helikoid.si/recomb14/zitnik-zupan-recomb14.png for our full award-winning poster on data fusion.

Lesson 1: Everything is a Matrix

In many data mining applications there are plenty of potentially beneficial data available. However, these data naturally come in various formats and at different levels of granularity, can be represented in totally different input data spaces and typically describe distinct data types.

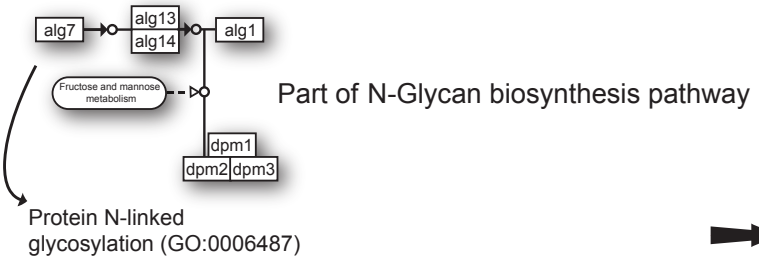
For joint predictive modeling of heterogeneous data we need a generic way to encode data that might be fundamentally different from each other, both in type and in structure. An effective way to organize a data compendium is to view each data set a *matrix*. Matrices describe *dyadic* relationships, which are relationships between two groups of objects. A matrix relates objects in the rows to objects in the columns. Examples of data matrices commonly used in the analysis of biological data include degrees of protein-protein interactions from the STRING database that are represented in a gene-to-gene matrix:

Gene interaction network can easily be converted to a matrix. Each wighted edge in a network corresponds to a matrix entry.



Binary matrices can be used to associate Gene ontology terms with cellular pathways:

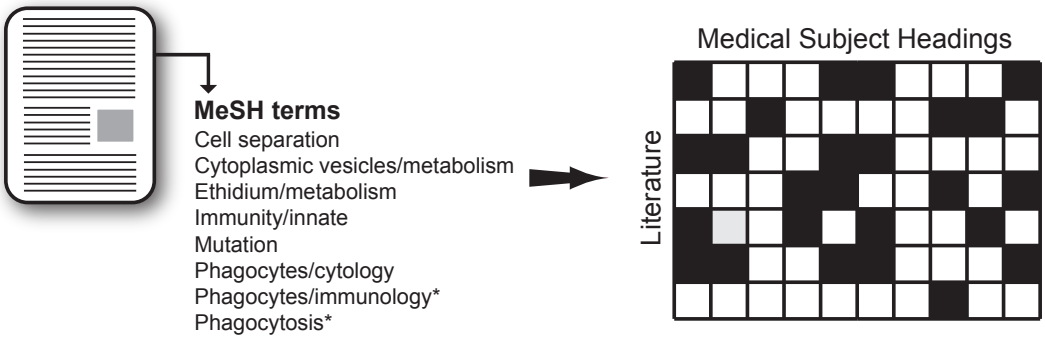
Binary relations between two object types can be represented with a binary matrix.



Orthology	Ontology
Dolichol kinase (K00902)	GO:0004168
Alpha-mannosidase II (K01231)	GO:0004572
Oligosaccharyltransferase complex (K12668)	GO:0008250

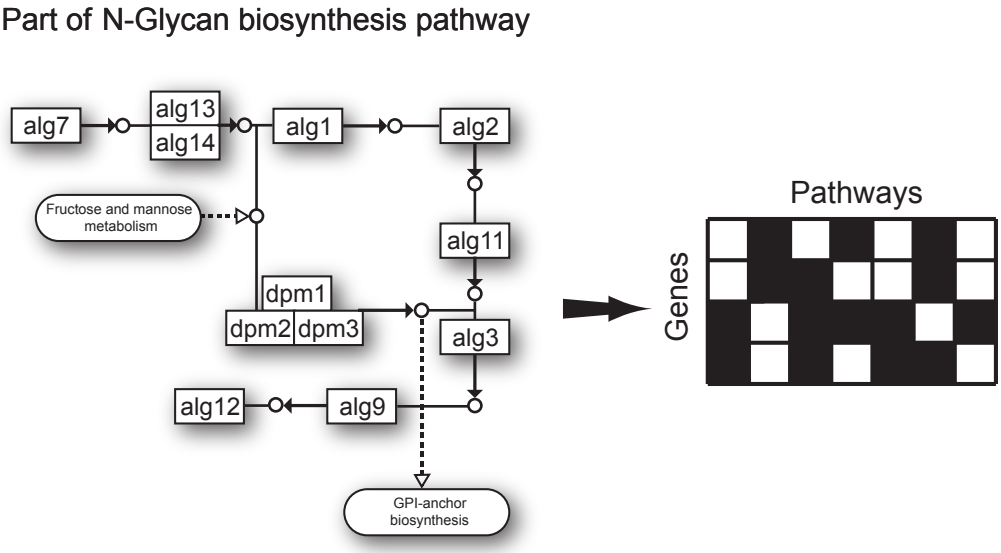
research articles with Medical Subject Headings (MeSH):

Papers cited in PubMed are tagged with MeSH terms. We can use one large binary matrix to encode relations between research articles and MeSH terms.



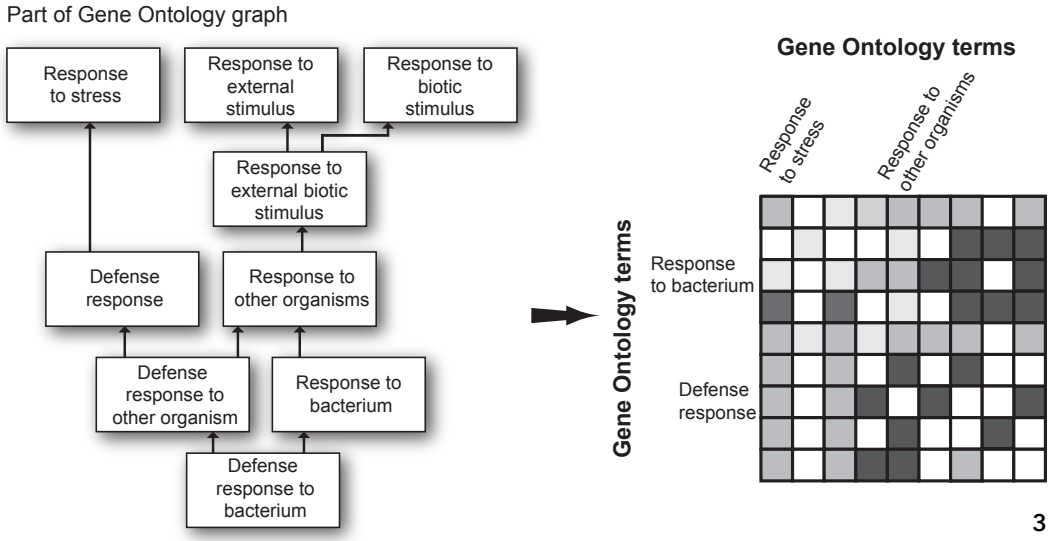
or membership of genes in pathway, one column for each pathway:

Just like the relations between MeSH terms and scientific papers, we can encode pathway memberships of genes in one large matrix that has genes in rows, pathways in commons.



The structure of Gene Ontology can be represented with a real-valued matrix whose elements represent distance or semantic similarity between the corresponding ontological terms:

Any ontology can be represented with a square matrix. We use ontology to measure distances between its entities, and encode these distances in a distance matrix.



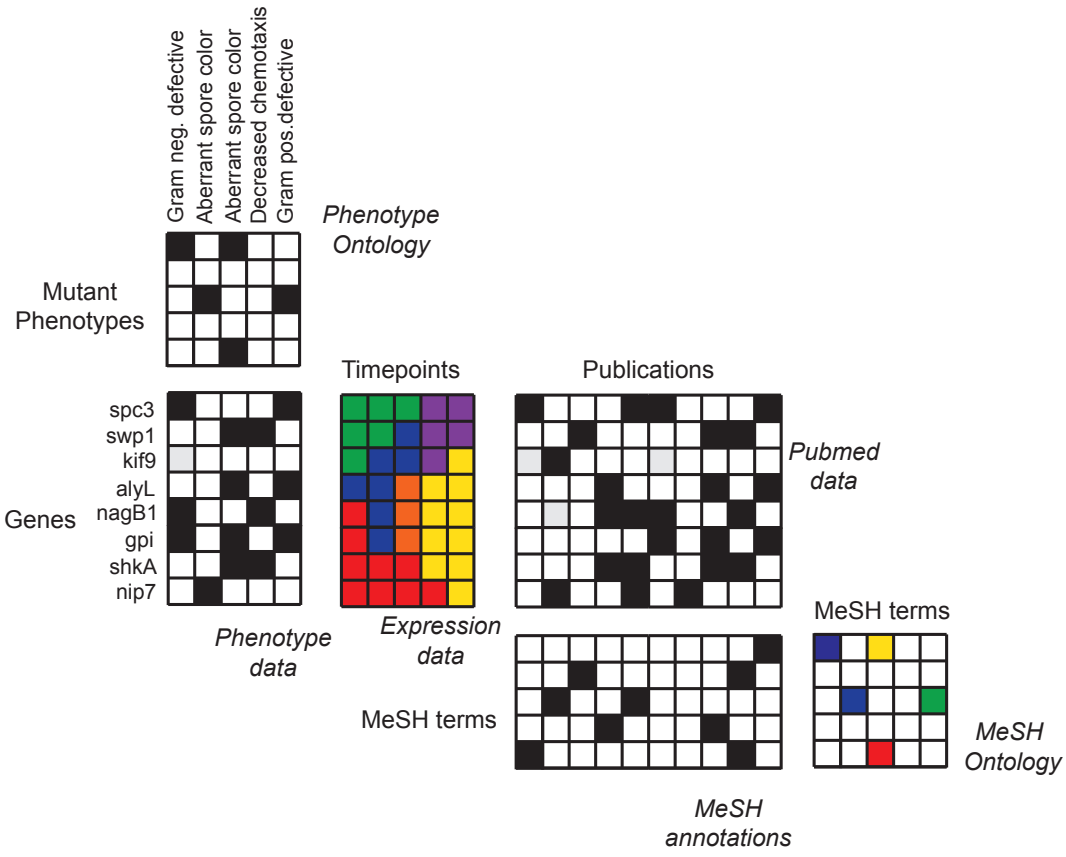
Lesson 2: The Challenge

Suppose we would like to identify genes whose mutants exhibit a certain phenotype, e.g., genes that are sensitive to Gram negative bacteria. In addition to current knowledge about phenotypical annotations, i.e. data encoded in a gene-to-phenotype matrix, which might be incomplete and contain some erroneous information, there exists a variety of circumstantial evidence, such as gene expression data, literature data, annotations of research articles etc.

An obvious question is how to *link these seemingly disparate data sets*. In many applications there exists some correspondence between different input dimensions. For example, genes can be linked to MeSH terms via gene-to-publication and publication-to-MeSH-term data matrices. This is an important observation, which we exploit to define a *relational structure of the entire data system.*

The major challenge for such problems is how to jointly model multiple types of data heterogeneity in a mutually beneficial way. For example, in the scheme below, can information about the relatedness of MeSH terms and similarity between phenotypes from the Phenotype Ontology help us to improve the accuracy of recognizing Gram negative defective genes?

The data excerpt on the right comes from a gene prioritization problem where our goal was to find candidates for bacterial response genes in a social amoeba *Dictyostelium*. Other than for a few seed genes, there was not any data from which we could directly infer the bacterial phenotype of mutants. Hence, we considered circumstantial data sets and hoped that their fusion would uncover interesting new bacterial response genes.



Lesson 3: Recommender Systems

Sparse matrices and matrix completion have been thoroughly addressed in the area of machine learning called recommender systems. Several methods from this field form foundation for matrix-based data fusion. Hence, we diverge here from fusion to recommender systems, and for a while, from biology to movies.

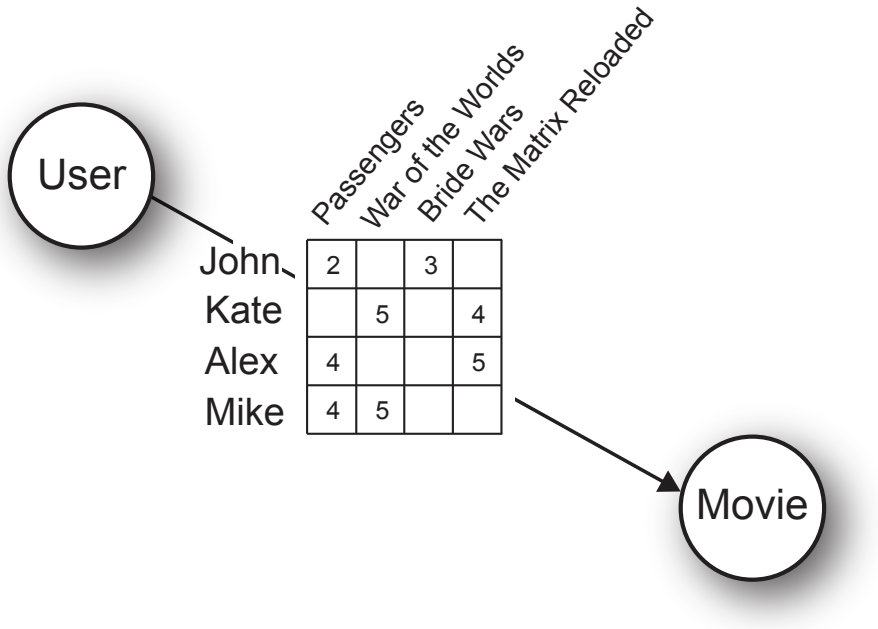
How would you decide which movie to recommend to a friend? Obviously, a useful source of information might be ratings of the movies your friend had seen in the past, i.e. one star up to five stars. Movie recommender systems primarily use user ratings information from which they estimate correlations between different movies and similarities between users and infer a prediction model which can be used to make recommendations about which movie a user should see next.

For example, in the figure below we see a movie ratings data matrix containing information for four users and four movies. Notice that in a real setting such matrices can contain information for millions of users and hundreds of thousands of movies. However, each individual user typically sees only a small proportion of all the movies and rates even fewer of them. Hence, data matrices in recommender systems are typically *extremely sparse*, e.g., it is common that up to ~99% of matrix elements are unknown. This characteristic together with a strong relational structure of the data, i.e. "you might enjoy movies, which users similar to you, are enthusiastic about" and "you might like movies that are similar to the movies you have already seen and rated favorably."

Is there an analogy between recommender systems and challenges in systems biology? We will answer these questions in the next lessons.

John, Kate, Alex and Mike rated a selection from four movies, Passengers, War of the Worlds, Bride Wars and The Matrix Reloaded. John, for example, has seen Passengers and Bride Wars, and did not like them so much. Which of the two other movies, if any, he should see?

The movie rating matrix has users in rows and movies in columns. We made this explicit in this simple graph: object types are represented as nodes (User, Movie) and an edge is labeled with a matrix that relates them.

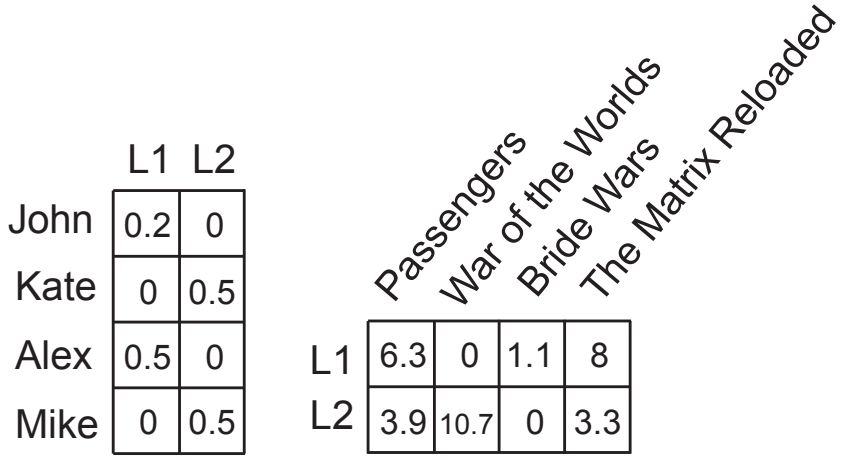


Lesson 4: Matrix Factorization and Completion

Taking our four-by-four movie ratings matrix we can try to *factorize* it into a product of two much smaller *latent matrices* called *latent factors*. One latent matrix describes latent profiles of the users and the other matrix contains latent data representation of the movies.

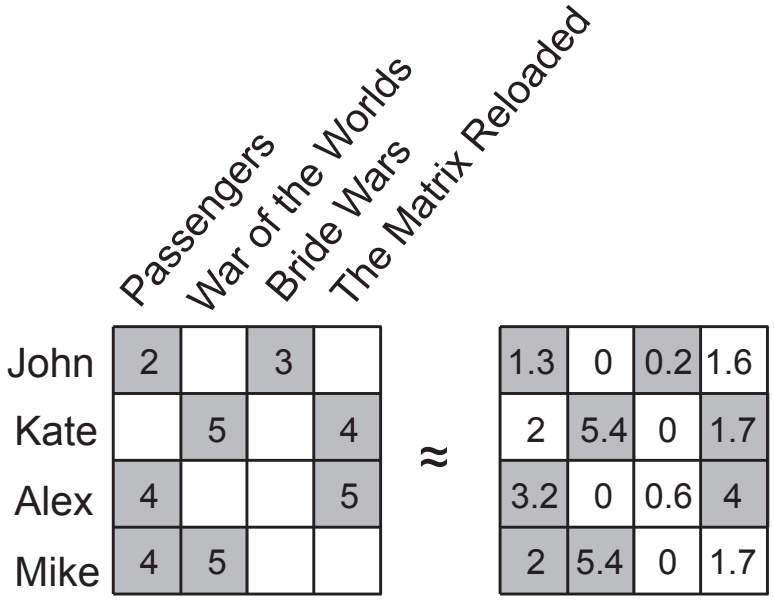
For example, each of our four users is described by a latent profile of length two and similarly, each movie is explained via two latent components, i.e. L1 and L2. The dimensionality of a latent model is typically called *factorization rank*.

Two-factorization of a user-movie rating matrix from the previous page. Factorization rank of 2 was used. Should factorization rank be the same for both latent matrices?



The challenge of matrix factorization stems from the difficulty of estimating the latent matrices in a way that their matrix product minimizes some *measure of discrepancy* between the input data matrix and its reconstruction obtained by factorization. Importantly, reconstructed matrix is *complete*, i.e. all of its elements are defined, which we exploit for making predictions.

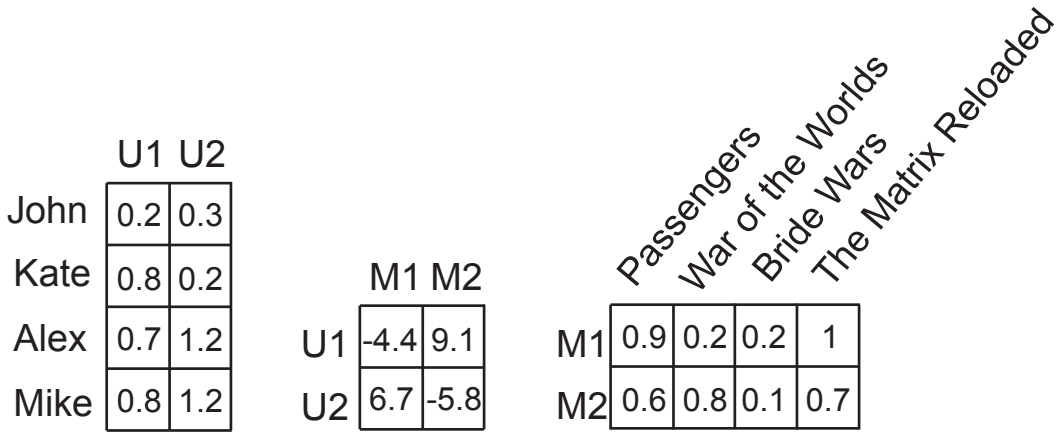
Latent model, that is, the two latent matrices are complete, hence their product is also a complete matrix. This product (matrix on the right) is an estimate of an original matrix (matrix on the left). How good is our reconstruction? Which of the two movies should be recommend to Mike?



Lesson 5: Matrix Tri-Factorization

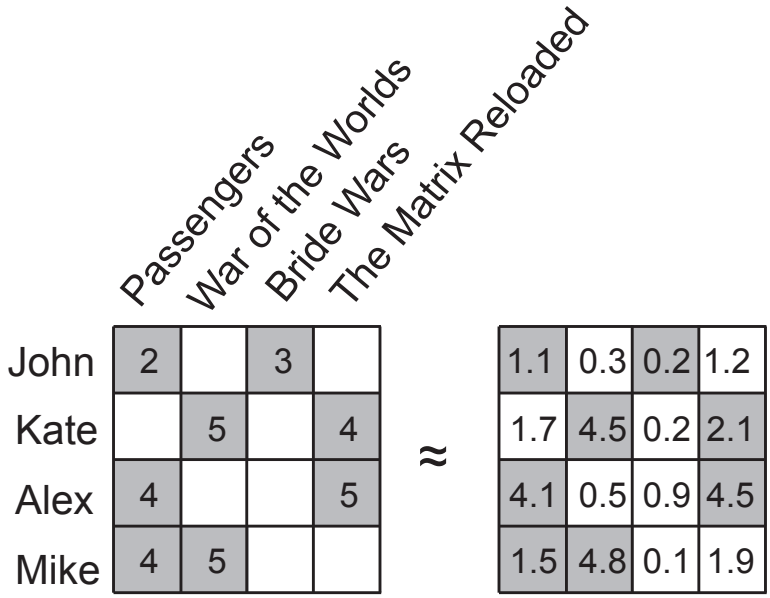
So far, we found a decomposition of the movie ratings matrix into *two* latent matrices. An alternative approach is to factorize it into *three* latent matrices; one latent matrix that expresses the degrees of user membership to each of the latent components, i.e. *user recipe matrix;* another latent matrix with memberships of movies to movie-specific latent components, i.e. *movie recipe matrix.* A third matrix, i.e. *a backbone matrix,* captures the interactions between latent components specific to the users, i.e. U1, U2, and components specific to the movies, i.e. M1, M2.

The backbone matrix (a 2x2 matrix in the middle) could be seen as a compressed version of original user-movies rating matrix. It has "meta" users in rows and "meta" movies in columns. We can use the two recipe matrices (left and right matrix) to transform the backbone matrix back to the original user-movies space.



Similarly to the previous lesson, the goal of matrix ti-factorization is to estimate three latent matrices that provide a quality approximation of observed entries in the input data matrix. By selecting sufficiently small factorization rank, we *compress* the data, which ensures *generalization* and consequently prediction of how a given user would enjoy a particular movie he has not seen before.

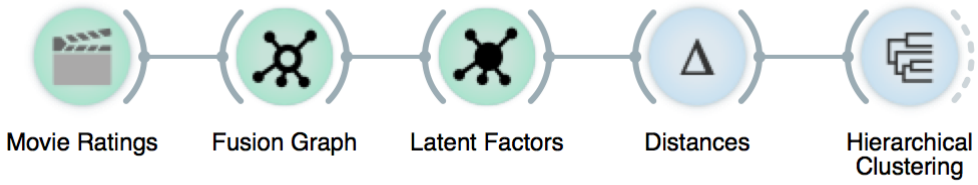
Just like for two-factorization, in tri-factorization the latent matrices are complete. So is their product (the matrix on the right). This product is also an estimate of the original matrix. How good is our estimate? Which movie should be recommended to Mike?



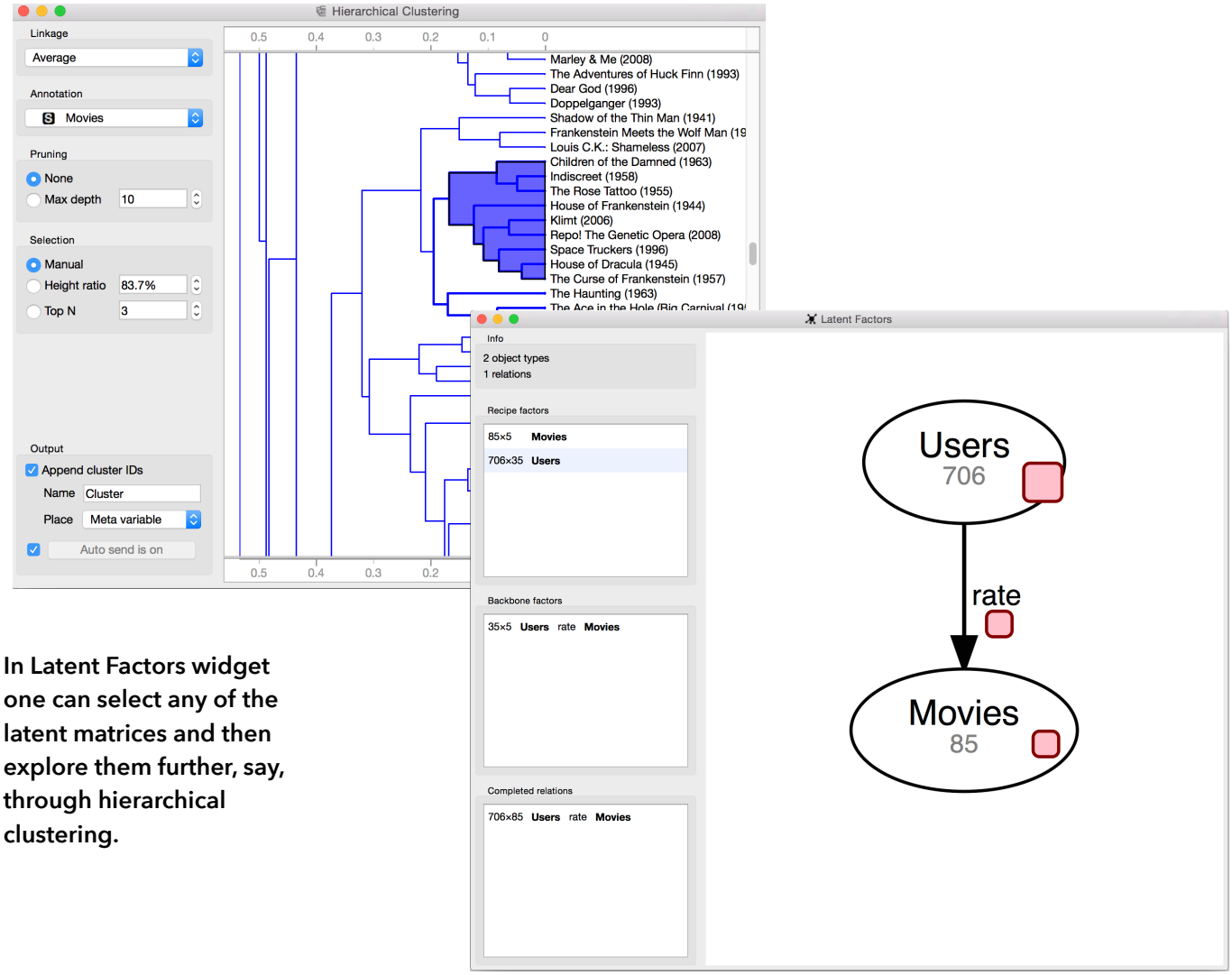
Lesson 6: Tri-Factorization in Orange

In Orange workflows its components (widgets) load or process data and pass the information to other widgets. Widgets inputs are on its left, and outputs on its right. Try adding a Data Table widget to display an input data set and any of the latent factors!

Let's try matrix tri-factorization in practice. We construct a visual workflow in Orange, a data mining suite. The workflow loads movie ratings, represents them with a data matrix, tri-factorizes it and explores the latent factors.



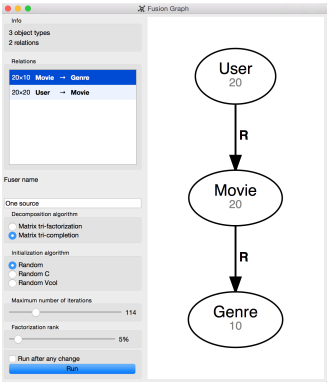
In this tutorial we organize the data sets using a structure that we call *a data fusion graph*. It shows the relational structure of entire data collection. Each distinct type of objects, e.g., users, movies, is represented with a node and each data set corresponds to an edge that relates two types of objects, e.g., movie ratings data relate users with the movies.



In Latent Factors widget one can select any of the latent matrices and then explore them further, say, through hierarchical clustering.

Lesson 7: *Collective* Matrix Factorization and *Sharing* of Latent Factors

Orange workflow on this page adds another data source: movie genres. How does that effect the results of the movie clustering?

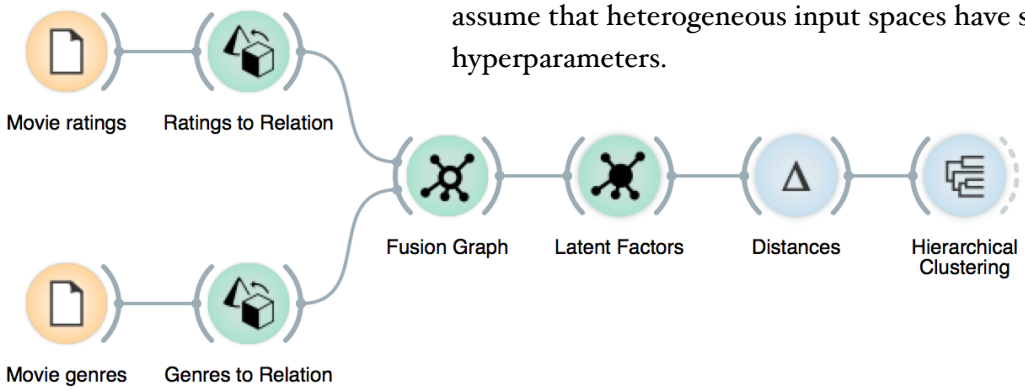


In the previous lesson we analyzed a single data set. Ultimately, we would like to *collectively tri-factorize* many heterogeneous data sets across different input spaces.

Suppose we have collected information about movie genres. This is a *relation* that relates movies to genres, hence our data fusion graph gets an additional node, i.e. genres, and an edge linking movies with genres.

To fuse heterogeneous data at large scales we need to define *the kind of knowledge that can be transferred* between related data matrices, types of objects and prediction tasks. Data fusion algorithms typically rely on one of the following three assumptions:

> *Relation transfer:* We build the relational map called a data fusion graph of all the relations considered in data fusion and relax the assumptions about independently and identically distributed relations.
>
> *Object type transfer:* We assume that there exists a common feature space shared by the input spaces, which can be used as a bridge to transfer knowledge.
>
> *Parameter transfer:* We make use of latent model parameterization and assume that heterogeneous input spaces have shared latent parameters and hyperparameters.



In collective matrix factorization we achieve data fusion by *sharing* latent matrices across related data sets.

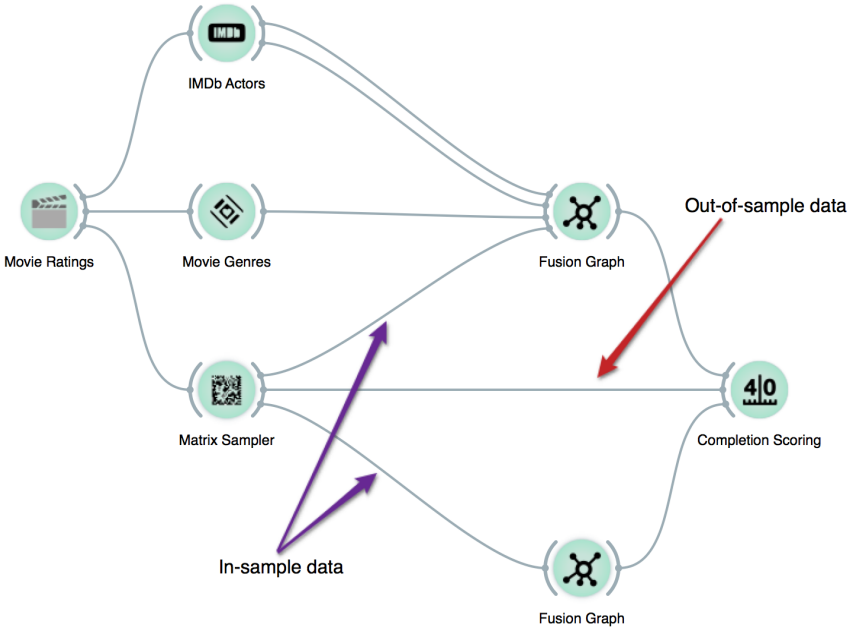
In our running example we reuse the movie recipe matrix in both decompositions of user-to-movie as well as movie-to-genre matrices. Importantly, collective matrix factorization estimates the latent matrices for all data sets in a compendium *simultaneously,* which ensures transfer of knowledge between data, i.e. data fusion, and presents many unique opportunities from the application perspective and challenges in algorithmic design.

Lesson 8: More Complex Fusion Schemes, Data Sampling and Completion Scoring

So far we fused at most two data sets. Let's proceed by constructing a larger data compendium. There are many other sources of information that might be informative for movie recommendation, for example, user demographics profiles, movie casting, information about movie directors and screenplays, scenery, etc.

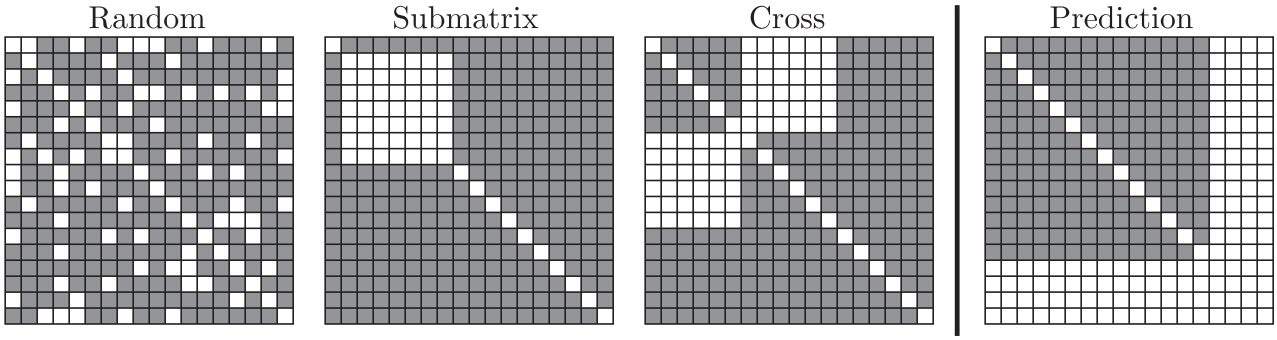
We construct an Orange workflow that considers four data sources, i.e. movie ratings, movie casting, genres and relationships between actors, and fuse them via collective matrix factorization.

This data fusion configuration is already a complex one. We are using four different data sources. Try having a Fusion Graph widget window open, so that you can see the data fusion schema as it shapes up when adding the data sets.

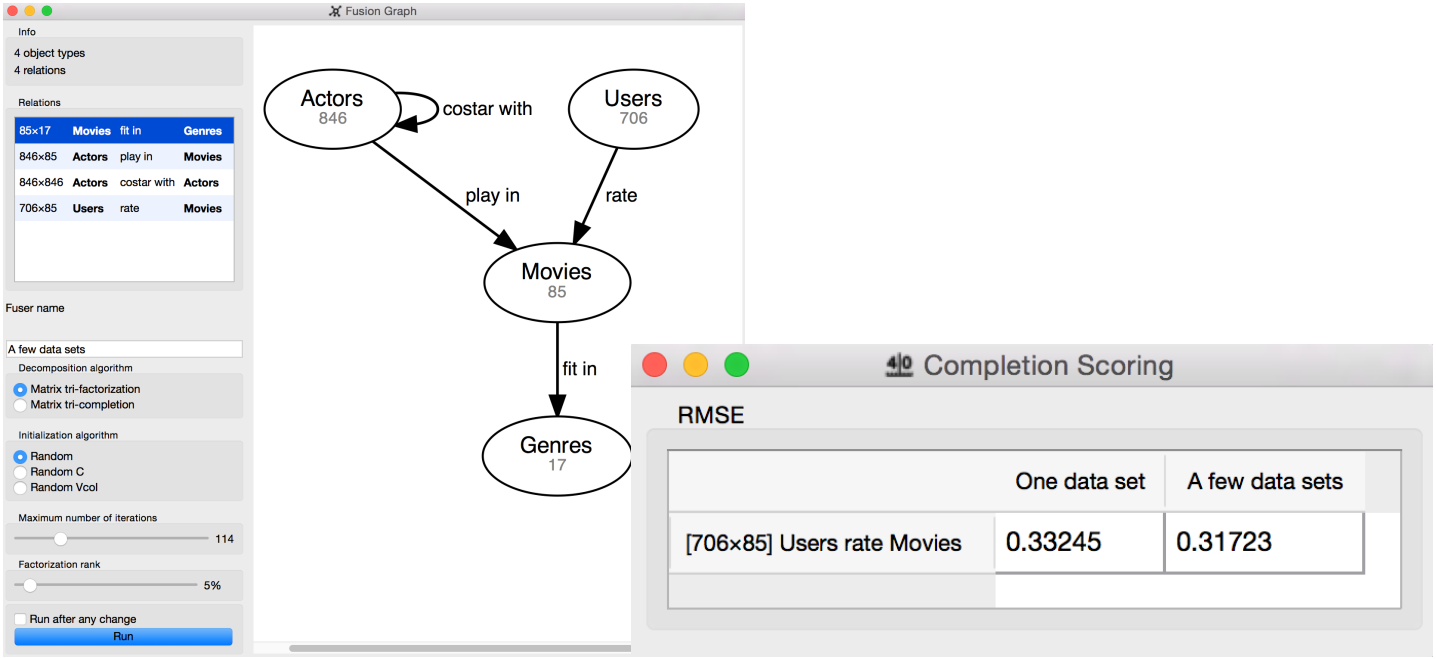


A simple way to assess the benefits of integrative data analysis over the analysis of a single homogeneous data set is to measure the quality of predictions made by data fusion versus the quality of prediction model inferred from a part of data collection.

The assessment is fair if we evaluate predictions for *data that are hidden from the algorithm during prediction model inference.* There are four different ways of partitioning a data matrix into a training and a test set:



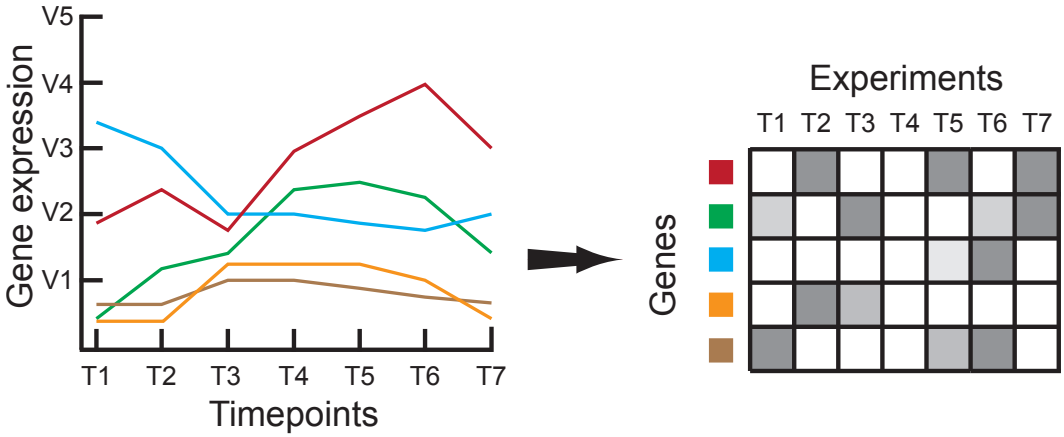
In predictive modeling tasks, such as movie recommendation, where we *regress* against the target variable, i.e. movie rating, we can evaluate model quality by reporting a variety of measures, including the root mean squared error (RMSE). A lower RMSE value indicate a better model. Alternatively, if our goal would be to rank the movies from what the model believes are the most enjoyable to the least enjoyable for a given user, we would use the area under curve (AUC).



How does the quality of reconstruction change when adding or removing data sets from the fusion schema? Try it out! Should RMSE always decrease with new data sources being added?

Lesson 9: "Meta Genes" - Latent Profiling

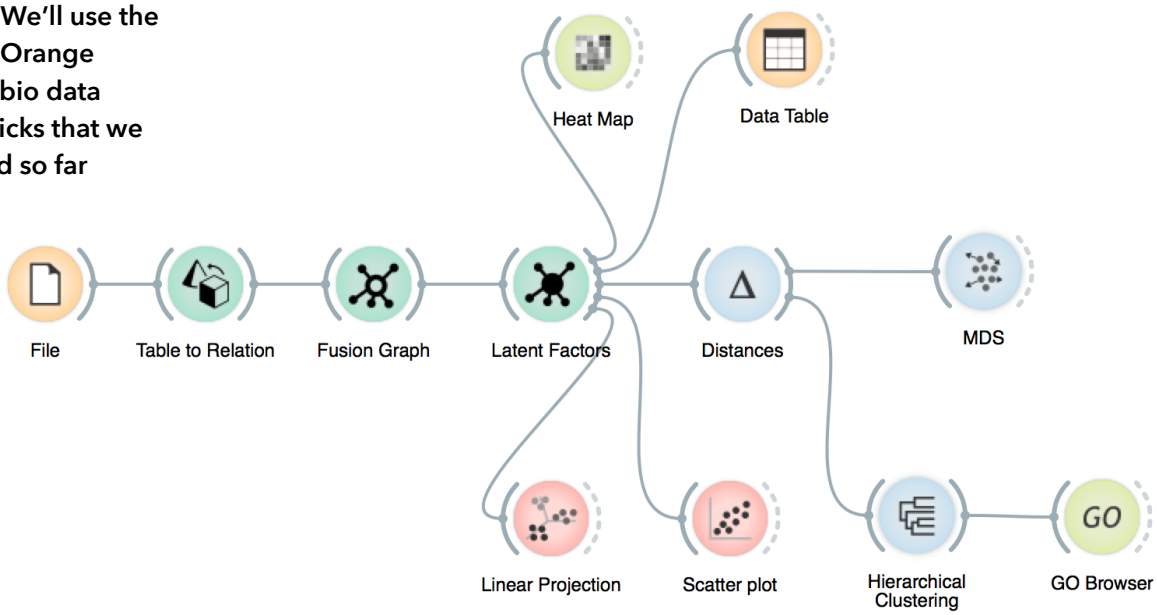
Until now we focused on non-biological data. We now apply a latent factor model to gene expression data. The microarray data for this example is from an influential paper by DeRisi, Iyer, and Brown (Science 1997), who explored the metabolic and genetic control of gene expression on a genomic scale. The authors used DNA microarrays to study temporal gene expression of almost all genes in baker's yeast *Saccharomyces cerevisiae* during the metabolic shift from fermentation to respiration. Expression levels were measured at seven time points during the diauxic shift, e.g. T1 to T7.



As we will see in this and in the next lessons, collective matrix factorization is a generic and flexible tool for integrative data analysis in different domains, e.g., recommender systems and functional genomics.

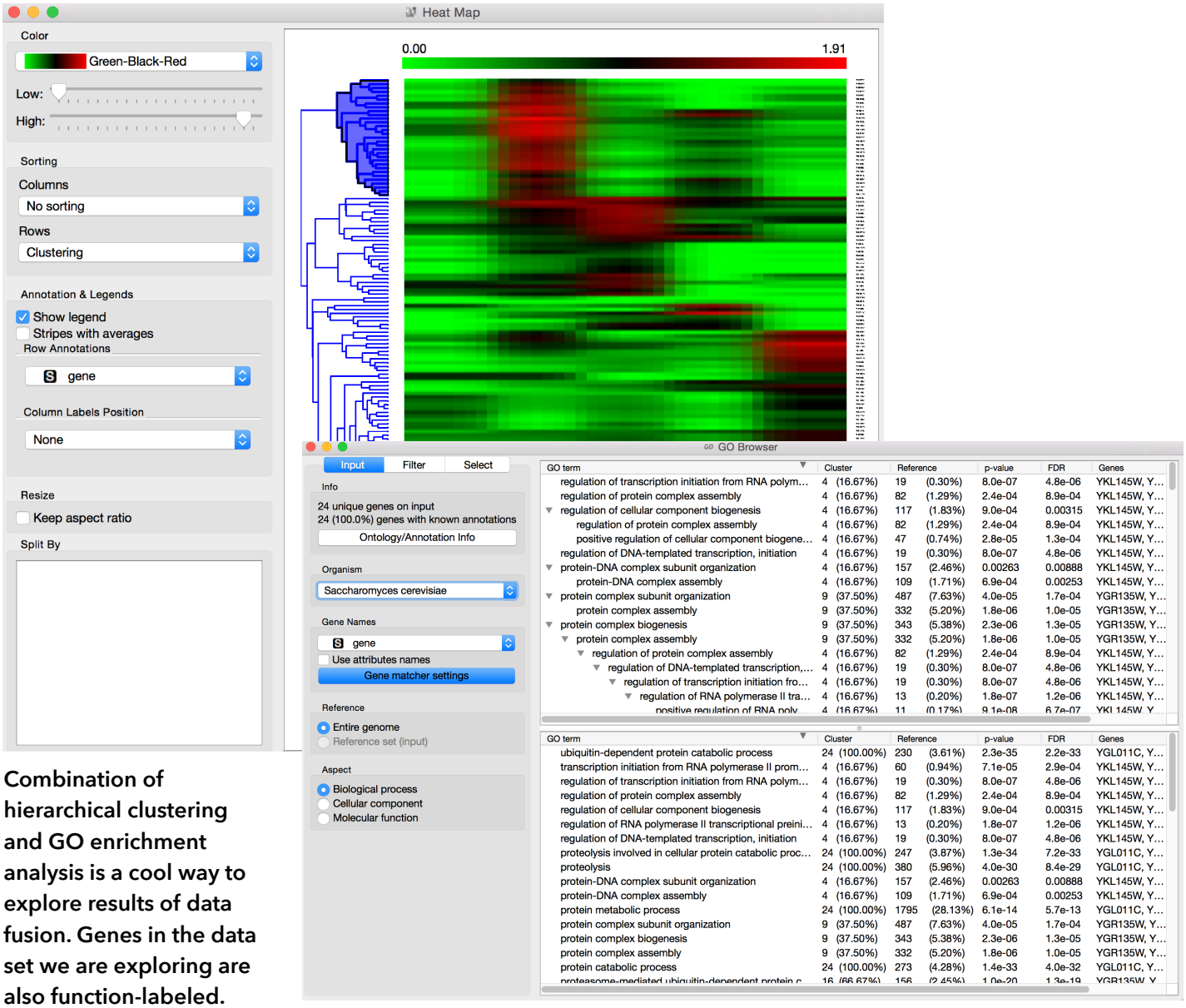
What is similar between matrix-based movie recommendation system and data fusion in molecular biology? Everything! We'll use the same set of Orange widgets for bio data fusion. All tricks that we have learned so far apply.

We construct an Orange workflow that reads the expression data into Orange using Table to Relation widget, tri-factorizes the data, and explores the estimated latent data representation using various Orange widgets, such as Linear Projection, Scatter Plot and Multi-dimensional Scaling (MDS).



By factorizing gene expression data we obtained three latent data matrices: a gene recipe matrix, an experiment recipe matrix, and a backbone matrix that relates both recipe matrices in the latent space.

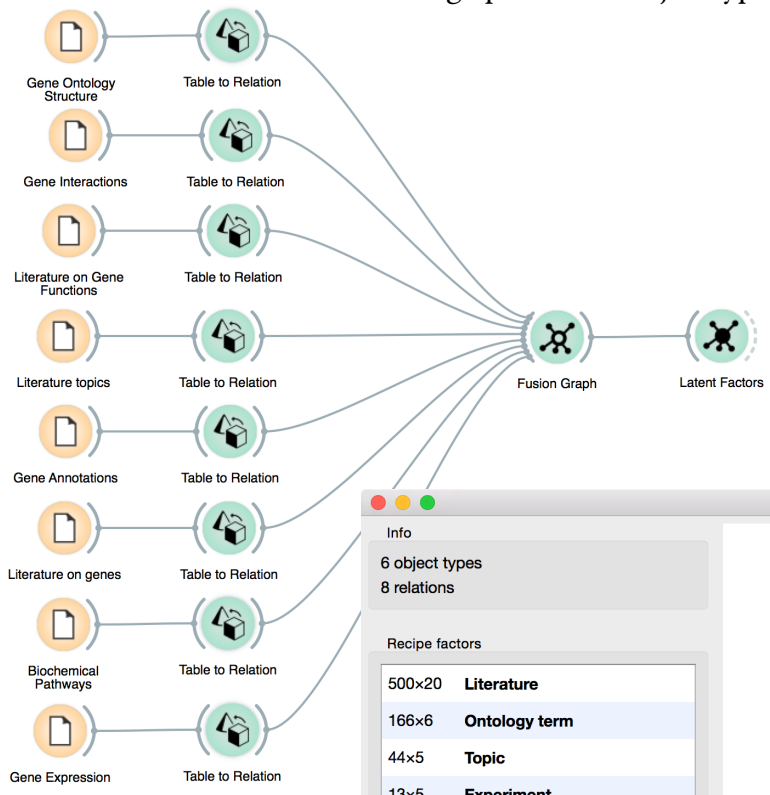
It is common in matrix factorization algorithms to interpret the experiment recipe matrix as a matrix that reports on expression of "meta genes," i.e. "genes," whose profiles are obtained from the original gene expression profiles by a linear (or, non-linear, depending on a latent factor model) transformation. Similarly, one can see the gene recipe matrix as a matrix that reports on expression of genes in "meta experiments," i.e. "experiments," which cannot be interpreted in an intuitive manner but which can improve the quality of prediction models applied to them, e.g., clustering of genes based on their recipe matrix and enrichment analysis of detected clusters.



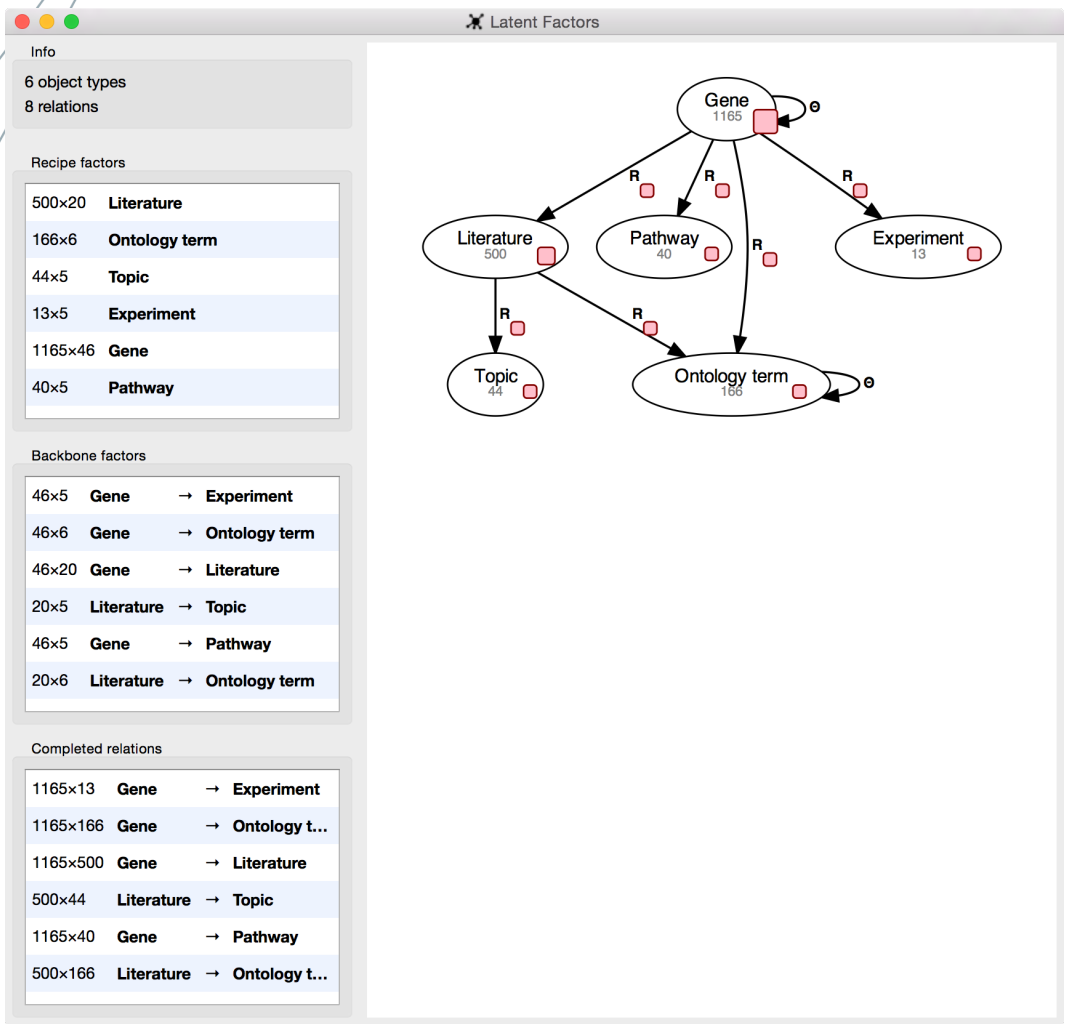
Combination of hierarchical clustering and GO enrichment analysis is a cool way to explore results of data fusion. Genes in the data set we are exploring are also function-labeled. Any other ideas how to use latent matrices? Classification, perhaps? And then estimation of AUC in cross-validation?

Lesson 10: The Yeast Case Study

Next, we collectively analyze eight data sets from molecular biology of yeast *S. cervisiae* (load the data sets from http://bit.ly/1Gb8SJ7). We organize them in a data fusion graph with six object types and eight edges, one for each data set.



This schema looks boring. But it offers so much for the patient one! Try adding matrix sampling and RMSE-based evaluation! Or clustering with gene set enrichment. Or data projection based on any of the latent matrices.



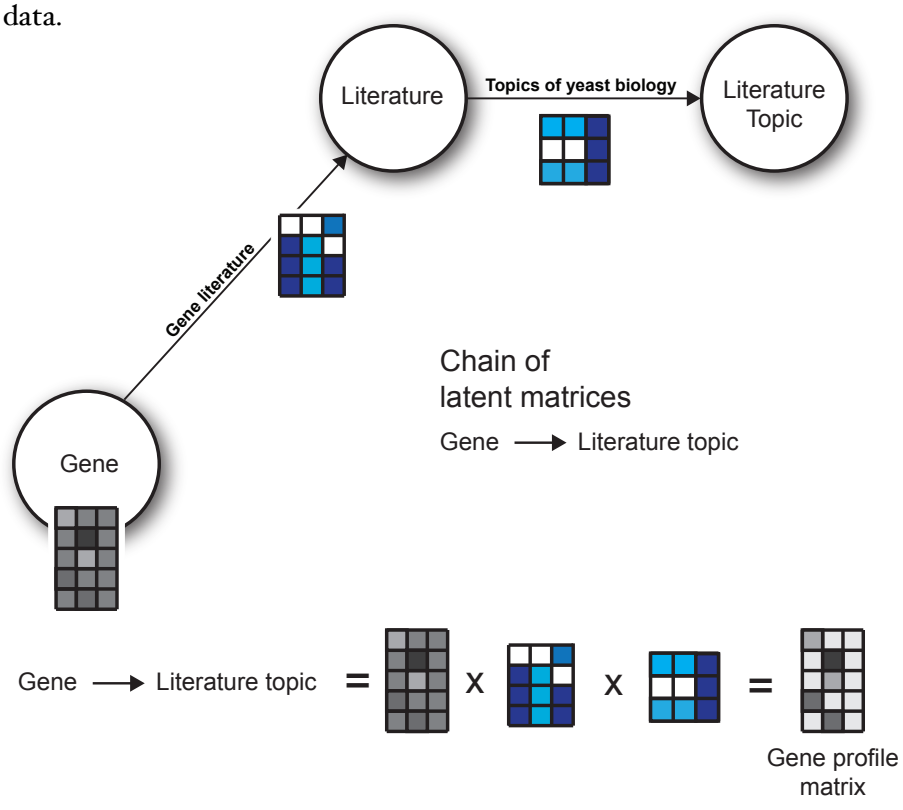
Lesson 11: Latent Matrix Chaining

The concept of *chaining latent matrices* is important because it allows us to profile objects in the latent space of any other object type based on the connectivity in the data fusion graph.

In the simplest scenario, where object types are adjacent in the fusion graph, e.g., "Genes" and "Experiments" from Lesson 9, chaining construct data profiles of one object type, e.g., genes, in the latent space of another object type, e.g., experiments, by multiplying the recipe matrix of the first object type by the backbone matrix of the data set. The resulting profile matrix has objects of the first type, e.g., genes, in rows and the latent components of the second type, e.g., experiments, in columns.

However, the power of chaining becomes apparent when we would like to *profile objects whose types are not direct neighbors in the fusion graph*, such as "Genes" and "Literature Topics," i.e. MeSH terms, in the fusion graph from Lesson 10. To profile genes in the latent space of literature topics chaining starts with the recipe matrix of genes and multiplies it by backbone matrices of gene-to-literature and literature-to-literature-topic data sets on the path from "Genes" to "Literature Topics" in the fusion graph. This procedure yields profiles of genes in the latent space of literature topics.

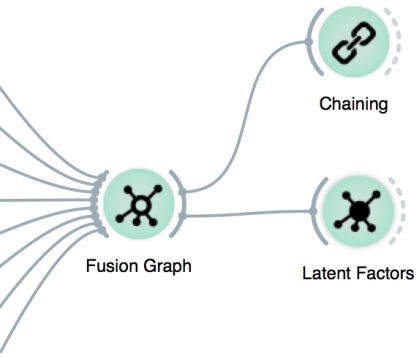
Latent matrix chaining constructs dense profiles that include the most informative features obtained by collectively compressing data via matrix factorization. Intuitively, chaining is able to establish links between genes and literature topics even though relationships between these object types are not available in input data.

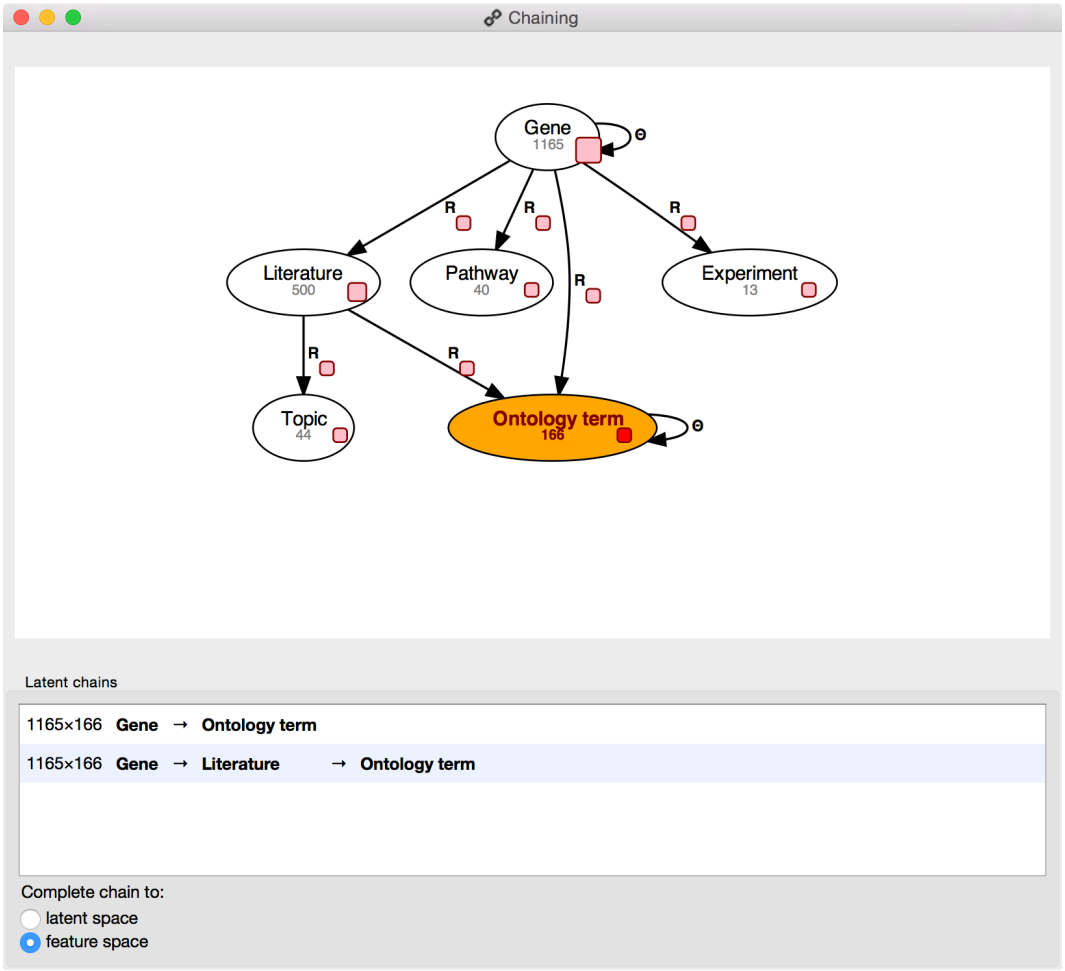


A conceptual presentation of profiling of genes in the latent space of MeSH terms. The MeSH-based gene profiles are constructed by multiplying latent factors on the path from one to another object type.

In Orange we chain the latent matrices of a data system using the Chaining widget.

Chaining widget allows us to select a start object type and a target object type (highlighted in orange below) in the latent fusion graph. It then computes the chains associated with selected nodes from the fusion graph. The so obtained profile matrices can be used for further data analysis.

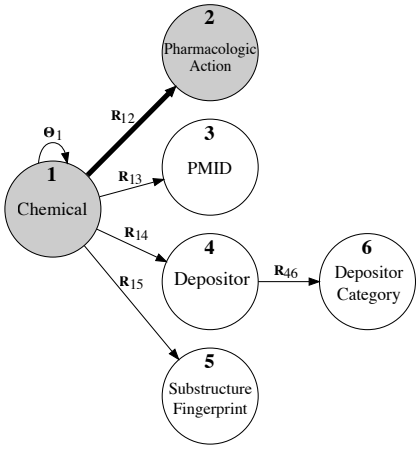




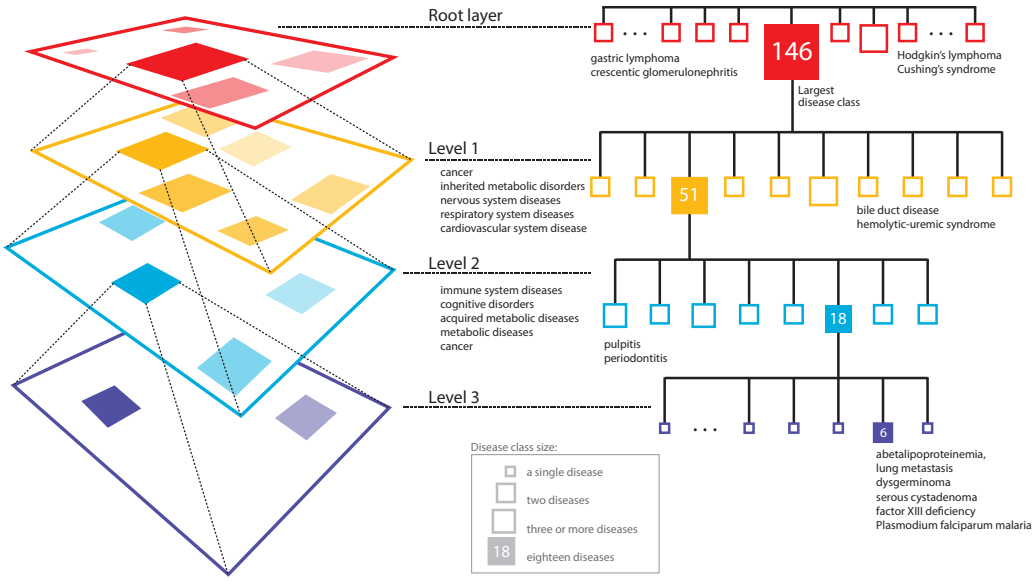
Chaining and construction of relations for objects that were originally not related in any input data set comes as an extra benefit of matrix-based data fusion. Try exploring the chaining results by feeding them into a data table first, and then push them through unsupervised or supervised analysis pipeline.

Lesson 12: Case Studies in Data Fusion

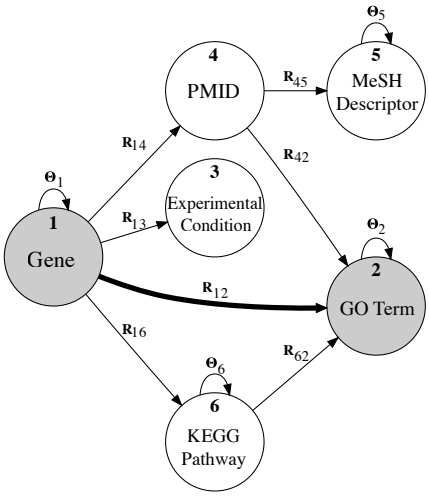
Identification of the mechanisms of action of chemical compounds is a crucial task in drug discovery. We have integrated 6 data sets to improve prediction pharmacologic actions of chemical compounds (IEEE TPAMI 2015).



We have fused 11 systems-level molecular data sets to predict disease-disease associations (Sci Reports 2013).

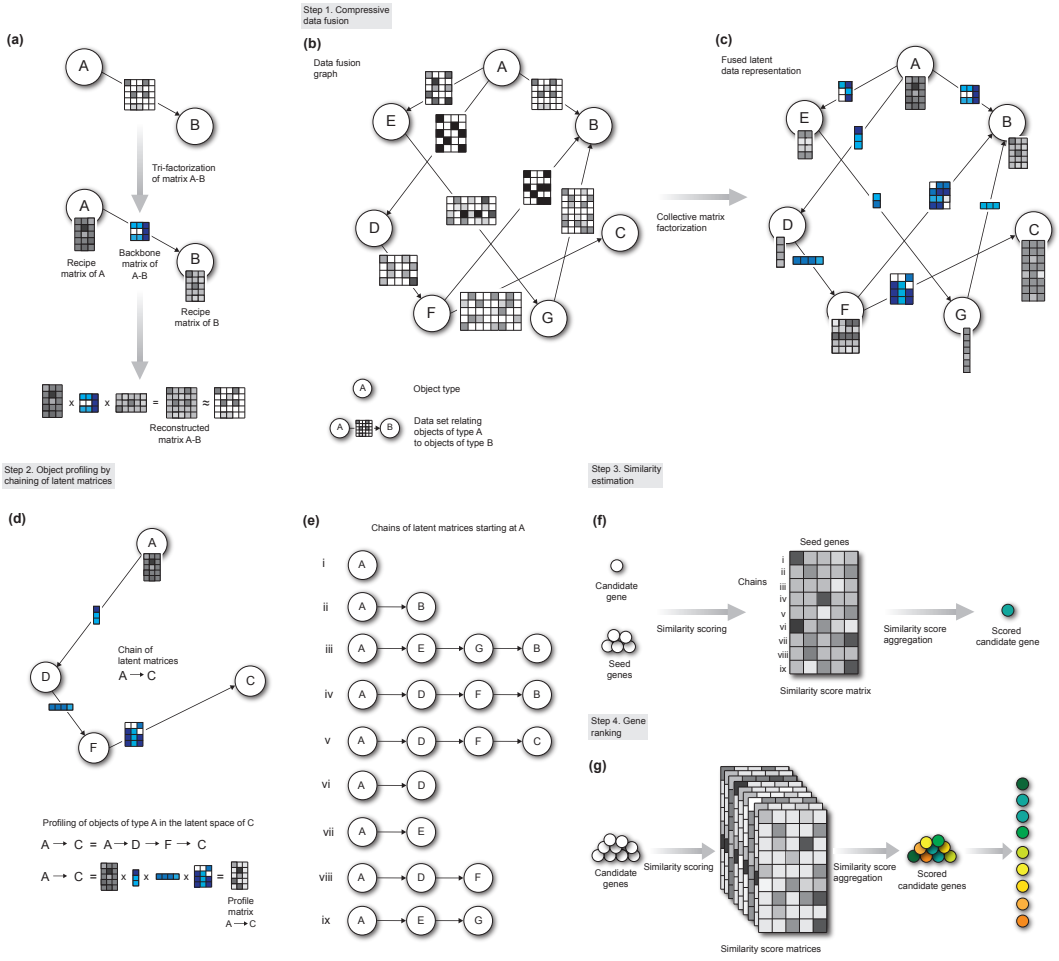


Data fusion of 11 data sets substantial raised the accuracy of gene function predictions, also when compared to kernel-based data integration approach (IEEE TPAMI 2015).

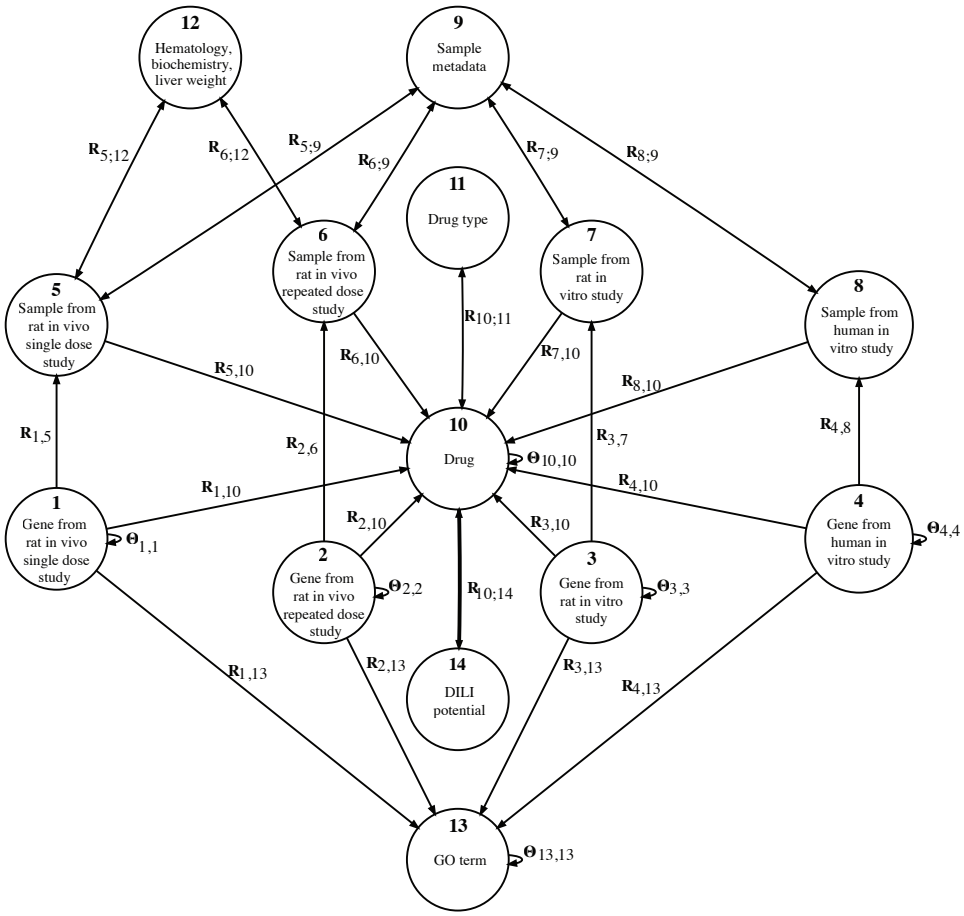


GO term name	Term identifier	Namespace	Size	DFMF		MKL		RF		tri-SPMF	
				F_1	AUC	F_1	AUC	F_1	AUC	F_1	AUC
Activation of adeny. cyc. act.	0007190	BP	11	0.834	0.844	0.770	0.781	0.758	0.601	0.729	0.731
Chemotaxis	0006935	BP	58	0.981	0.980	0.794	0.786	0.538	0.724	0.804	0.810
Chemotaxis to cAMP	0043327	BP	21	0.922	0.910	0.835	0.862	0.798	0.767	0.838	0.815
Phagocytosis	0006909	BP	33	0.956	0.932	0.892	0.901	0.789	0.619	0.836	0.810
Response to bacterium	0009617	BP	51	0.899	0.870	0.788	0.761	0.785	0.761	0.817	0.831
Cell-cell adhesion	0016337	BP	14	0.883	0.861	0.867	0.856	0.728	0.725	0.799	0.834
Actin binding	0003779	MF	43	0.676	0.781	0.664	0.658	0.642	0.737	0.671	0.682
Lysozyme activity	0003796	MF	4	0.782	0.750	0.774	0.750	0.754	0.625	0.747	0.625
Seq.-spec. DNA bind. t. f. a.	0003700	MF	79	0.956	0.948	0.894	0.901	0.732	0.759	0.892	0.852

Prioritization of genes in a quest to identify the most promising candidates for bacterial response in *Dictyostelium* fused 13 input data sets. Out of 9 top-rated candidates, 8 predictions were confirmed in the wet lab (submitted, 2015).



In drug toxicity prediction the task was to distinguish between compounds that represent little or no health concern and those with the greatest likelihood to cause adverse effects in humans (CAMDA 2013). High-throughput and toxicogenomic screening coupled with a plethora of circumstantial evidence provide a challenge for improved toxicity prediction and require appropriate computational methods that integrate various biological, chemical and toxicological data. Fusion of 29 data sets allowed us to improve prediction accuracy well above that achieved by standard supervised approaches (Sys Biomed 2014).



Lesson 13: Related Work on Data Fusion

Lanckriet, Gert R.G., *et al.* A statistical framework for genomic data fusion. *Bioinformatics* 20.16 (2004): 2626-2635. **The first study to propose a kernel-based integration as a way of intermediate data integration.**

Schadt, Eric E., *et al.* An integrative genomics approach to infer causal associations between gene expression and disease. *Nature Genetics* 37.7 (2005): 710-717. **This study integrated DNA variation and gene expression data to identify drivers of complex traits.**

Aerts, Stein, *et al.* Gene prioritization through genomic data fusion. *Nature Biotechnology* 24.5 (2006): 537-544. **The paper describes Endeavour, a tool to prioritize candidate genes underlying biological processes or diseases, based on their similarity to known genes involved in these phenomena.**

Mostafavi, Sara, *et al.* GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology* 9.Suppl 1 (2008): S4. **GeneMANIA is a tool that integrates multiple functional association networks and predicts gene functions using label propagation.**

Zitnik, Marinka, *et al.* Discovering disease-disease associations by fusing systems-level molecular data. *Scientific Reports* 3 (2013). **A study of relationships between diseases based on evidence from fusing available molecular interaction and ontology data.**

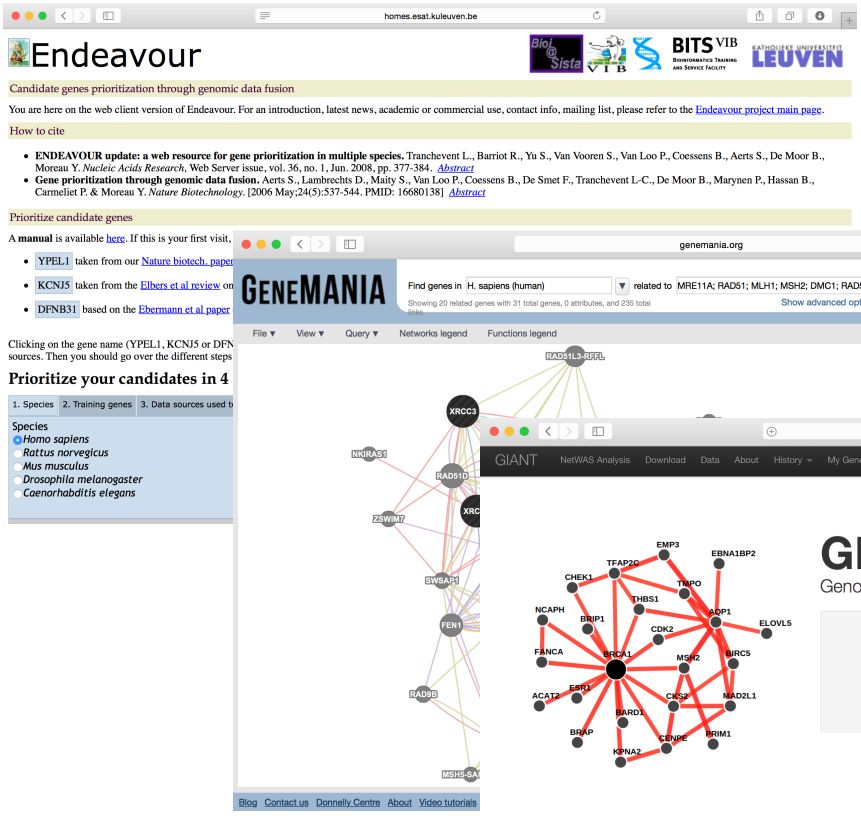
Wang, Bo, *et al.* Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* 11.3 (2014): 333-337. **Fusion of cancer patient similarity networks by combining mRNA expression, DNA methylation and microRNA expression data.**

Zitnik, Marinka, and Zupan, Blaz. Matrix factorization-based data fusion for drug-induced liver injury prediction. *Systems Biomedicine* 2.1 (2014): 16-22. **An application of a data fusion approach for prediction of drug toxicity in humans using 29 data sets provided by the CAMDA 2013 Challenge.**

Ritchie, Marylyn D., *et al.* Methods of integrating data to uncover genotype-phenotype interactions. *Nature Reviews Genetics* 16.2 (2015): 85-97. **This review explores emerging approaches for data integration including multi-staged, meta-dimensional and factor analysis.**

Zitnik, Marinka, and Zupan, Blaz. Data fusion by matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.1 (2015): 41-53. **An introduction and formalization of collective matrix factorization as presented in this tutorial. The paper also provides mathematical derivation of optimization approach.**

Lesson 14: Related Tools for Data Fusion



Lesson 15: Data Fusion in Python

We have developed a scripting library in Python, which implements collective matrix factorization and completion, and is suitable for fusion of large data compendia.

The official source code repository is at http://github.com/marinkaz/scikit-fusion.

```
>>> from skfusion import fusion, datasets
>>> dicty = datasets.load_dicty()
>>> print(dicty)
FusionGraph(Object types: 3, Relations: 3)
>>> print(dicty.object_types)
{ObjectType("GO term"), ObjectType("Experimental condition"), ObjectType("Gene")}
>>> print(dicty.relations)
{Relation(ObjectType("Gene"), ObjectType("GO term")),
 Relation(ObjectType("Gene"), ObjectType("Gene")),
 Relation(ObjectType("Gene"), ObjectType("Experimental condition"))}
>>> dfmf = fusion.Dfmf(max_iter=100)
>>> dfmf.fuse(dicty)
```