Hands-on: MOFA (Multi-Omics Factor Analysis)

A Comprehensive Guide to Multi-Omics Integration

Data Preparation

Formatting multi-omics datasets

Model Training

Running MOFA analysis

Factor Interpretation

Understanding learned factors

Variance Decomposition

Attributing variance to factors

Downstream Analysis

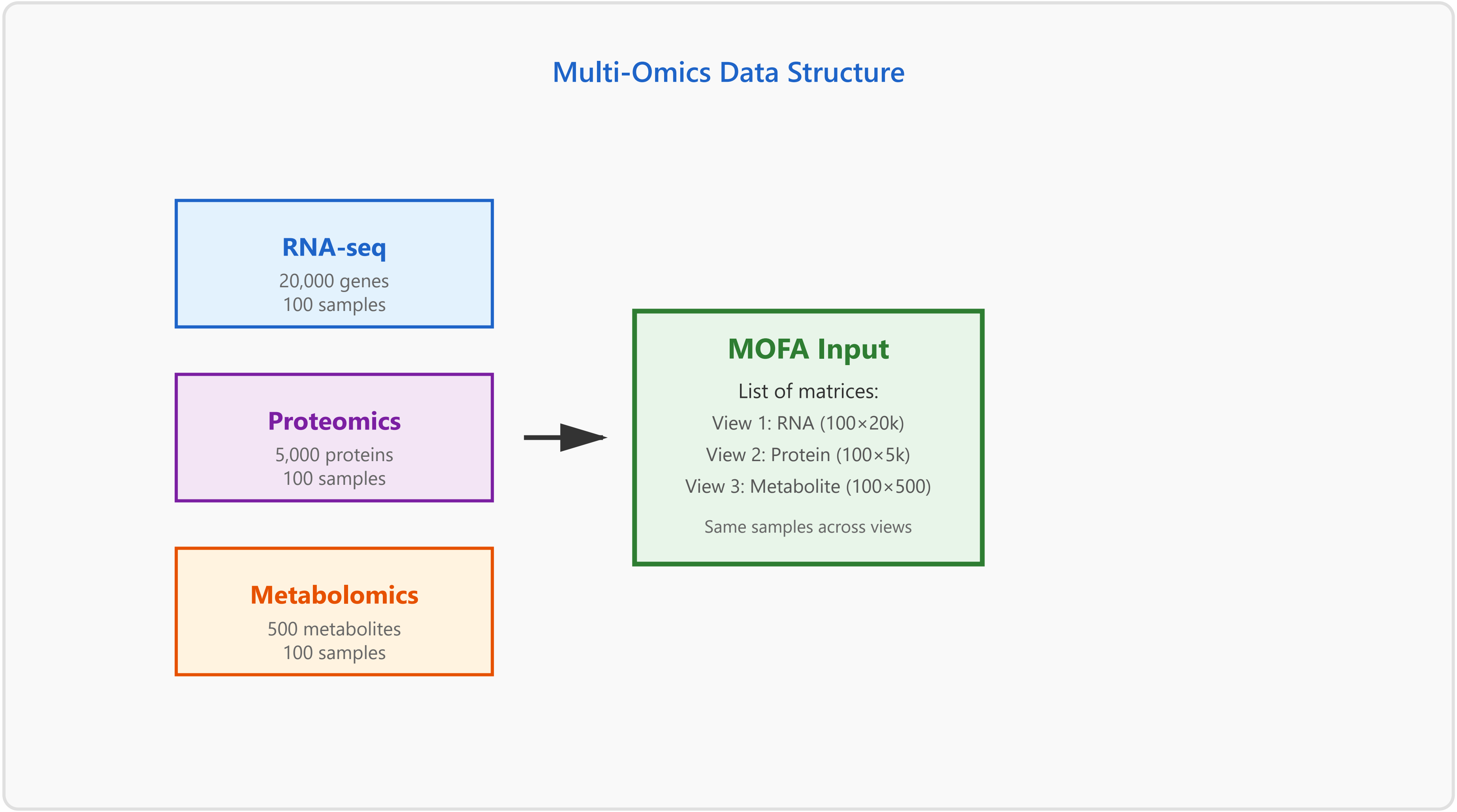
Using factors for prediction

1 Data Preparation

Data preparation is the foundational step in MOFA analysis. It involves organizing multiple omics datasets (e.g., genomics, transcriptomics, proteomics, metabolomics) into a structured format that MOFA can process. The key is to ensure that all datasets share the same samples while potentially having different features.

Key Requirements:

- **Sample alignment:** All omics layers must have measurements for the same set of samples
- **Data format:** Typically matrices where rows are samples and columns are features
- **Normalization:** Each omics layer should be appropriately normalized (log-transformation, standardization, etc.)
- **Missing data:** MOFA can handle missing values, but extreme missingness should be addressed
- **Feature selection:** Remove low-variance or non-informative features to improve computational efficiency



```
# Example R code for data preparation library(MOFA2) # Load your omics data rna_data <-
read.csv("rna_seq.csv", row.names=1) protein_data <- read.csv("proteomics.csv", row.names=1)
metabolite_data <- read.csv("metabolomics.csv", row.names=1) # Ensure samples match across datasets
common_samples <- Reduce(intersect, list(rownames(rna_data), rownames(protein_data),
rownames(metabolite_data))) # Create a list of matrices data_list <- list( "RNA" =
as.matrix(rna_data[common_samples, ]), "Protein" = as.matrix(protein_data[common_samples, ]),
"Metabolite" = as.matrix(metabolite_data[common_samples, ]) ) # Create MOFA object MOFAobject <-
create_mofa(data_list)
```

💡 Key Points

- Always verify sample matching across all omics layers before analysis
- Consider filtering features with high missingness (>50%)
- Apply appropriate transformations (log for count data, scaling for continuous data)
- Document preprocessing steps for reproducibility

2 Model Training

Model training in MOFA involves learning latent factors that capture coordinated variation across multiple omics layers. MOFA uses a Bayesian framework with automatic relevance determination to identify the optimal number of factors and their relevance to each omics view.

Training Process:

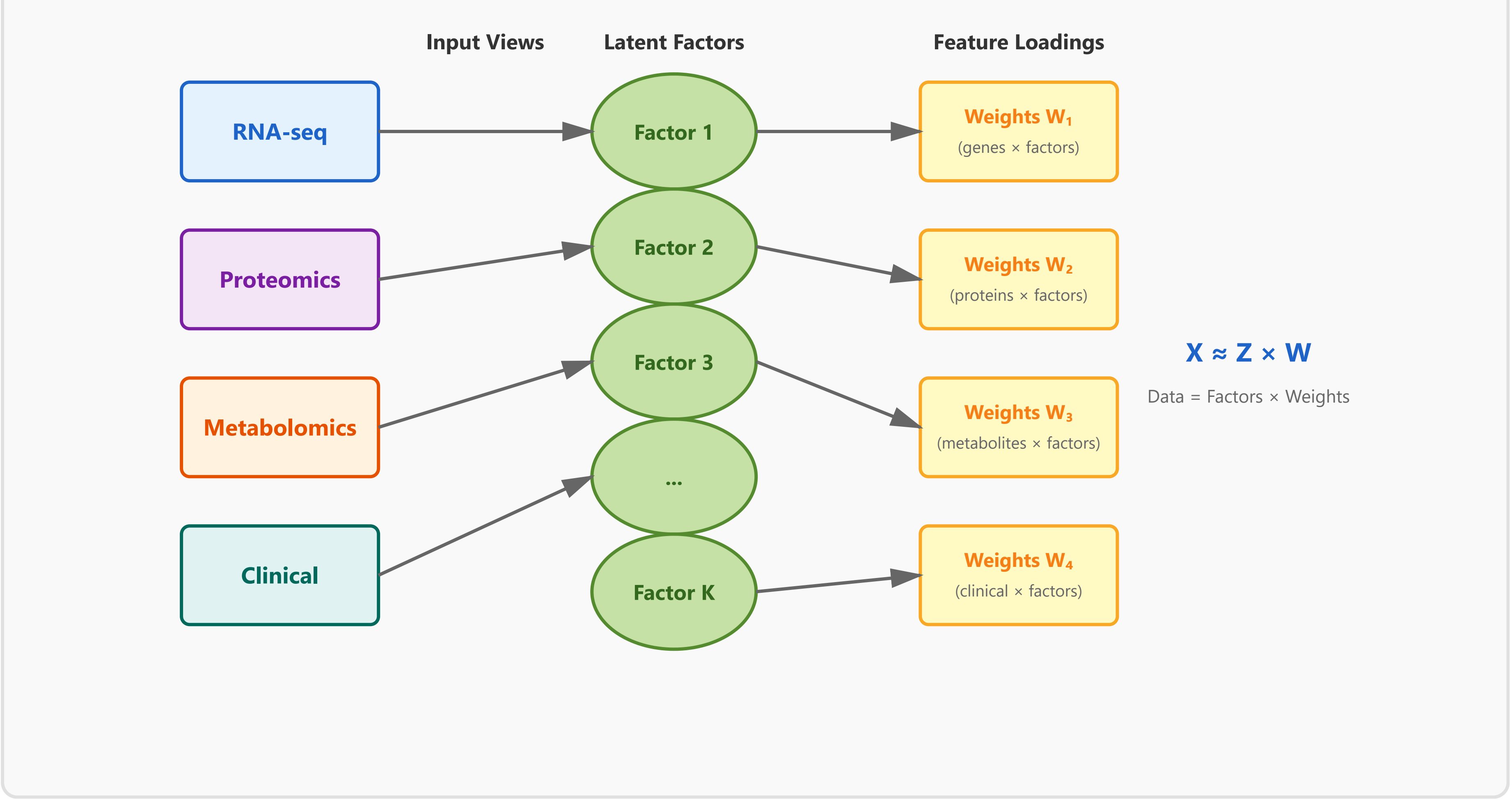
- **Factor initialization:** Start with more factors than expected (MOFA will prune irrelevant ones)

- **Iterative optimization:** Alternating updates of factor values and loadings
- **Convergence criteria:** Stop when the change in ELBO (Evidence Lower BOund) is small
- **Hyperparameter tuning:** Options for sparsity, number of factors, convergence thresholds

Model Parameters:

- **num_factors:** Maximum number of factors (typically 10-25)
- **convergence_mode:** "fast", "medium", or "slow" convergence
- **sparsity:** Degree of feature sparsity in loadings
- **seed:** Random seed for reproducibility

MOFA Model Architecture



```
# Configure and train MOFA model # Set model options model_opts <- get_default_model_options(MOFAobject)
model_opts$num_factors <- 15 # Maximum number of factors model_opts$sparsity <- TRUE # Enable sparse
loadings # Set training options train_opts <- get_default_training_options(MOFAobject)
train_opts$convergence_mode <- "medium" train_opts$seed <- 42 # For reproducibility train_opts$maxiter
<- 1000 # Prepare MOFA object MOFAobject <- prepare_mofa( object = MOFAobject, model_options =
model_opts, training_options = train_opts ) # Run MOFA training MOFAobject <- run_mofa(MOFAobject,
outfile = "model.hdf5") # Training typically takes 5-30 minutes depending on data size
```

💡 Key Points

- Start with more factors than expected (15-25) - MOFA will automatically prune unnecessary ones
- Convergence can take 5-30 minutes for typical datasets (100 samples, 3-5 views)
- Save the trained model to avoid re-training
- Monitor convergence by checking the ELBO plot

3 Factor Interpretation

After training, each latent factor represents a coordinated pattern of variation across omics layers. Interpreting these factors involves examining factor values (scores) for samples and feature weights (loadings) to understand biological meaning.

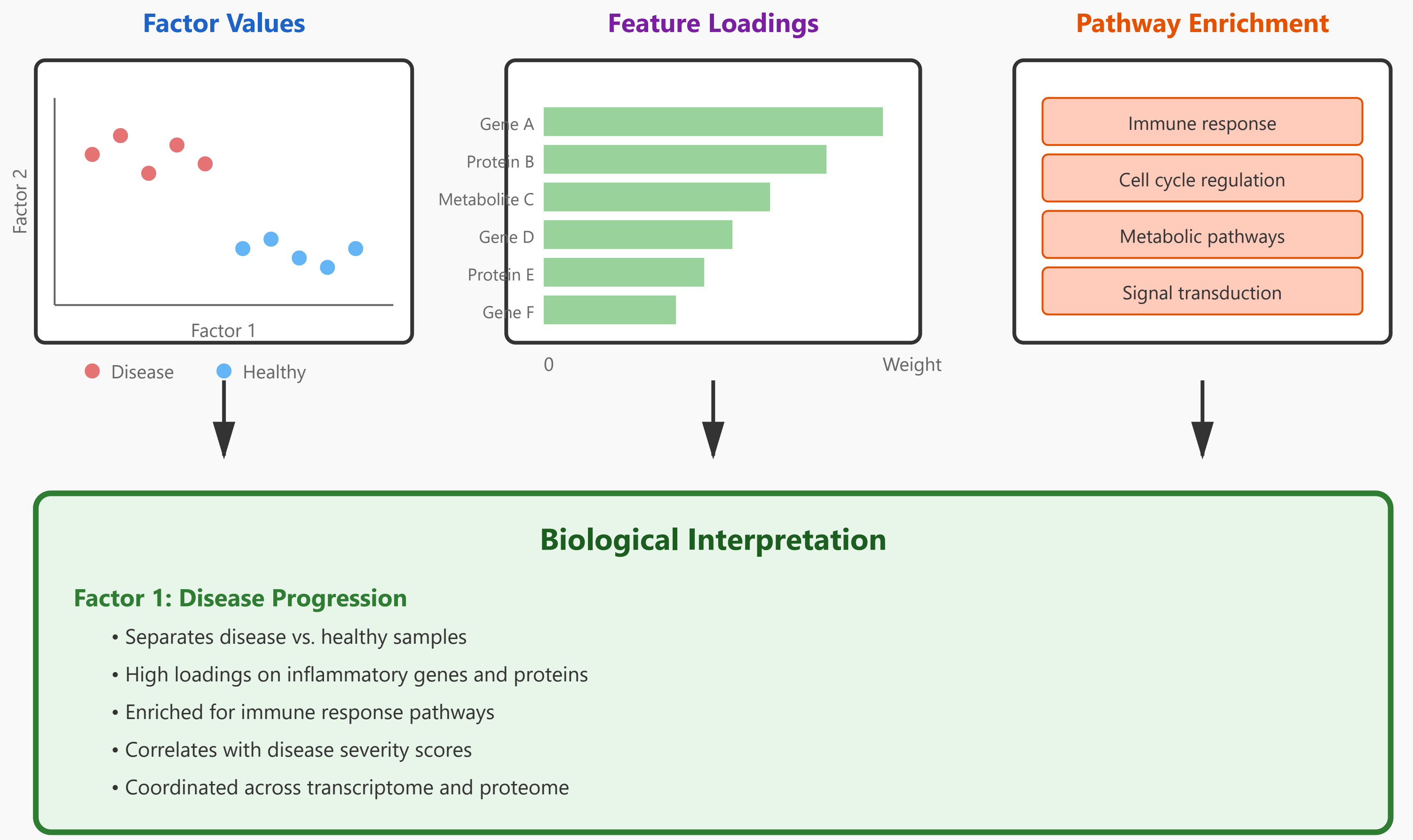
Interpretation Approaches:

- **Factor values:** Examine how samples are distributed along each factor (e.g., by disease state, treatment group)
- **Feature weights:** Identify which genes/proteins/metabolites contribute most to each factor
- **Enrichment analysis:** Test for pathway/GO term enrichment in top weighted features
- **Correlation analysis:** Relate factors to clinical variables or phenotypes
- **View-specific patterns:** Determine which omics layers are most active in each factor

Biological Insights:

- Factors may capture disease subtypes, treatment responses, or biological processes
- Multi-omics factors reveal coordinated regulation across molecular layers
- View-specific factors highlight layer-specific biological variation

Factor Interpretation Workflow



Factor 1: Disease Progression

- Separates disease vs. healthy samples
- High loadings on inflammatory genes and proteins
- Enriched for immune response pathways
- Correlates with disease severity scores
- Coordinated across transcriptome and proteome

```
# Extract and visualize factor values # Get factor values (Z matrix) factors <- get_factors(MOFAobject,
factors = 1:5) # Plot factors colored by metadata plot_factors(MOFAobject, factors = c(1,2), color_by =
"disease_status", shape_by = "treatment") # Get top weighted features weights <- get_weights(MOFAobject,
views = "RNA", factors = 1, abs = TRUE) top_genes <- head(sort(weights, decreasing=TRUE), 20) #
Visualize feature weights plot_weights(MOFAobject, view = "RNA", factor = 1, nfeatures = 20) # Pathway
enrichment analysis library(clusterProfiler) gene_list <- names(top_genes) enrichment <- enrichGO(gene =
gene_list, OrgDb = org.Hs.eg.db, ont = "BP", pAdjustMethod = "BH") # Correlate factors with clinical
variables plot_factor_cor(MOFAobject, factors = 1:10, metadata = c("age", "BMI", "severity_score"))
```

💡 **Key Points**

- Always visualize factor values in the context of sample metadata (disease, treatment, etc.)
- Examine both positive and negative feature loadings for complete interpretation
- Use multiple sources of evidence (enrichment, literature, databases) to validate interpretations
- Not all factors need to have clear biological meaning - some capture technical variation

4 Variance Decomposition

Variance decomposition in MOFA quantifies how much variance in each omics layer is explained by each factor. This analysis reveals which factors are most important overall and which are view-specific or shared across views.

Key Metrics:

- **R^2 per factor per view:** Proportion of variance explained in each omics layer by each factor
- **Total R^2:** Overall variance explained by all factors in each view
- **Shared vs. specific factors:** Identify factors active across multiple views vs. view-specific
- **Factor importance ranking:** Order factors by total variance explained

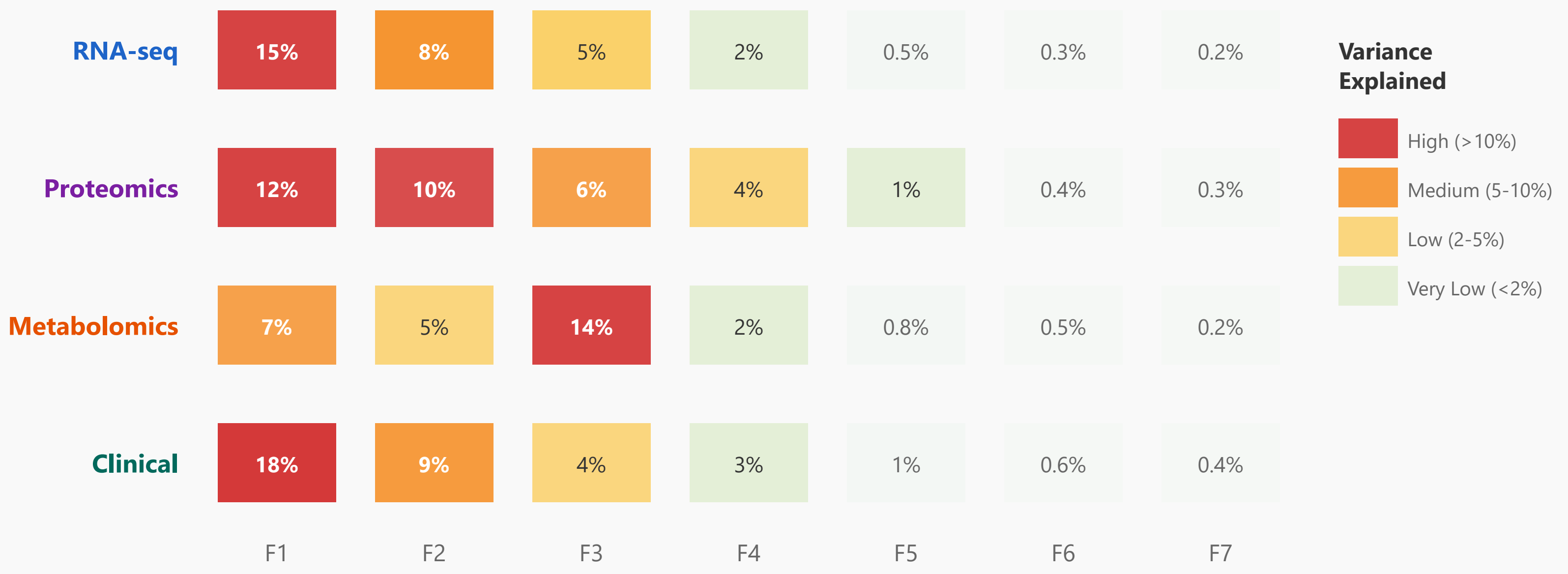
Interpretation Guidelines:

- Factors explaining >5% variance in a view are typically considered important

- Shared factors (active in multiple views) represent coordinated biological processes
- View-specific factors capture layer-specific technical or biological variation
- Low total R^2 may indicate noise, missing data, or need for more factors

Variance Decomposition Heatmap

Variance Explained (R^2) by Each Factor



	F1	F2	F3	F4	F5	F6	F7
RNA-seq	15%	8%	5%	2%	0.5%	0.3%	0.2%
Proteomics	12%	10%	6%	4%	1%	0.4%	0.3%
Metabolomics	7%	5%	14%	2%	0.8%	0.5%	0.2%
Clinical	18%	9%	4%	3%	1%	0.6%	0.4%

Variance Explained

🟥	High (>10%)
🟧	Medium (5-10%)
🟨	Low (2-5%)
🟩	Very Low (<2%)

Key Observations

- is shared: high variance explained across all views (disease-related)
- is metabolomics-specific: captures metabolic variation
- Total variance explained: RNA (31%), Protein (34%), Metabolite (29%), Clinical (36%)

```
# Variance decomposition analysis # Calculate variance explained r2 <-
calculate_variance_explained(MOFAobject) # Plot variance explained by each factor in each view
plot_variance_explained(MOFAobject, x = "view", y = "factor") # Get total variance explained per view
total_r2 <- r2$r2_per_factor %>% group_by(view) %>% summarize(total_variance = sum(value))
print(total_r2) # view total_variance # RNA 0.31 # Proteomics 0.34 # Metabolomics 0.29 # Clinical 0.36 #
Identify factors with >5% variance in any view important_factors <- r2$r2_per_factor %>% filter(value >
0.05) %>% pull(factor) %>% unique() # Plot factor correlation structure plot_factor_cor(MOFAobject) #
Identify shared vs view-specific factors plot_factors_by_groups(MOFAobject, factors = 1:5, color_by =
"disease_status")
```

Key Points

- Focus interpretation on factors explaining >5% variance in at least one view
- Shared factors (high R^2 across multiple views) are most biologically interesting
- View-specific factors may represent technical artifacts or layer-specific biology
- Low total R^2 (<20%) suggests factors may not fully capture data structure

5 Downstream Analysis

The latent factors learned by MOFA serve as powerful features for downstream machine learning and statistical analyses. These compressed representations capture coordinated multi-omics variation and can improve prediction, clustering, and classification tasks.

Common Applications:

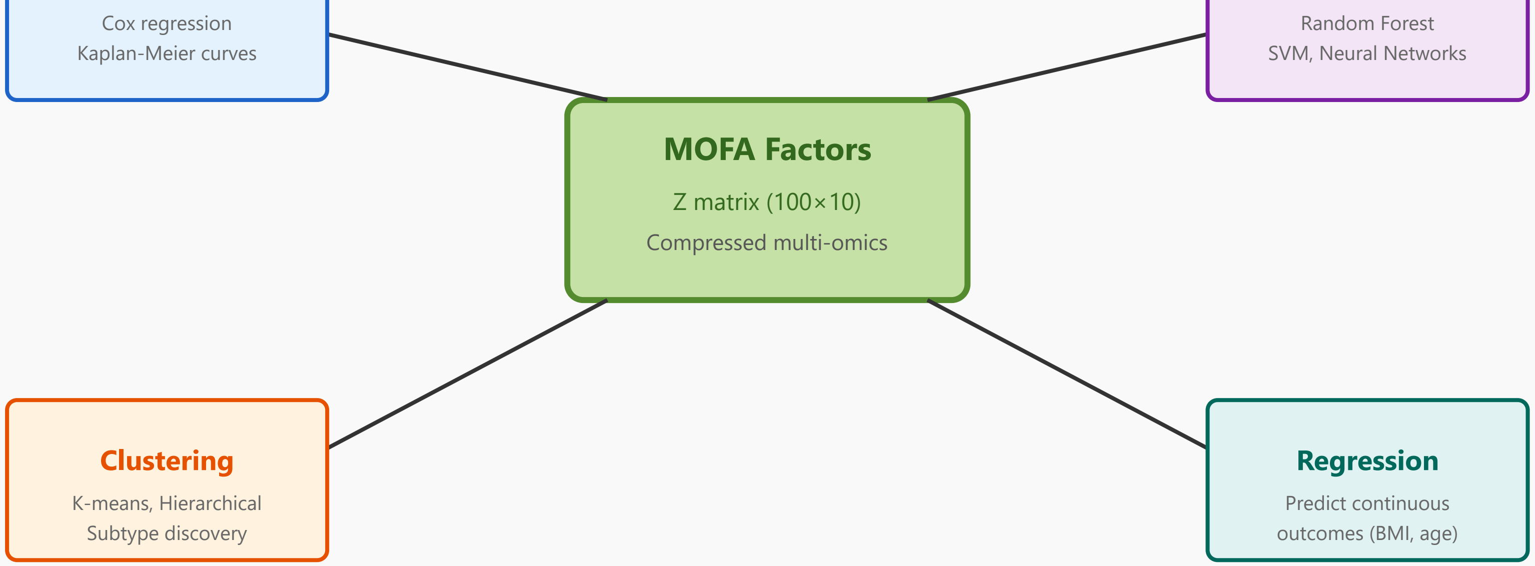
- **Predictive modeling:** Use factors as input features for predicting clinical outcomes (survival, response, disease progression)

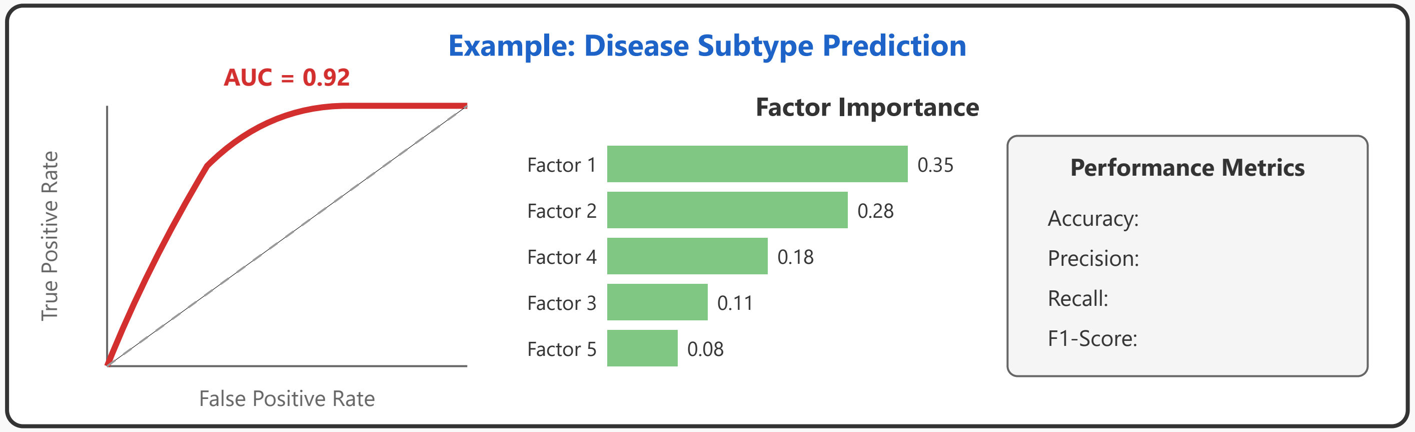
- **Sample clustering:** Group samples based on factor profiles to identify subtypes
- **Dimensionality reduction:** Visualize samples in reduced factor space (PCA-like but multi-omics)
- **Biomarker discovery:** Identify factor-associated features as potential biomarkers
- **Integration with external data:** Relate factors to independent datasets or clinical variables

Advantages of Using MOFA Factors:

- Reduced dimensionality (from thousands of features to 5-15 factors) improves model interpretability
- Factors integrate information across omics layers, capturing coordinated biology
- Less prone to overfitting compared to using raw high-dimensional data
- Biologically meaningful representations aid interpretation of predictions

Downstream Analysis Workflow





```
# Downstream analysis examples # 1. Extract factors as features factors_df <- get_factors(MOFAobject,
as.data.frame = TRUE) metadata <- samples_metadata(MOFAobject) data_for_ml <- cbind(metadata,
factors_df) # 2. Predictive modeling with Random Forest library(randomForest) library(caret) # Split
data set.seed(42) trainIndex <- createDataPartition(data_for_ml$outcome, p=0.7, list=FALSE) train_data
<- data_for_ml[trainIndex, ] test_data <- data_for_ml[-trainIndex, ] # Train model using MOFA factors
rf_model <- randomForest(outcome ~ Factor1 + Factor2 + Factor3 + Factor4 + Factor5, data = train_data,
ntree = 500, importance = TRUE) # Make predictions predictions <- predict(rf_model, test_data)
```

```
confusionMatrix(predictions, test_data$outcome) # 3. Survival analysis library(survival)
library(survminer) surv_data <- data_for_ml %>% select(Factor1, Factor2, Factor3, time, status) # Cox
proportional hazards model cox_model <- coxph(Surv(time, status) ~ Factor1 + Factor2 + Factor3, data =
surv_data) summary(cox_model) # Kaplan-Meier curves stratified by Factor 1 surv_data$Factor1_group <-
ifelse(surv_data$Factor1 > median(surv_data$Factor1), "High", "Low") fit <- survfit(Surv(time, status) ~
Factor1_group, data = surv_data) ggsurvplot(fit, pval = TRUE, risk.table = TRUE) # 4. Clustering for
subtype discovery factors_matrix <- as.matrix(factors_df) kmeans_result <- kmeans(factors_matrix,
centers = 3, nstart = 50) # Visualize clusters plot_factors(MOFAobject, factors = c(1,2), color_by =
kmeans_result$cluster, legend = TRUE) # 5. Correlation with clinical variables clinical_vars <- c("age",
"BMI", "tumor_size", "grade") cor_results <- cor(factors_df, metadata[, clinical_vars], use =
"pairwise.complete.obs") pheatmap(cor_results, cluster_rows = FALSE, cluster_cols = FALSE, color =
colorRampPalette(c("blue", "white", "red"))(100))
```

💡 Key Points

- MOFA factors provide interpretable, low-dimensional features that capture multi-omics variation
- Use cross-validation to assess predictive performance and avoid overfitting
- Factors often outperform single-omics data in prediction tasks due to information integration
- Combine MOFA with domain knowledge and validation in independent cohorts
- Factor-based models are more interpretable than black-box approaches on raw data

Summary

MOFA provides a comprehensive framework for multi-omics integration, from data preparation through downstream analysis. By learning latent factors that capture coordinated variation across molecular layers, MOFA enables biological interpretation, variance decomposition, and improved predictive modeling. The workflow presented here demonstrates the power of integrative approaches in understanding complex biological systems.