Node-centric linear models in spatial-omics

Ian Dirk Fichter
Supervisor: Elyas Heidari

Fischer, DS, Schaar, AC, and Theis, FJ (2022). Modeling intercellular communication in tissues using spatial graphs of cells. Nat Biotechnol, 1–5.

nature biotechnology

BRIEF COMMUNICATION

https://doi.org/10.1038/s41587-022-01467-z

Check for updates

OPEN

Modeling intercellular communication in tissues using spatial graphs of cells

David S. Fischer[1,2,4], Anna C. Schaar[1,3,4] and Fabian J. Theis[1,2,3] ✉

Models of intercellular communication in tissues are based on molecular profiles of dissociated cells, are limited to receptor–ligand signaling and ignore spatial proximity in situ. We present node-centric expression modeling, a method based on graph neural networks that estimates the effects of niche composition on gene expression in an unbiased manner from spatial molecular profiling data. We recover signatures of molecular processes known to underlie cell communication.

GAP

Current cell-cell communication modeling methods:

1. **Do not account for cell-cell communication (niche effects)**

and rely on:

2. **Receptor-ligand interactions**
    - Not suited for protocols with limited R-L expression capture
    - E.g. CellPhoneDB & NicheNet
3. Profiles of dissociated cells
    - No spatial information
4. Leave-one-gene hold-out models which lead to false discoveries
    - E.g. MISTy & SVAE

Spatial models comparison

method	constrained by niches observed in spatial data	is predictive	models cell-cell communication events in space	works on targeted protocols without capture of cognate ligands and receptors	does not use potentially limited leave-one-gene-out cross-validation
NCEM	yes	yes	yes	yes	yes
SpaOTsc[1]	yes		yes		yes
SVCA[2]	yes	yes	yes	yes	
Garcia-Alonso *et al.*[3]	yes		yes		yes
GNCG[4]	yes				yes
MISTy[5]	yes	yes	yes	yes	
stLearn[6]	yes		yes	yes	
Neighborhood enrichment: Giotto[7] squidpy[8]	yes			yes	yes

NCEM

- Model cell neighbourhood (niche) effects

- Recover molecular signatures of processes underlying cell-cell communication

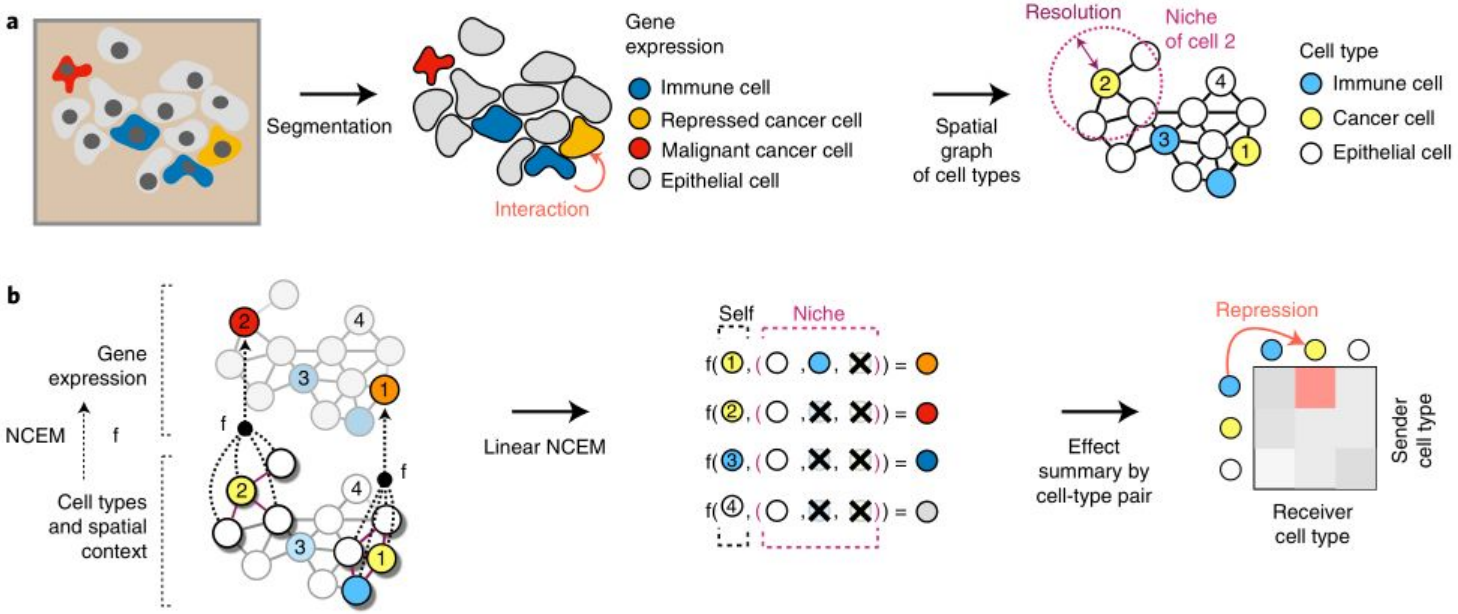
- Account for spatial dimension

- Linear and non-linear models

- Extensible to cellular resolution (Cell2Location)

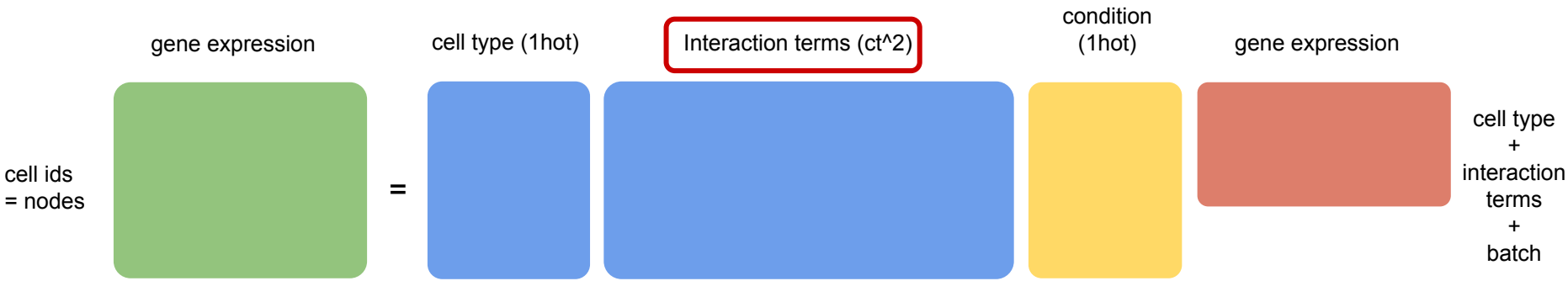
Flagship: **linear NCEM model**

Predicts gene expression based on index cell type and niche composition



Linear model design $\hat{Y} = X^D \beta$

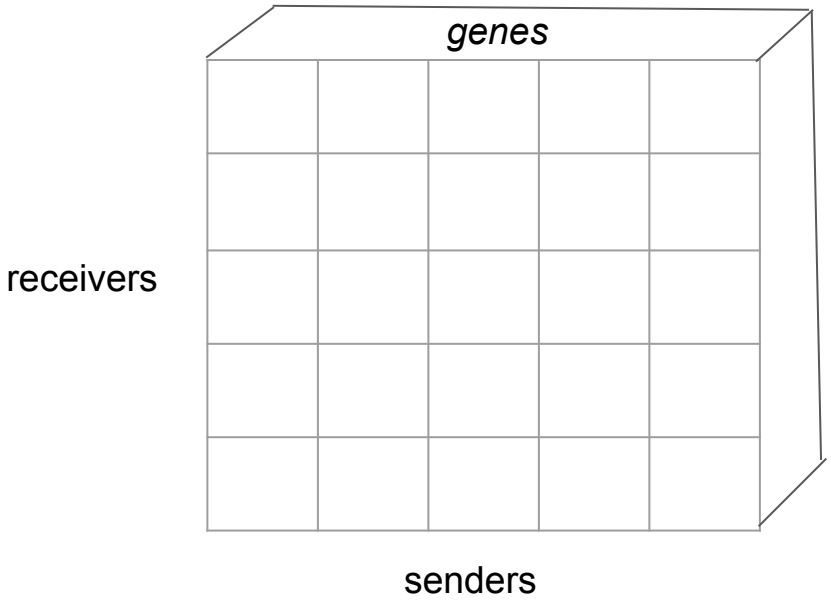
$$X^D = (X^l, X^{TS}, X^c) \in R^{N \times (L+L^2+C)} \qquad \beta \in R^{(L+L^2+C) \times J}$$



Linear model: output of interest

Interaction terms:

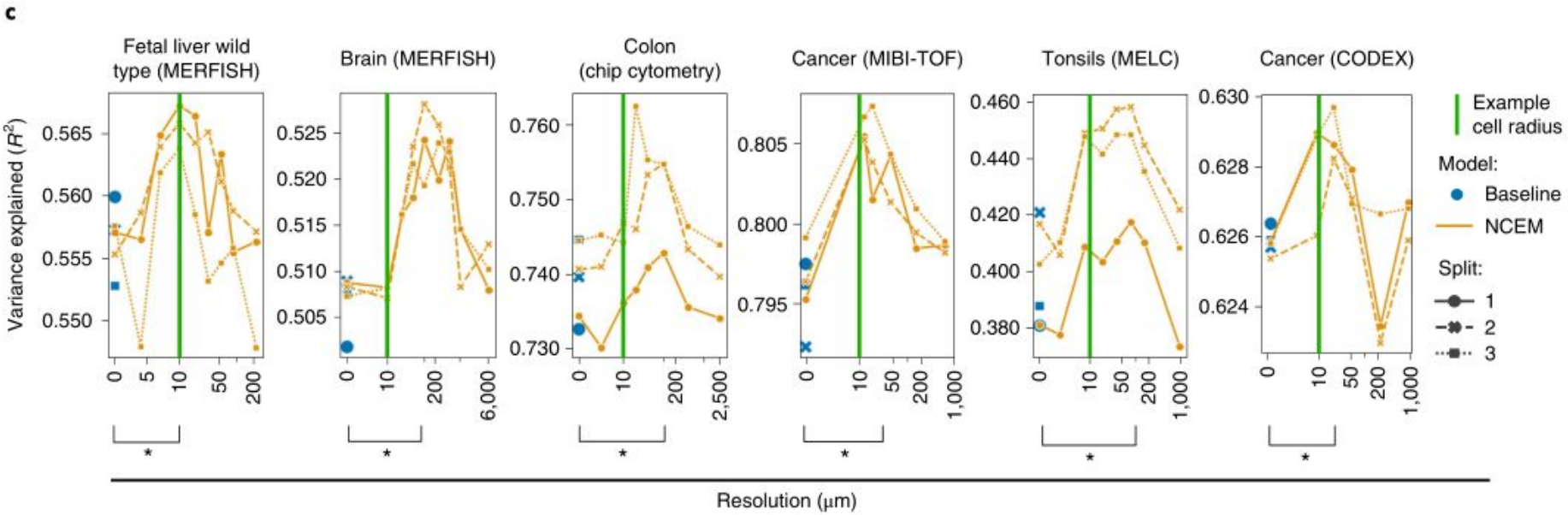
- Matrix: sender cell-type x receiver cell-type x genes



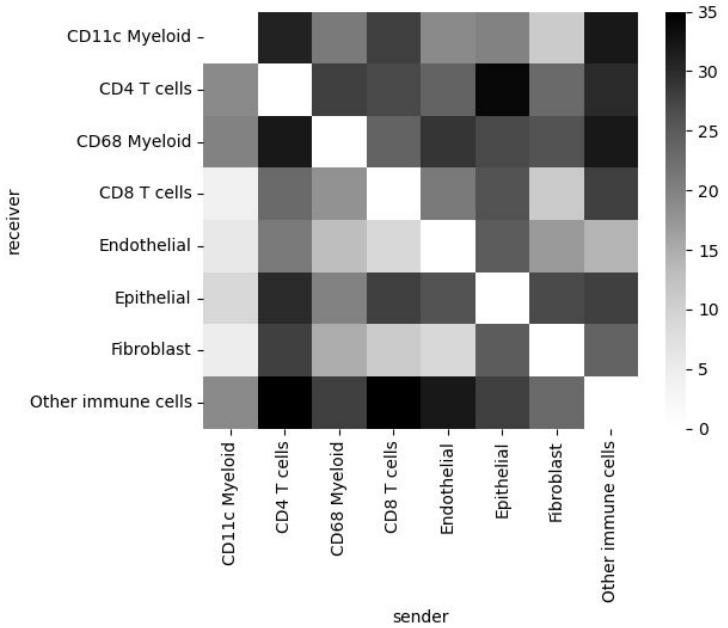
Linear NCEMs

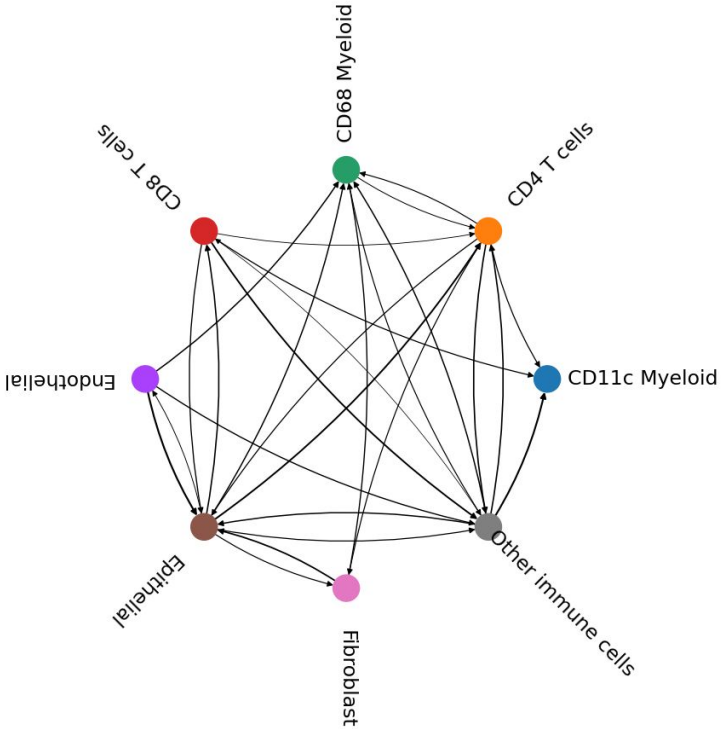
- Infer and recover cell-cell communication signatures
- Outperformed nonspatial baseline models consistently by an average ΔR2 of 0.016
- Recover L-R interactions from CellPhoneDB and NicheNet
- Robust to:
    - data downsampling
    - out-of-domain data from an unseen genetic knockout condition
    - to simulated segmentation errors
    - removal of the interaction terms from the linear model

Tissue and technology variation

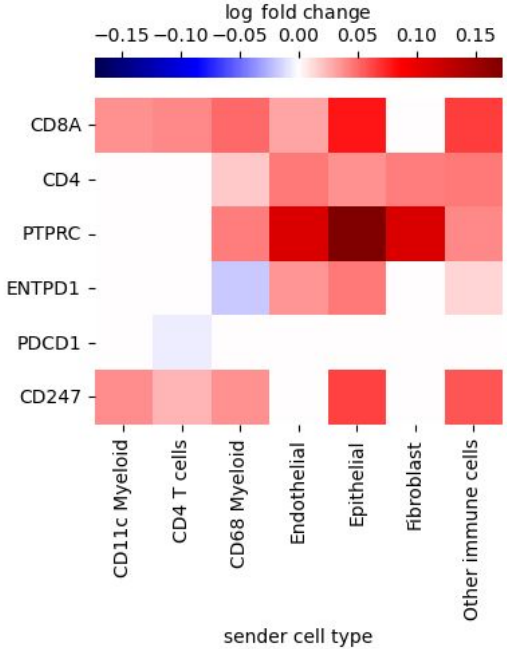


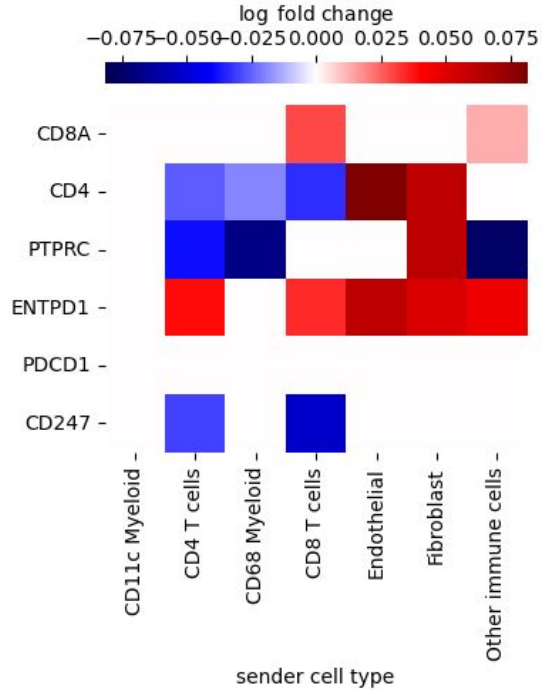
Type-coupling analysis

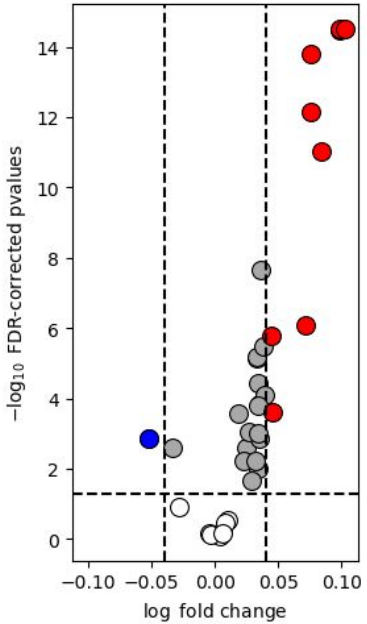




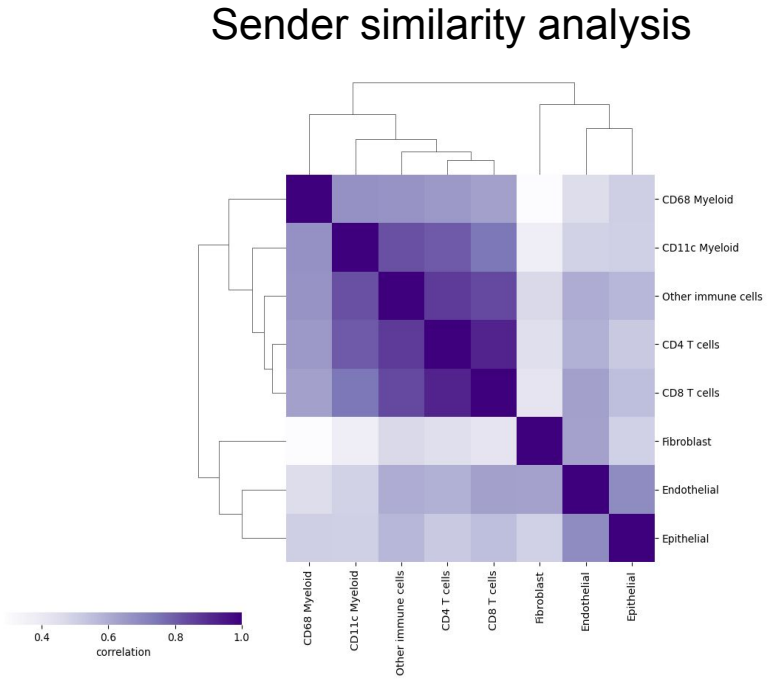
Sender/receiver effects on CD8 T-cells





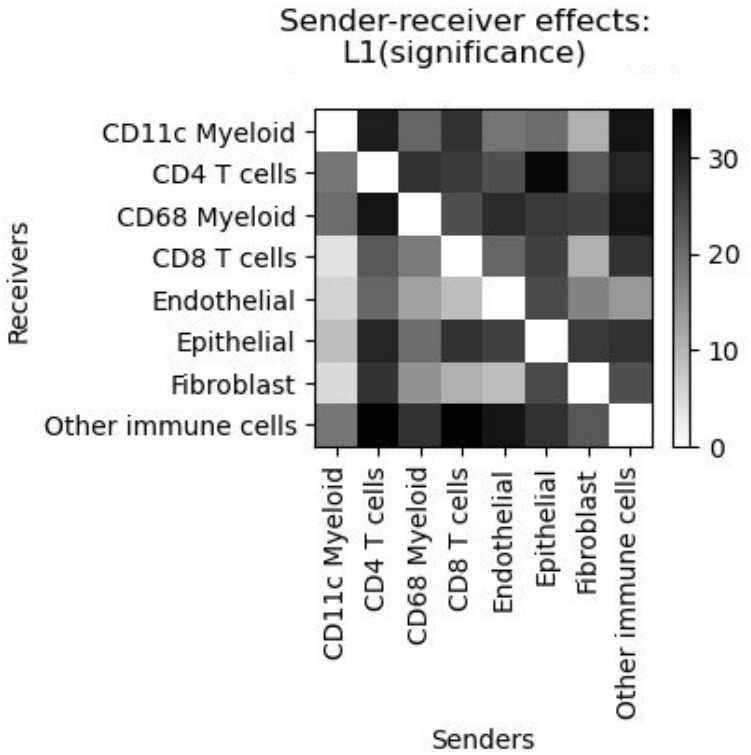


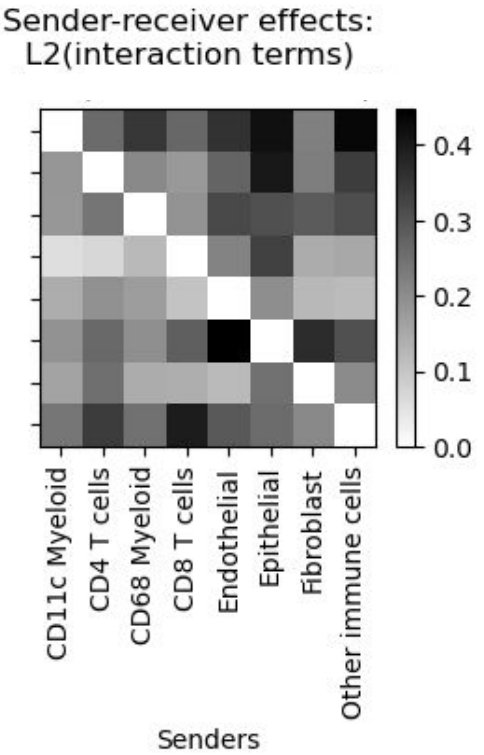
Sender profile identity preservation



Wrapper output: interaction terms

```
# Applying the function
ncem_custom_graph = custom_effect_graph(
    data_loader='hartmann',
    data_path='input-data/raw-data/Hartmann-2021/',
    radius=35,
    n_eval_nodes=10,
    qval_thresh=0.05,
    l1_thresh=0,
    fc_thresh=0,
    l2_thresh=0
)
```





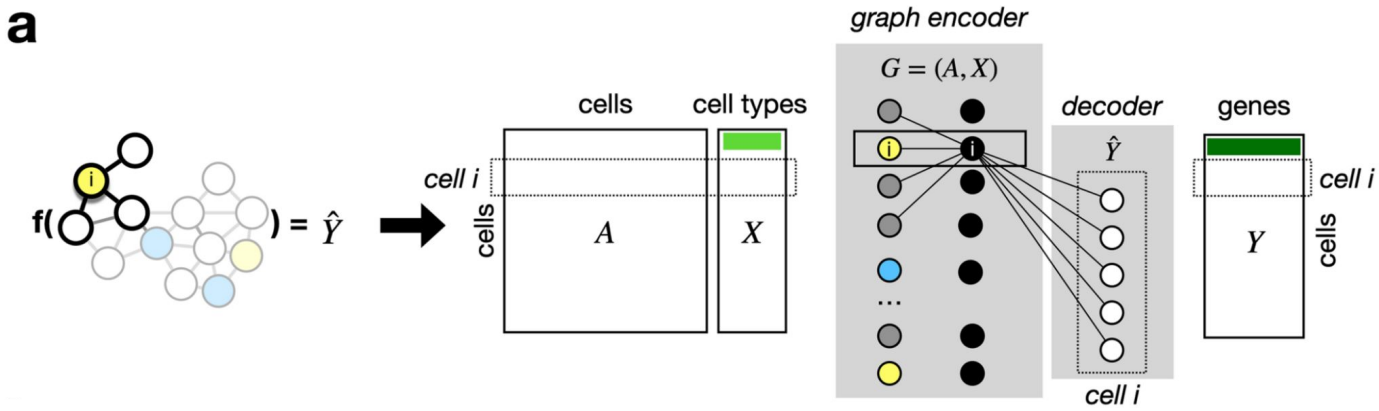
Non-linear NCEM models

Encode expression vectors

- Graph neural network
- Conditional variational autoencoder (CVAE)
- Ligand–receptor autoencoder with L-R latent space

method	models source-target interactions	models niche motives of higher order (>2 participating cells)	models intrinsic variation alongside extrinsic effects from niche	feature space for interaction modeling
linear NCEM	yes	no	no	categorical cell types
nonlinear NCEM	yes	yes	no	categorical cell types
nonlinear NCEM-LR	yes	no	no	ligand and receptor gene expression
CVAE-NCEM	yes	yes	yes	categorical cell types

GNN

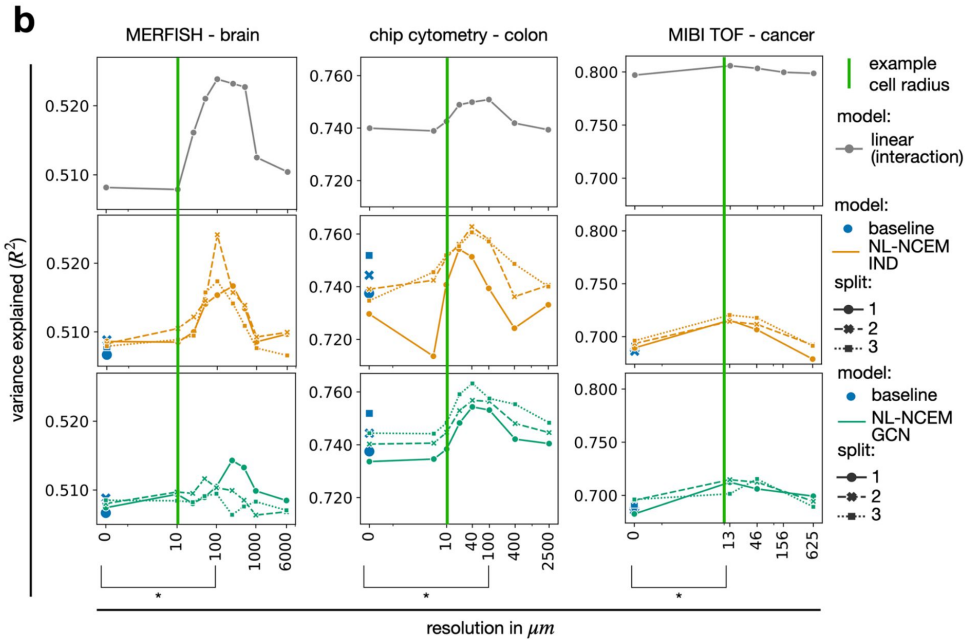


local graph embedding:

$$f_{\text{enc}} : q_\phi(z_i | X_i^l, g(A, X^1)_i, X_i^c)$$

GNN

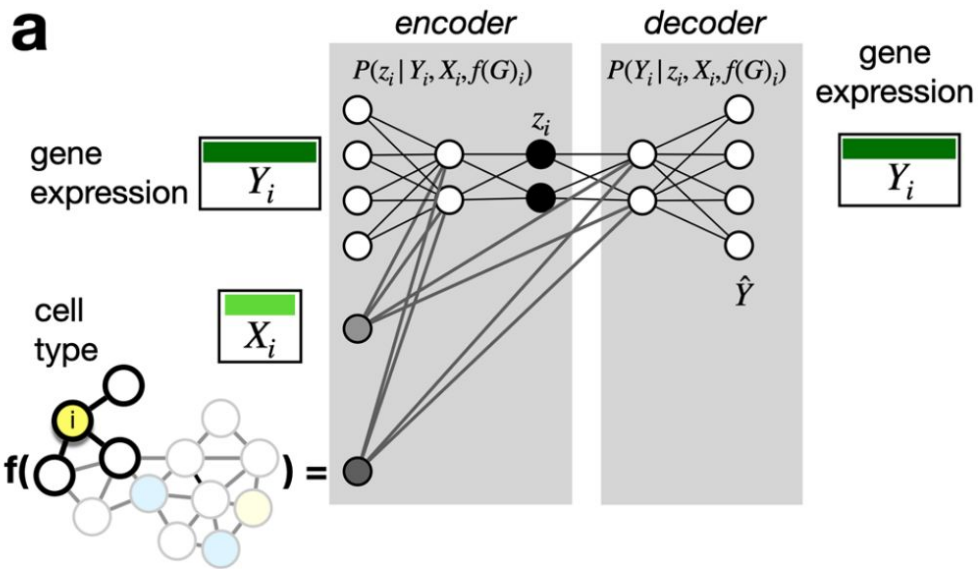
- Did not outperform linear models in gene expression prediction



CVAE

Cell type and neighbours predictors as a condition in the variational posterior and the likelihood model

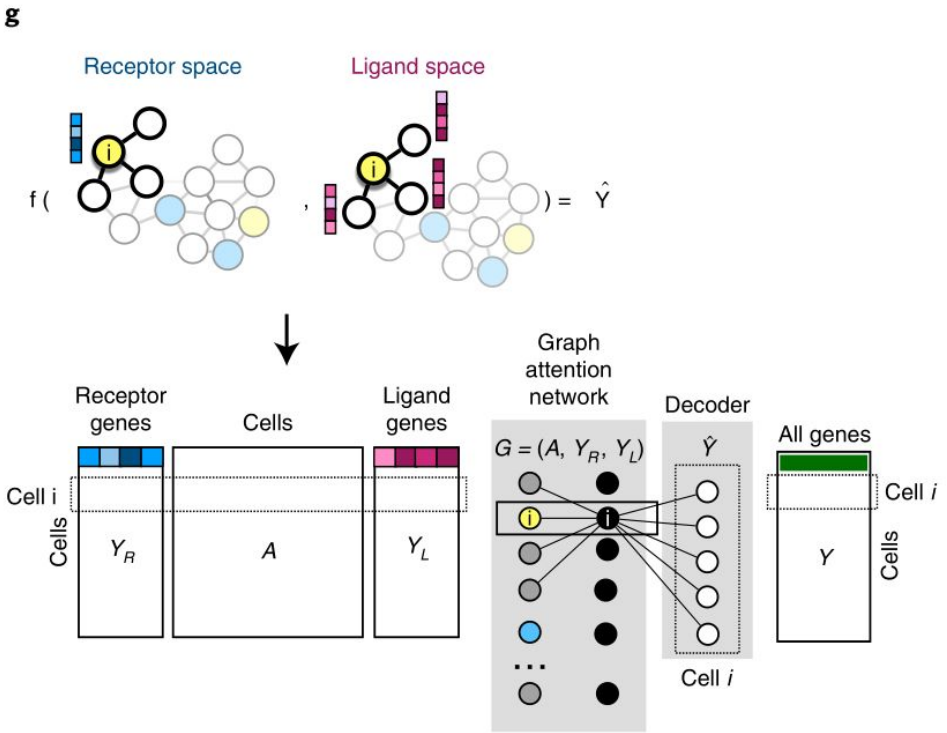
$$f_{\text{enc}} : q_\phi(z_i | Y_i, X_i^{\text{l}}, g(A, X^{\text{l}})_i, X_i^{\text{c}})$$



R-L kernel attention graph

- Receptor-dimensional latent space (z)
- Higher predictive performance than CVAE

$$z_{ik} = g(A_i, Y_{i,r(k)}, Y_{:,l(k)})$$



Outlook

Method specific:

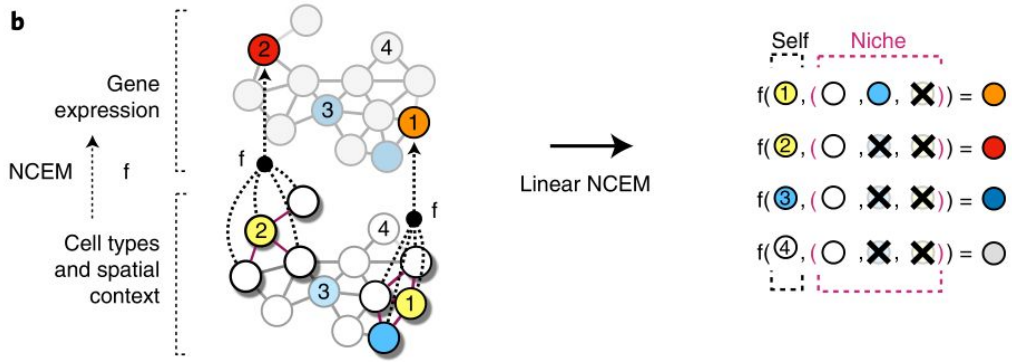
- Improve niche heterogeneity capture via:
  - Segmentation level improvement
  - 3D data
  - Niche perturbation
- L-R graph kernel networks explain extrinsic variation and could be used for further modelling using cell intrinsic variables

Personal

- Benchmark interaction matrices against other spatial interaction methods e.g. MISTy or DIALOGUE

THM

- NCEMs are graph-based expression models that model niche effects
- Linear NCEMs are the most predictive and recover cell-cell communication profiles
- Overcome non-spatial and L-R-based models' limitations



NCEM model training and testing workflow

1. Define parameters
2. Initiate model
3. Train
4. Evaluation
    - Optionally by cell-type

Defining parameters

Dataset specific inputs

```
data_set = 'hartmann'
data_path = data_path_base + '/Hartmann-2021/'
log_transform = False
use_domain = True
scale_node_size=False
merge_node_types_predefined = True
covar_selection = []
output_layer='gaussian'
```

Manual inputs

```
model_class = 'cvae'
optimizer = 'adam'
domain_type = 'patient'

learning_rate = 0.05
l1 = 0.
l2 = 0.

batch_size = 58
radius = 10
n_eval_nodes = 10

gs_id = f"tutorial_{model_class}_{radius}_{data_set}_{domain_type}"
```

Model initiation

```
trainer.estimator.split_data_node(
    validation_split=0.1,
    test_split=0.1,
    seed=0
)
```

```
Using split method: node.
 Train-test-validation split is based on total number of nodes per patients c

Excluded 0 cells with the following unannotated cell type: [None]

Whole dataset: 63747 cells out of 58 images from 4 patients.
Test dataset: 6376 cells out of 58 images from 4 patients.
Training dataset: 51930 cells out of 58 images from 4 patients.
Validation dataset: 5738 cells out of 58 images from 4 patients.
```

```
trainer.estimator.init_model(
    optimizer=optimizer,
    learning_rate=learning_rate,
    n_eval_nodes_per_graph=n_eval_nodes,

    l1_coef=l1,
    l2_coef=l2,
    use_domain=use_domain,
    use_batch_norm=False,
    scale_node_size=scale_node_size,
    output_layer=output_layer,

    latent_dim=10,
    dropout_rate=0.1,
    intermediate_dim_enc=128,
    intermediate_dim_dec=128,
    depth_enc=1,
    depth_dec=1,
)
trainer.estimator.model.training_model.summary()
```

Model training

```
trainer.estimator.train(
    epochs=epochs,
    epochs_warmup=epochs_warmup, # Integer number of times to iterate over the training data arrays in warm up (without early stopping)
    max_steps_per_epoch=max_steps_per_epoch,
    batch_size=batch_size,
    validation_batch_size=val_bs,
    max_validation_steps=max_val_steps_per_epoch,
    patience=patience, # Number of epochs with no improvement.
    lr_schedule_min_lr=lr_schedule_min_lr,
    lr_schedule_factor=lr_schedule_factor,
    lr_schedule_patience=lr_schedule_patience,
    monitor_partition="val",
    monitor_metric="loss",
    shuffle_buffer_size=shuffle_buffer_size,
    early_stopping=True,
    reduce_lr_plateau=True,
    decoder_epochs=decoder_epochs,
    decoder_patience=decoder_patience
)
```

Model evaluation

```
evaluation_test = trainer.estimator.evaluate_any(
    img_keys=trainer.estimator.img_keys_test,
    node_idx=trainer.estimator.nodes_idx_test
)
```

```
split_per_node_type, evaluation_per_node_type = trainer.estimator.evaluate_per_node_type()
```

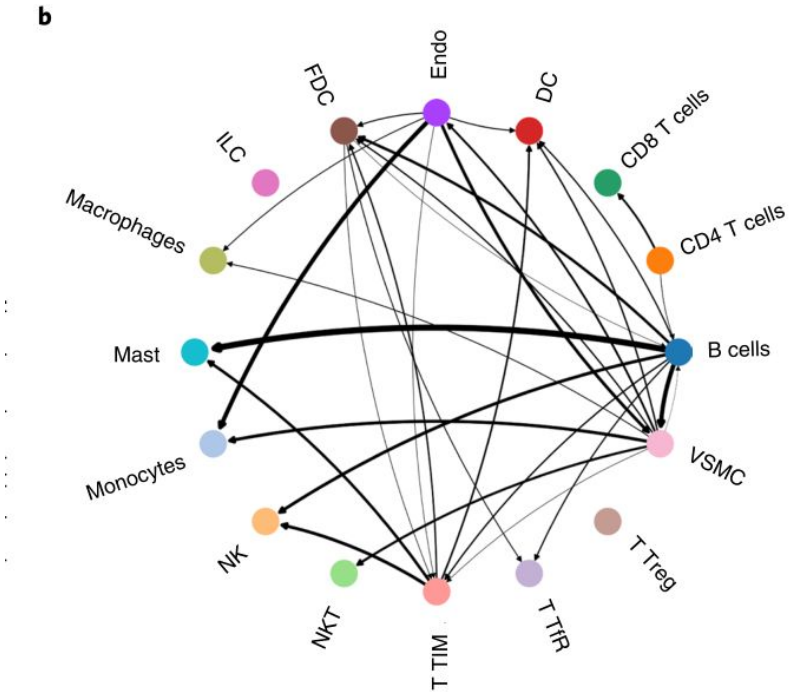
```
evaluation_per_node_type['Fibroblast']
```


```
{'loss': 48.6131477355957,
 'reconstruction_loss': 47.417274475097656,
 'bottleneck_loss': 1.1958887577056885,
 'reconstruction_custom_mae': 0.5758668184280396,
 'reconstruction_custom_mean_sd': 1.180565357208252,
 'reconstruction_custom_mse': 0.5665685534477234,
 'reconstruction_custom_mse_scaled': 0.4724542498588562,
 'reconstruction_gaussian_reconstruction_loss': 47.41725540161133,
 'reconstruction_r_squared': -6.493439197540283,
 'reconstruction_r_squared_linreg': 0.03642991557717323,
 'bottleneck_custom_kl': 119.58879852294922,
 'elbo': 59.3761389160156}
```

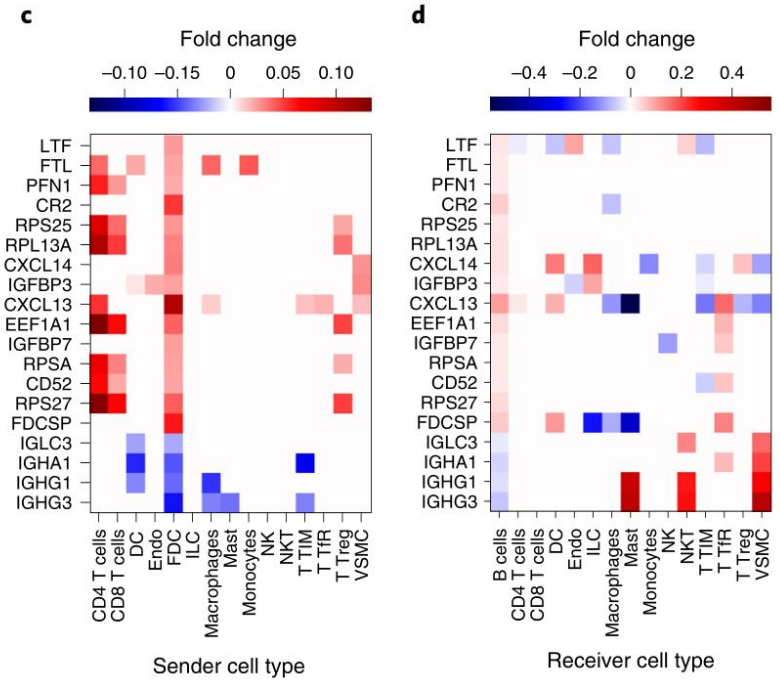
Backup slides

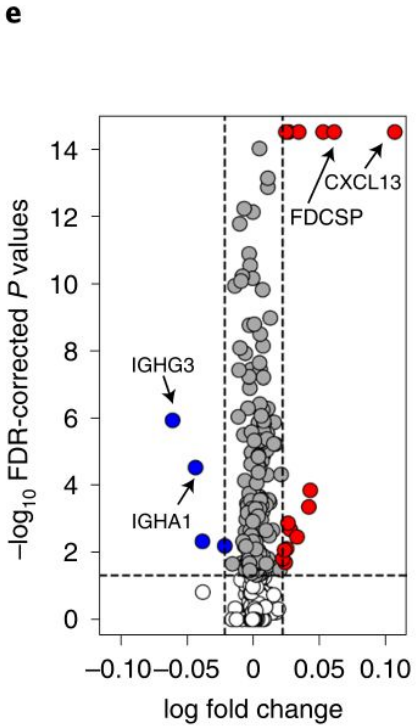
Type-coupling analysis

- Couplings with germinal centers



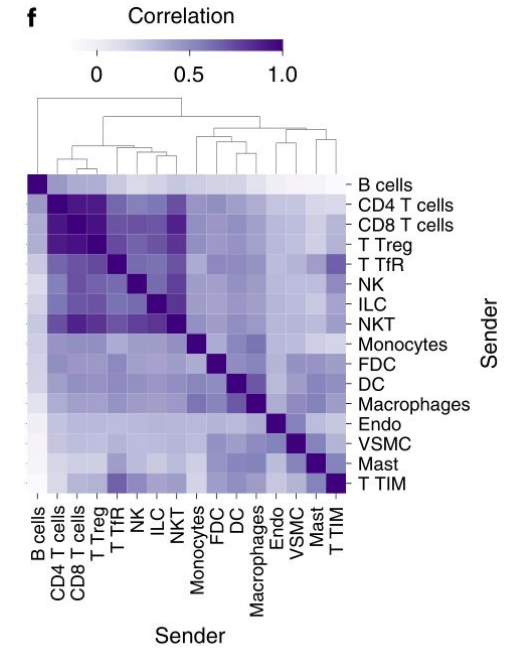
Sender/Receiver effects on FDC-B cell axis



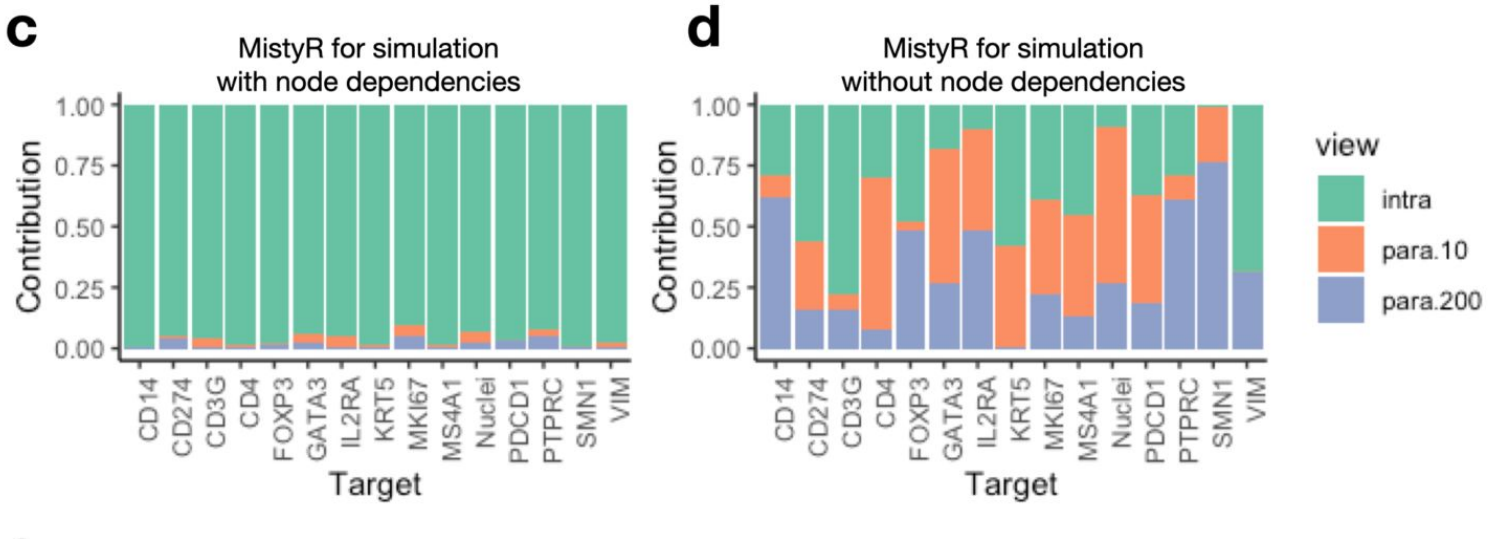


Sender profile identity preservation

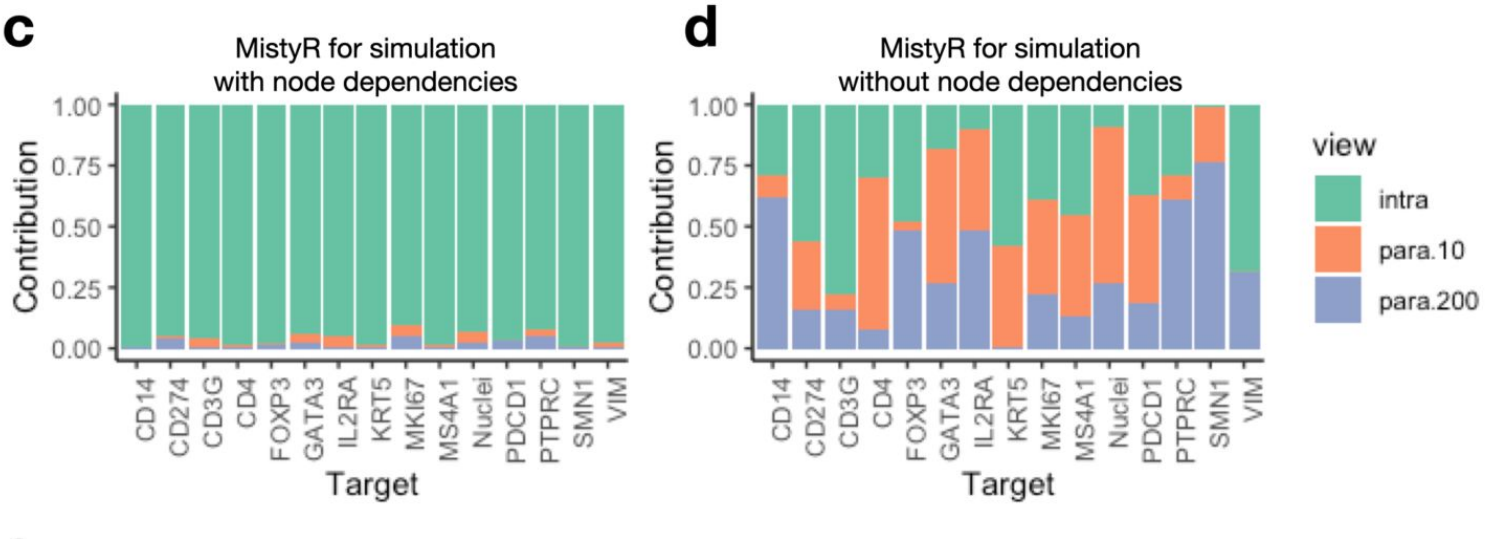
Sender similarity analysis



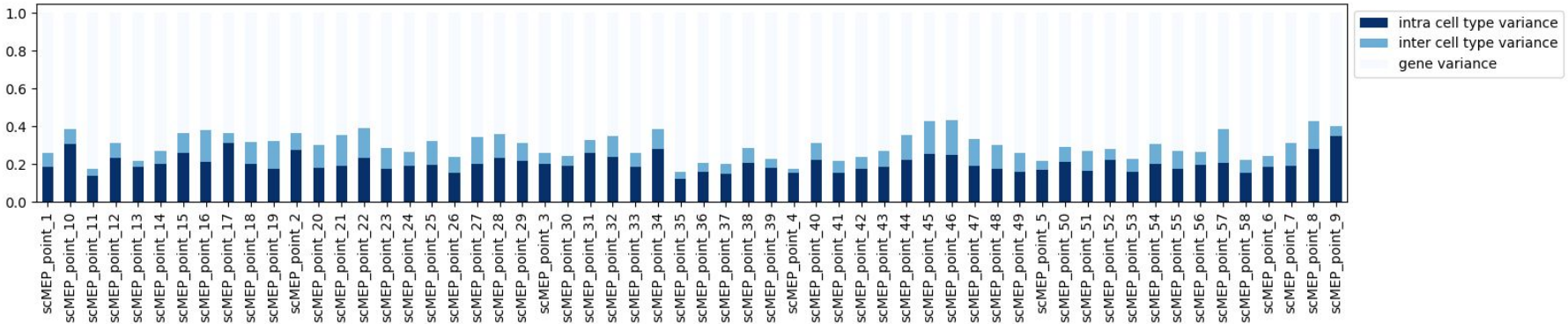
Leave-one-gene out: MISTy



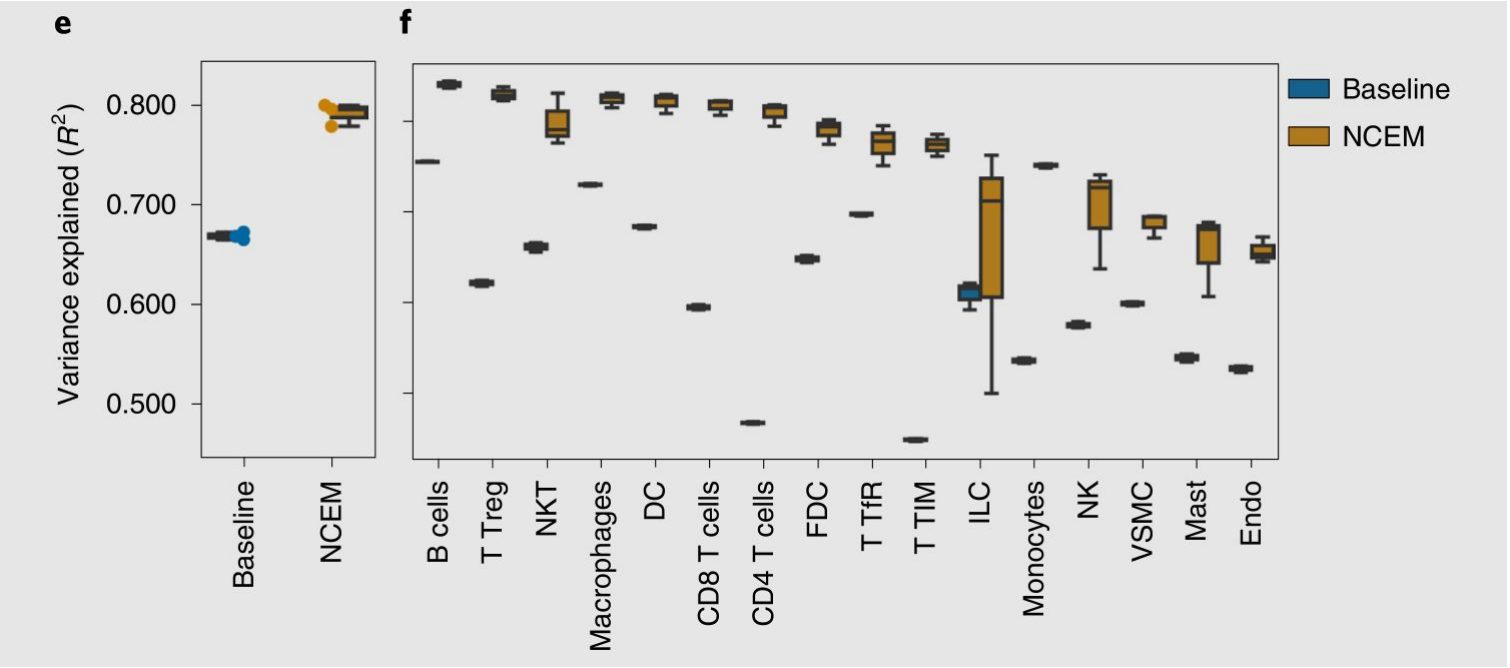
Leave-one-gene out: SVCA



Variance decomposition



Linear models outperform baseline non-spatial models in spot-based technologies



Sender-receiver effects workflow

1. Create AnnData object
2. Input directory and file structure blueprint based on DataLoader class
3. Call custom function

```
# Custom wrapper function
def custom_sender_receiver_effect(
    data_loader: str=None,
    data_path: str=None,
    radius: int=35,
    n_eval_nodes: int=10,
    qval_thresh: float=0.05,
    l1_thresh: int=0,
    fc_thresh: tuple=[0.00, 0.00],
    l2_thresh = 0,
    plot: bool=True
    ):
    '''
    Calculates a custom sender-receiver graph based on interpreter.sender_effect()
     affected gene numbers per sender-receiver pair.

    Prerequisites:
     - import ncem
     - know what data loader you want to use
     - spatial data directories and files structured after the data loaders structure

    Input
     - Spatial features data
     - Parameters:
        - qval_thresh: FDR corrected p-value to be used as an upper-bound significance threshold
          (both L1 and L2 arrays)
        - l1_thresh: Minimum number of significant genes threshold for the L1 array.
        - fc_thresh:
            - Option 1: a list of two int or float numbers to be used as an upper-bound (first number)
              and lower-bound (second number)
              thresholds for the original coefficient values (only applied to L2 array).
            - Option 2: an int or float number. Assuming a normal distribution of coefficient values,
              the number indicates the standard deviation to be used as a cutoff. Only values above and
              below the mean +/- the standard deviation will be included (only on L2 norm array).
        - l2_thresh:
            - Option 1: Lower-bound threshold for computed L2 norm values (L2 array only).
            - Option 2: Use a specific percentile as a lower-bound threshold for computed
              L2 norm values (L2 array only).
              Notation str('AXX'), where 'A' is short for 'Above' and the 'XX' are strictly
              two numbers (int) to be considered as the lower-bound percentile threshold.

    Output:
     - List of 2 numpy arrays:
        - 1. Type-coupling (Sender-receiver effect) interaction matrix
             L1 norm of the number of significant interaction terms per cell
        - 2. Sender-receiver magnitude effect interaction matrix
             L2 norm of interacton term coefficients of genes per cell
        - dim(cell-types x cell-types)
            - [receiver, sender]
    '''
```

Data loaders

- Hartmann2021 MIBI-TOF - colorectal carcinoma & healthy colon.
- Zhang2020 MERFISH - Brain
- PascualReguant2021 MELC - Tonsils
- Schürch2020 - Colorectal Cancer

- Jarosch
- Lohoff
- 10xVisiumMouseBrain
- 10xLymphnode

Data loader Hartmann

```python
class DataLoaderHartmann(DataLoader):
    """DataLoaderHartmann class. Inherits all functions from DataLoader."""

    cell_type_merge_dict = {
        "Imm_other": "Other immune cells",
        "Epithelial": "Epithelial",
        "Tcell_CD4": "CD4 T cells",
        "Myeloid_CD68": "CD68 Myeloid",
        "Fibroblast": "Fibroblast",
        "Tcell_CD8": "CD8 T cells",
        "Endothelial": "Endothelial",
        "Myeloid_CD11c": "CD11c Myeloid",
    }

    def _register_celldata(self, n_top_genes: Optional[int] = None):
        """Load AnnData object of complete dataset."""
        metadata = {
            "lateral_resolution": 400 / 1024,
            "fn": ["scMEP_MIBI_singlecell/scMEP_MIBI_singlecell.csv", "scMEP_sample_description.xlsx"],
            "image_col": "point",
            "pos_cols": ["center_colcoord", "center_rowcoord"],
            "cluster_col": "Cluster",
            "cluster_col_preprocessed": "Cluster_preprocessed",
            "patient_col": "donor",
        }
        celldata_df = read_csv(os.path.join(self.data_path, metadata["fn"][0]))
        celldata_df["point"] = [f"scMEP_point_{str(x)}" for x in celldata_df["point"]]
        celldata_df = celldata_df.fillna(0)
        # celldata_df = celldata_df.dropna(inplace=False).reset_index()
        feature_cols = [
            "H3",
            "vimentin",
            "SMA",
            "CD98",
            "NRF2p",
            "CD4",
            "CD14",
```

Type-coupling analysis

	sender	receiver	0	magnitude	de_genes	de_genes_abs
1	CD11c Myeloid	CD4 T cells	19	0.183663	0.483871	19
2	CD11c Myeloid	CD68 Myeloid	20	0.182613	0.516129	20
3	CD11c Myeloid	CD8 T cells	4	0.059416	0.000000	4
4	CD11c Myeloid	Endothelial	6	0.143681	0.064516	6
5	CD11c Myeloid	Epithelial	9	0.191141	0.161290	9
6	CD11c Myeloid	Fibroblast	5	0.159925	0.032258	5
7	CD11c Myeloid	Other immune cells	19	0.240403	0.483871	19
8	CD4 T cells	CD11c Myeloid	31	0.259384	0.870968	31
10	CD4 T cells	CD68 Myeloid	32	0.242240	0.903226	32
11	CD4 T cells	CD8 T cells	23	0.070209	0.612903	23
12	CD4 T cells	Endothelial	21	0.192594	0.548387	21
13	CD4 T cells	Epithelial	30	0.260940	0.838710	30
14	CD4 T cells	Fibroblast	28	0.251974	0.774194	28
15	CD4 T cells	Other immune cells	35	0.342568	1.000000	35
16	CD68 Myeloid	CD11c Myeloid	21	0.349795	0.548387	21
17	CD68 Myeloid	CD4 T cells	28	0.207439	0.774194	28
19	CD68 Myeloid	CD8 T cells	18	0.123444	0.451613	18
20	CD68 Myeloid	Endothelial	13	0.171432	0.290323	13
21	CD68 Myeloid	Epithelial	20	0.196243	0.516129	20
22	CD68 Myeloid	Fibroblast	15	0.146196	0.354839	15
23	CD68 Myeloid	Other immune cells	28	0.246856	0.774194	28
24	CD8 T cells	CD11c Myeloid	28	0.265812	0.774194	28

Squidpy: neighbourhood enrichment

Graph and spatial patterns analysis. *Neighborhood enrichment test.* The association between label pairs in the connectivity graph is estimated by counting the sum of nodes that belong to classes i and j (for example cluster annotation) and are proximal to each other, noted x_{ij}. To estimate the deviation of this number versus a random configuration of cluster labels in the same connectivity graph, we scramble the cluster labels while maintaining the connectivities and then recount the number of nodes recovered in each iteration (1,000 times by default). Using these estimates, we calculate expected means (μ_{ij}) and standard deviations (σ_{ij}) for each pair and a z score as,

$$Z_{ij} = \left(x_{ij} - \mu_{ij}\right) / \sigma_{ij}$$

The z score indicates if a cluster pair is over-represented or over-depleted for node–node interactions in the connectivity graph. This approach was described by Schapiro et al.[54]. The analysis and visualization can be performed with the analysis code shown below.