

Normalizing Flows for Conditional Generation of Synthetic Transcriptomic Samples

Antonio Guillén Teruel

Focus Period Halmstad University 2026

Why do we need synthetic transcriptomic data?

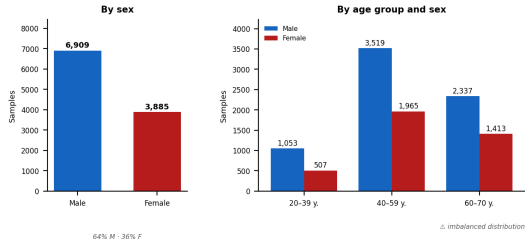
The problem

- Studies are **expensive and slow**
- Scarce groups: rare tissues, advanced ages
- Clinical imbalance
- ML models need **large, balanced volumes of data**

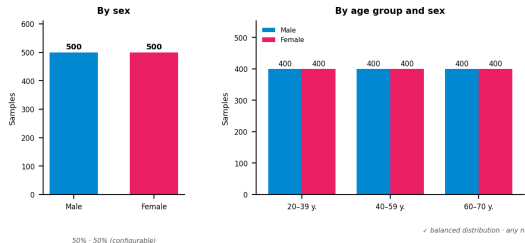
Synthetic data enable

- Data augmentation
- Simulation of virtual experiments
- Protecting donor privacy
- **Training predictive models** on “difficult” cohorts

Real data · GTEx v11



Synthetic data · simulated



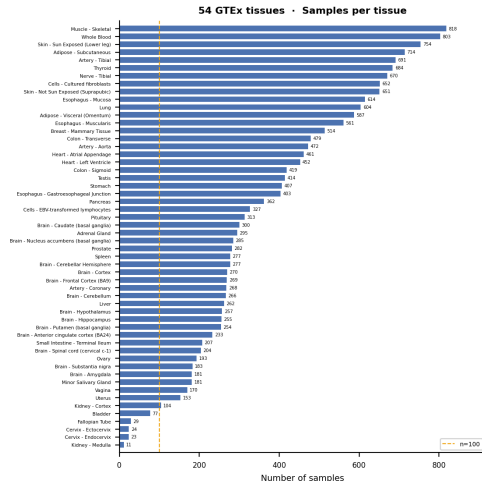
The dataset: GTEx

Genotype-Tissue Expression (GTEx v11)

- **19,616** samples of *bulk* RNA-seq
- **946** unique donors (363M, 190F)
- **54** different tissues
- **10,000** genes (selected HVGs)

Available covariates

Covariate	Detail
Sex	M: 6,909 / F: 3,885
Age	20 – 70 years
Tissue	54 classes
With sex <i>and</i> age	~10,794 samples



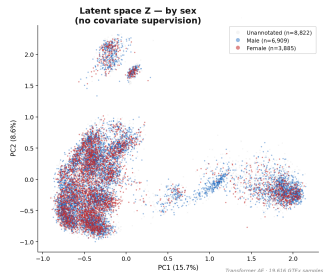
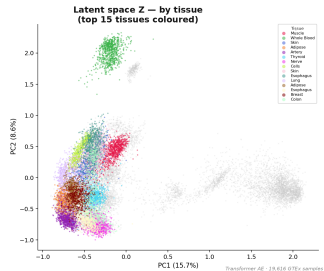
The challenge: a complex gene expression distribution

The high-dimensionality problem

- Each sample = a **10,000-dimensional** vector
- It differs across tissues, ages and sexes
- Modelling $p(X)$ directly in 10k dims: infeasible

Why not simply resample what we observed?

- It generates no new variability
- It does not allow conditioning on covariates
- It does not generalise to unobserved combinations



What is a Normalizing Flow?

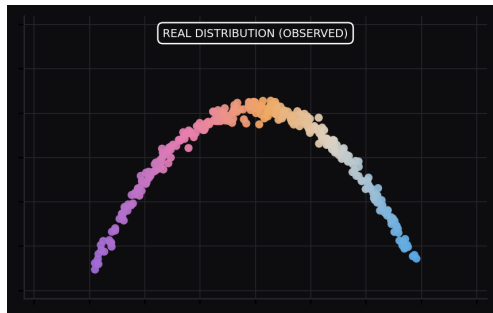
Core idea

An invertible function f that maps a complex distribution p_X into a simple one p_U :

$$U = f(X), \quad X = f^{-1}(U)$$

- **Forward** $X \rightarrow U$: training
by applying translations and rotations
- **Inverse** $Z \rightarrow X$: generation
sample $u \sim \mathcal{N}(0, I)$, then apply f^{-1}
- **No loss of information**
 f is bijective

Normalizing Flows



What is a Normalizing Flow?

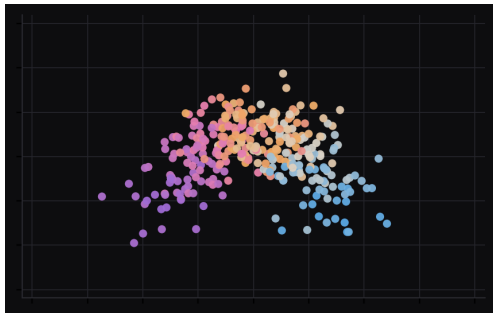
Core idea

An invertible function f that maps a complex distribution p_X into a simple one p_U :

$$U = f(X), \quad X = f^{-1}(U)$$

- **Forward** $X \rightarrow U$: training
by applying translations and rotations
- **Inverse** $Z \rightarrow X$: generation
sample $u \sim \mathcal{N}(0, I)$, then apply f^{-1}
- **No loss of information**
 f is bijective

Normalizing Flows



What is a Normalizing Flow?

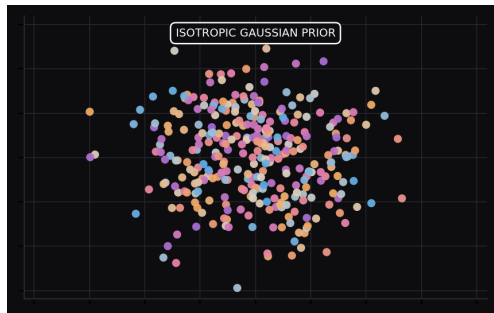
Core idea

An invertible function f that maps a complex distribution p_X into a simple one p_U :

$$U = f(X), \quad X = f^{-1}(U)$$

- **Forward** $X \rightarrow U$: training
by applying translations and rotations
- **Inverse** $Z \rightarrow X$: generation
sample $u \sim \mathcal{N}(0, I)$, then apply f^{-1}
- **No loss of information**
 f is bijective

Normalizing Flows

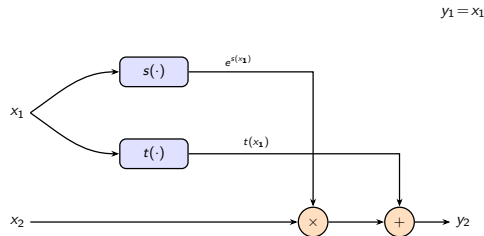


How does an NF work?

Objective function

$$\log p_X(x) = \log p_U(f(x)) + \log |\det J_f(x)|$$

- p_U : standard Gaussian
- $\log |\det J_f|$: how much f “stretches” space: it corrects the density
- We maximise this likelihood during training



How does RealNVP work?

Split $x = (x_1, x_2)$:

$$y_1 = x_1, \quad y_2 = x_2 \cdot e^{s(x_1)} + t(x_1)$$

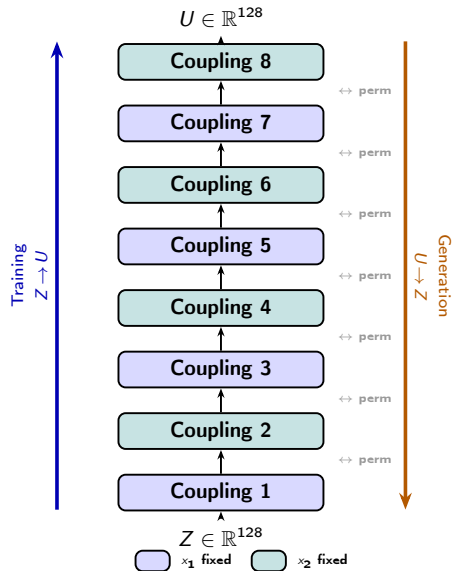
- s, t : *arbitrary* neural networks (they need not be invertible). The roles are then swapped.

RealNVP: architecture used

Configuration

Parameter	Value
Coupling layers	8
Hidden units	256
Activation	ReLU + tanh
Training epochs	500
Learning rate	3×10^{-4}
Input dimension	128 (Z space)

- Prior scaling of Z : StandardScaler $\rightarrow Z_{scaled}$
- Loss: **NLL** (negative log-likelihood)



Advantages and drawbacks of Normalizing Flows

✓ Advantages

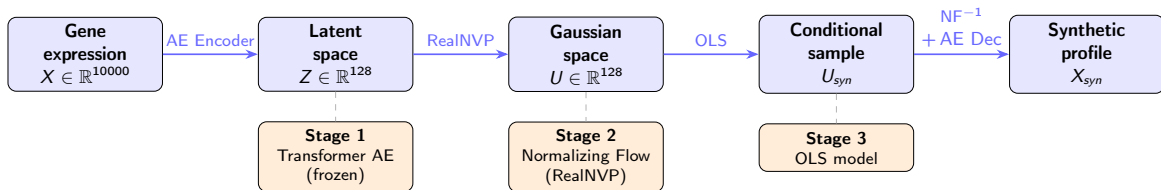
- **Exact likelihood:** $p(x)$ can be evaluated directly
- **Perfect inversion:** no separate decoder needed
- **Gaussian** latent space $\rightarrow$ linear modelling becomes possible
- No training collapse (vs. some GANs)

✗ Drawbacks

- Less flexible than diffusion for images
- Latent dim. = input dim.: **no compression**
- Sensitive to input scaling

	NF	VAE	GAN	Diffusion
Exact likelihood	✓	✗	✗	✗
Efficient inversion	✓	✗	✗	✗
Regular latent space	✓	✓	✗	✗
Generative quality (images)	○	○	✓	✓

Core idea: a three-stage pipeline



Compress

10,000 genes $\rightarrow$ 128 dims
(Transformer + attention)

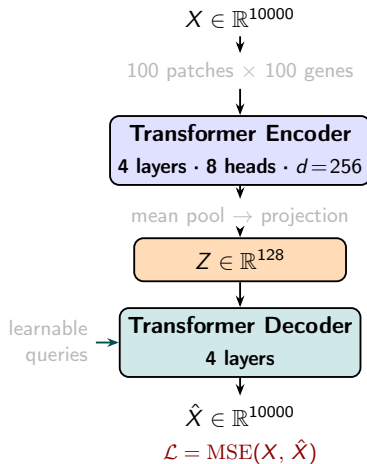
Normalise

Complex distribution $\rightarrow \mathcal{N}(0, I)$
(invertible, exact likelihood)

Condition

$U_{syn} \sim \mathcal{N}(\mu_U(\text{sex, age, tissue}), \Sigma_{res})$
(linear in the U space)

Stage 1: Transformer Autoencoder

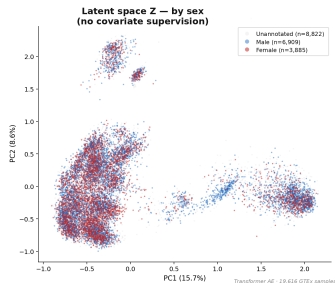
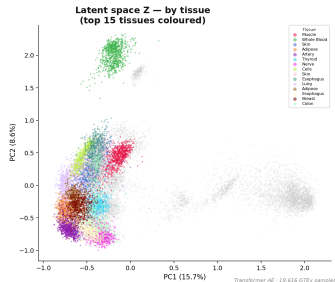


The latent space Z : emergent structure

- Z is a **128-dim** representation of the individual profile
- The AE **never saw** tissue, sex or age during training
- In Z :
 - Tissues group into well-separated clusters
 - Age correlates weakly ($R^2 = 0.185$)

Implication

In Z , the biological variation is **compressed**.
The NFs and the OLS model operate on this representation in order to organise it.



Stage 2: several specialised Normalizing Flows

Why several NFs?

Different covariates require different subsets of the training data

NF	Samples	Use
Universal	19,616 (all)	Tissue
Age NF	10,794 (with age)	Age ($\pm$ tissue)
Sex NF	10,794 (with sex)	Sex ($\pm$ tissue)

- Each NF: $Z \rightarrow U$ (post-flow space, nearly $\mathcal{N}(0, I)$)
- **Automatic pipeline:** the appropriate NF is chosen according to the covariates requested from the generator
- Example: $\{\text{age} + \text{tissue}\} \rightarrow \text{Age+Tissue NF}$

Stage 3: linear OLS model in the U space

Why linear in U ?

In U the distribution is **approximately Gaussian** $\Rightarrow$ linear regression is valid and efficient

OLS model (full example)

$$U = \beta_0 + \beta_s \cdot s + \beta_a \cdot \frac{a - \mu_a}{\sigma_a} + \sum_k \beta_k \cdot \mathbb{1}_{[\text{tis}=k]} + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \Sigma_{\text{res}})$$

- $\beta_0, \beta_s, \beta_a, \beta_k \in \mathbb{R}^{128}$: covariate directions
- Σ_{res} : *unexplained* variation

Generating U_{syn}

$$U_{\text{syn}} \sim \mathcal{N}(\mu_U, \Sigma_{\text{res}})$$

$$\mu_U = \beta_0 + \beta_s s + \beta_a a + \beta_k$$

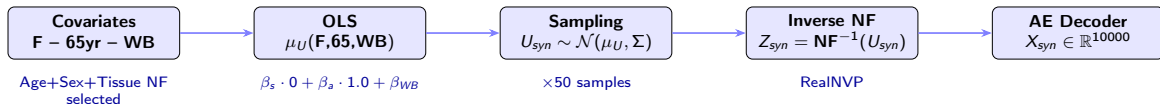
Then:

$$Z_{\text{syn}} = \text{NF}^{-1}(U_{\text{syn}})$$

$$X_{\text{syn}} = \text{AE.decode}(Z_{\text{syn}})$$

Conditional generation example

Example: “Generate 50 Whole Blood samples, female, 65 years old”



API

```
gen.generate(  
  covariates={  
    "sex": ["F"],  
    "age": [65],  
    "tissue": ["WB"]  
  },  
  instances=50)
```

Number of combinations

E.g.: 2 sexes × 3 ages × 2 tissues × 500 samples
= **6,000 synthetic profiles**

How do we evaluate a generator? Three questions

1. FIDELITY

Does it look like real data?

Pearson / Spearman on gene means
Kendall τ on variances
MMD · co-expression ARI

2. UTILITY

Does it carry the
requested signal?

Train-on-Real / Test-on-Synthetic
Accuracy, F1, AUC, MAE
Limma Simpson (DEGs)

3. NOVELTY

Is it more than a copy?

pending
Nearest-neighbour distance ratio

Key point

A generator that scores perfectly on 1 and 2 while failing 3 is simply **memorising the training set**.

Results: Transformer Autoencoder

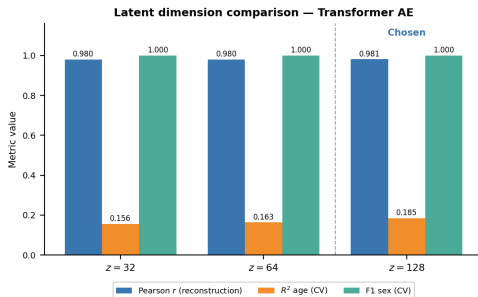
Reconstruction quality ($z = 128$)

Per-gene Pearson r **0.9452**

Co-expression preserv. **0.9991**

Why does co-expression matter?

A co-expression value of 0.9991 between the real and the reconstructed correlation matrices: the Transformer's attention captures joint gene regulation without losing co-expression network information.



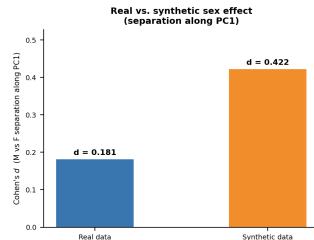
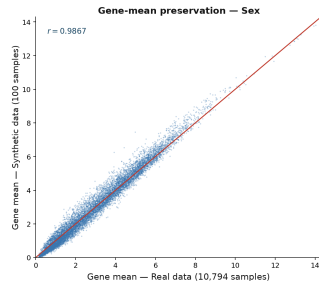
Results: conditioning on sex

Sex classifier (trained on real data)

Metric	Value
F1 macro (real data)	1.0000
AUC (real data)	1.0000
Acc. on synthetic Male	0.880
Acc. on synthetic Female	0.800
Overall acc. on synthetic	0.840

ScGFT biological validation

Pearson r gene means	0.993
Spearman r gene means	0.993
Kendall τ variance	0.713
Co-expression network ARI	0.468
Limma Simpson (sex DEGs)	1.000



Results: conditioning on age

Age prediction (trained on real, evaluated on synthetic)

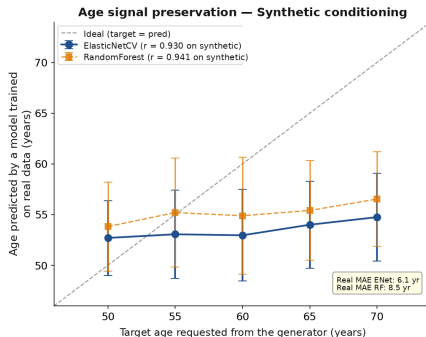
Model	Real MAE (years)	$r(\text{target, pred})$
RandomForest	8.55	0.941
ElasticNetCV	6.06	0.912

Mean MAE synth = 7.2

Real MAE = error on the real test data

Biological validation with ScGFT

Pearson r gene means	0.994
MMD (distance in PCA-50)	0.126
Kendall τ variance	0.709
Co-expression network ARI	0.385



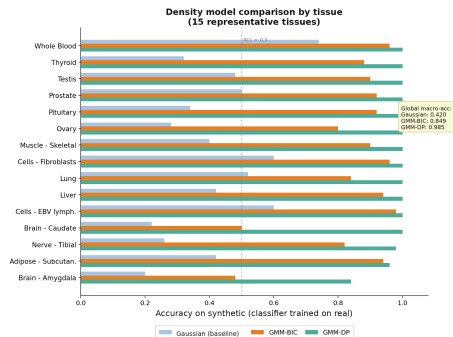
Results: conditioning on tissue

Accuracy on synthetic (classifier trained on real)

Density model	Real F1	Syn. acc.	Pearson r
Gaussian	0.954	0.356	0.973
GMM-BIC	0.954	0.726	0.995
GMM-DP	0.954	0.876	0.997

54 tissues · 50 synthetic/tissue · LogisticRegression

- Most represented tissue: Whole Blood
($r = 0.999$, MMD=0.145)
- Least represented tissue: Cervix-Ectocervix
($n = 24$, MMD=0.351)



Conclusions

- The **AE** $\rightarrow$ **NF** $\rightarrow$ **OLS** pipeline generates RNA-seq profiles conditioned on sex, age and tissue
- High biological fidelity: Pearson ≈ 0.99 on gene means, Limma Simpson = 1.0 for sex
- Covariate signal preserved in the synthetic data:
 $r_{\text{age}} = 0.91$; $\text{Acc}_{\text{sex}} = 0.84$; $F1_{\text{tissue}} = 0.95$
- The GMM-DP density in the U space beats a plain Gaussian for tissues (syn. acc.: 0.36 $\rightarrow$ **0.88**)

Future work

- 1 **Synthetic data to improve ML models**
- 2 **Full combined validation**
- 3 **Adaptation to other datasets: PPMI**
- 4 **A general-purpose “what if?” engine**

Reframing: this is not a GTEx simulator, it is a what-if engine

The general form of the problem

Given high-dimensional measurements X annotated with covariates c , learn $p(X | c)$ and **query it at any c** , including values of c that were never observed together.

$$\text{"What if } c = c^*?" \longrightarrow X_{syn} \sim p(X | c^*)$$

Why this architecture generalises

- The AE assumes only that X is high-dimensional and structured, nothing about genes.
- The NF turns *any* latent geometry into $\mathcal{N}(0, I)$.
- Once in U , the covariate effects become **directions we can add, scale and combine**.

One important caveat

$\mu_U(c^*)$ is defined for every c^* , but validity is not.

- **Interpolation** inside the observed range: reliable, testable by holding out a design cell.
- **Extrapolation** outside it: a linear guess. It should be flagged as such, with a distance-to-support score attached to every generated batch.
- These are *associational* what-ifs. Calling them *causal* requires the usual assumptions (no unmeasured confounding), which observational cohorts rarely grant.

Transferring the methodology to other sectors

Sector	Measurement X	Covariates c	“What if. . . ?” question
Manufacturing / predictive maintenance	Multivariate sensor windows (vibration, temperature, current)	Machine model, line speed, tool age, ambient conditions	<i>What if we ran line B 20% faster with a worn tool?</i> → synthesise rare failure modes that are too costly to observe
Energy	Daily load / generation curves per node	Weather, day type, tariff, installed capacity	<i>What if a heatwave hit on a weekday in 2035?</i> → stress scenarios for grid planning
Finance / credit	Applicant or transaction feature vectors	Macro regime, income band, product, region	<i>What if rates rose 300 bp?</i> → stress testing and fairness audits on under-represented groups
Clinical (beyond omics)	Patient trajectories / EHR vectors	Treatment arm, age, comorbidity	<i>What would the control arm have looked like?</i> → in-silico control arms, digital twins
Agrifood / remote sensing	Spectral signatures, yield profiles	Crop, soil, irrigation, climate year	<i>What if this variety faced a dry season?</i>

Common denominator: an expensive, imbalanced measurement process where the interesting cells of the design are exactly the empty ones.

Questions?

Thank you for your attention :-)