Aug 18, 2025

TemporalVAE for Temporal Prediction on Small Datasets

Nature Cell Biology

DOI
https://dx.doi.org/10.17504/protocols.io.eq2ly47yplx9/v1

Yijun Liu[1]

[1]Jilin University

TemporalVAE

Yijun Liu
Jilin University

OPEN ACCESS

DOI: https://dx.doi.org/10.17504/protocols.io.eq2ly47yplx9/v1

External link: https://doi.org/10.1038/s41556-025-01787-7

Protocol Citation: Yijun Liu 2025. TemporalVAE for Temporal Prediction on Small Datasets. **protocols.io** https://dx.doi.org/10.17504/protocols.io.eq2ly47yplx9/v1

Manuscript citation:
Liu, Y., Cai, F., Barile, M. *et al.* TemporalVAE: atlas-assisted temporal mapping of time-series single-cell transcriptomes during embryogenesis. *Nat Cell Biol* (2025). https://doi.org/10.1038/s41556-025-01787-7

Protocol status: Working
We use this protocol and it's working

Created: August 12, 2025

Last Modified: August 18, 2025

Protocol Integer ID: 224547

Keywords: temporalvae for temporal prediction, small datasets temporalvae, temporal prediction, temporalvae, biological time of cell, workflow of temporalvae, cells in the mouse development atlas, cell staging on mouse organogenesi, biological time, biological time of sample, mouse development atlas, based cell staging, deep generative model, mouse organogenesi, cell, implantation between in vivo, vivo, sensitive gene, implantation, latent space, compressed latent space

Abstract

TemporalVAE is a deep generative model in a dual-objective setting to infer the biological time of cells from a compressed latent space.

TemporalVAE exhibits scalability to millions of cells in the mouse development atlas and high accuracy in atlas-based cell staging on mouse organogenesis across platforms, as well as during human peri-implantation between in vivo and in vitro conditions.

Here we describe the workflow of TemporalVAE, including predicting the biological time of samples and identifying temporally sensitive genes.

Abstract

1 The most important function of TemporalVAE is to predict biological time of cells, we compare the performance of TemporalVAE with multiple existing methods rooted in varied theoretical principles, including **Psupertime**, **Calderon22**, OT-Regressor, Seurat, PCA, LR, and RF.

Here we use three small datasets from Psupertime's experiments, including **Acinar cells**, **Human germline F**, and **Embryonic beta cells**.

Before start: download orginal data file, convert it into a .CSV file with a unified format, and save at folder named demo/data_fromPsupertime

2 **Acinar dataset**
Directly downloaded through the R-package of Psupertime.
R code:

```
setwd('demo/Fig2_TemproalVAE_against_benchmark_methods')

# make Acinar cell dataset ----------------------------------------
---------------------
suppressPackageStartupMessages({
        library('psupertime')
        library('SingleCellExperiment')
})
knitr::opts_chunk$set(collapse = TRUE,comment =
"#>",package.startup.message = FALSE)

# load the data Acinar cells: total 8 donors
data(acinar_hvg_sce)

write.csv(acinar_hvg_sce@assays[[".-
>data"]]@listData[["logcounts"]], file =
"data_fromPsupertime/acinar_hvg_sce_X.csv", row.names = TRUE)
write.csv(acinar_hvg_sce$donor_age,file =
"data_fromPsupertime/acinar_hvg_sce_Y.csv", row.names = FALSE)
```

3 **Embryonic beta cells**
Orginal data can access on **GEO GSE87375** and the developmental stage of beta cells includes **E17.5, P0, P3, P9, P15, P18 and P60**. **We labelled E17.5 as -1**, which is an

embryonic stage before the other stage, and **the other stage as 0, 3, 9, 15, 18, 60**. Then we pre-processed data by the pre-processing function from Psupertime.

Python code:

```
# -*-coding:utf-8 -*-

from demo.Fig2_TemproalVAE_against_benchmark_methods.pypsupertime
import Psupertime
import anndata
import pandas as pd
import os
import numpy as np

os.chdir('demo/Fig2_TemproalVAE_against_benchmark_methods')


def main():
    # for mouse embryonic beta cells dataset:
    result_file_name = "embryoBeta"
    data_org =
pd.read_csv('data_fromPsupertime/GSE87375_Single_Cell_RNA-
seq_Gene_TPM.txt', index_col=0, sep="\t").T
    # get beta cell
    cell_id = data_org.index[1:]
    cell_beta_id = [i for i in cell_id if i[0] == "b"]
    data_org.columns = data_org.iloc[0]
    data_org = data_org[1:]
    data_org = data_org.loc[:, ~data_org.columns.duplicated()]

    adata = anndata.AnnData(data_org)
    adata.var_names_make_unique()
    adata = adata[cell_beta_id].copy()
    temp_time = np.array(adata.obs_names)
    temp_time = [eval(i.split("_")[0].replace("bP",
"").replace("bE17.5", "-1")) for i in temp_time]  # note bE17.5 is
before birth

    adata.obs["time"] = temp_time
    import scanpy as sc
    # sc.pl.highest_expr_genes(adata, n_top=20, )
    sc.pp.filter_genes(adata, min_cells=25)

    adata.var['ERCC'] = adata.var_names.str.startswith('ERCC-')  #
annotate the group of mitochondrial genes as 'ERCC'
    adata = adata[:, ~adata.var.ERCC]
    adata.var['RP'] = adata.var_names.str.startswith('RP')  #
annotate the group of mitochondrial genes as 'RP'
    adata = adata[:, ~adata.var.RP]
```

```
    # sc.pp.calculate_qc_metrics(adata, qc_vars=['ERCC'],
percent_top=None, log1p=False, inplace=True)
    # sc.pl.violin(adata, ['n_genes_by_counts', 'total_counts',
'pct_counts_ERCC'],jitter=0.4, multi_panel=True)
    # sc.pl.scatter(adata, x='total_counts', y='pct_counts_mt')
    # sc.pl.scatter(adata, x='total_counts',
y='n_genes_by_counts')
    preprocessing_params = {"select_genes": "hvg", "log": True}
    # to get hvg gene of humanGermline dataset
    tp = Psupertime(n_jobs=1, n_folds=5,
preprocessing_params=preprocessing_params)
    adata_hvg = tp.preprocessing.fit_transform(adata.copy())
    del tp

    hvg_gene_df = pd.DataFrame(adata_hvg.var_names)
    hvg_gene_df = hvg_gene_df.rename(columns={'Symbol':
'gene_name'})

hvg_gene_df.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_fil
e_name}_gene_list.csv', index=True)

    x_df = data_org.loc[adata_hvg.obs_names]
    x_df = x_df[hvg_gene_df["gene_name"]]
    x_df = x_df.T

x_df.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_file_name}
_X.csv', index=True)

    y_df = pd.DataFrame(adata_hvg.obs.time)

y_df.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_file_name}
_Y.csv', index=True)

    print("Finish save files.")


if __name__ == '__main__':
    main()
```

4 **Human germline F**

Orginal dataset can be access on **GEO GSE86146**. We downloaded the raw count dataset and selected **Female cells** and pre-processed data by the function from

Psupertime.

4.1 Use following R code to combine .txt.gz files:

R code:

```
setwd('demo/Fig2_TemproalVAE_against_benchmark_methods')
# make Human germline dataset ------------------------------------
------------------------

folder_path <- "data_fromPsupertime/GSE86146"

file_list <- list.files(folder_path, pattern = "\\.txt\\.gz$",
full.names = TRUE)

merged_data <- NULL
for (file_path in file_list) {
  data <- read.table(file_path, header = TRUE, sep = "\t", quote =
"",row.names=1)

  if (is.null(merged_data)) {
    merged_data <- data
  } else {
    merged_data <- cbind(merged_data, data)
  }
}

write.csv(merged_data, file =
"data_fromPsupertime/humanGermline_X.csv", row.names = TRUE)
# make label dataframe
new_data <- data.frame()

col_names <- colnames(merged_data)

for (i in 1:length(col_names)) {
  col_name <- col_names[i]
  col_name_parts <- unlist(strsplit(col_name, "_"))

  new_row <- data.frame(
    "sex" = col_name_parts[1],
    "time" = as.numeric(gsub("W", "", col_name_parts[2])),
    row.names = col_name
  )

  new_data <- rbind(new_data, new_row)
}

print(new_data)

write.csv(new_data,file =
```

```
"data_fromPsupertime/humanGermline_Y.csv", row.names = TRUE)
```

4.2 Remove male cell，re-generate data file.
Python code:

```
# -*-coding:utf-8 -*-


from demo.Fig2_TemproalVAE_against_benchmark_methods.pypsupertime
import Psupertime
import anndata
import pandas as pd
import os
import numpy as np

os.chdir('demo/Fig2_TemproalVAE_against_benchmark_methods')


def main():
    # for Human Germline dataset:
    result_file_name = "humanGermline"
    data_x_df =
pd.read_csv('data_fromPsupertime/humanGermline_X.csv',
index_col=0).T
    data_y_df =
pd.read_csv('data_fromPsupertime/humanGermline_Y.csv',
index_col=0)
    # only use female cell
    data_y_df = data_y_df[data_y_df['sex'] != 'M']
    data_x_df = data_x_df.loc[data_y_df.index]
    data_y_df = data_y_df["time"]
    preprocessing_params = {"select_genes": "hvg", "log": True}

    # START HERE
    adata = anndata.AnnData(data_x_df)
    adata.obs["time"] = data_y_df
    print(f"Input Data: n_genes={adata.n_vars}, n_cells=
{adata.n_obs}")

    # to get hvg gene of humanGermline dataset
    tp = Psupertime(n_jobs=1, n_folds=5,
                    preprocessing_params=preprocessing_params)
    adata_hvg = tp.preprocessing.fit_transform(adata.copy())
    hvg_gene = pd.DataFrame(adata_hvg.var_names, columns=
["gene_name"])

hvg_gene.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_file_n
ame}_gene_list.csv', index=True)
```

```
    x_df = data_x_df.loc[adata_hvg.obs_names]
    x_df = x_df[hvg_gene["gene_name"]]
    x_df = x_df.T

x_df.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_file_name}
_X.csv', index=True)

    y_df = pd.DataFrame(adata_hvg.obs.time)

y_df.to_csv(f'{os.getcwd()}/data_fromPsupertime/{result_file_name}
_Y.csv', index=True)

    print("Finish save files.")


if __name__ == '__main__':
    main()
```

Run the analysis code and get results

5 To facilitate the use of the Python code, we provide two GitHub repositories. **TemporalVAE** serves as a user-friendly method library, encapsulating all core functionalities of TemporalVAE for straightforward deployment. **TemporalVAE-reproducibility**, while integrating all the functionalities and source code from **TemporalVAE**, further includes demos that enable the reproduction of all results presented in the corresponding article.

5.1 Configure the environment according to the **README.md** provided in the GitHub repository.

5.2 Download code from https://github.com/StatBiomed/TemporalVAE-reproducibility/tree/releaseV1.0/demo and run the methods using the .py files listed in the table below.

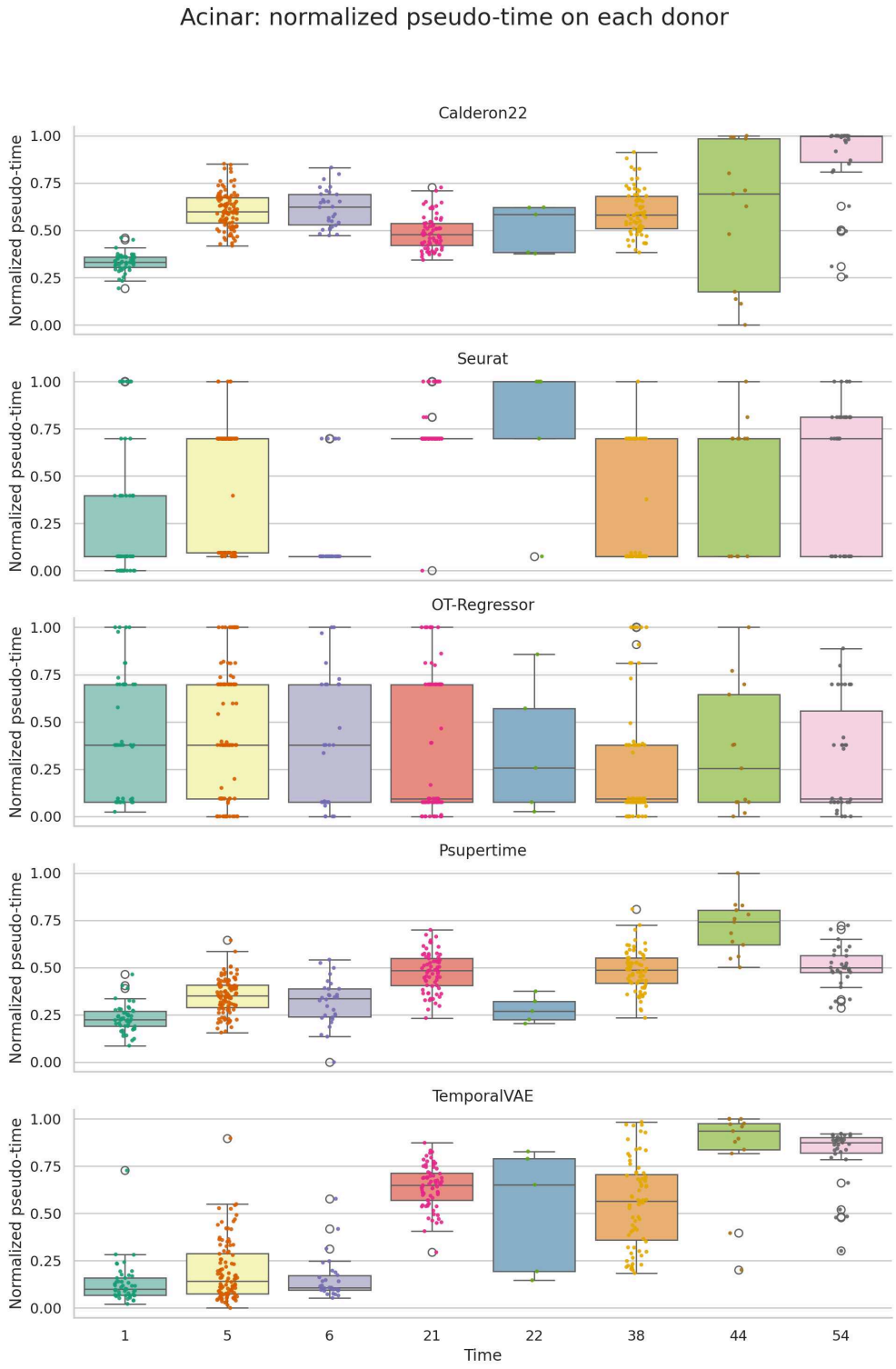
	Method	File name
	Linear regression	exp2_LR_toyDataset.py
	OT-Regressor	exp2_ot_toyDataset.py
	PCA	exp2_pca_toyDataset.py

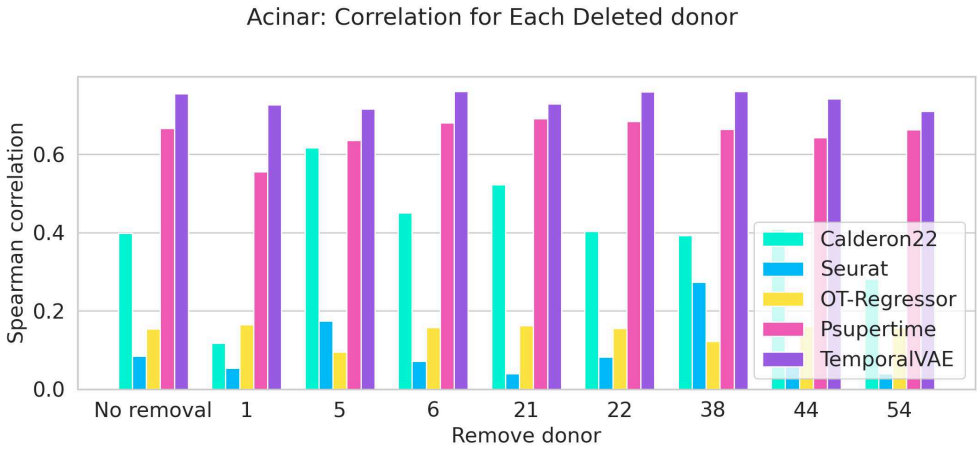
	Method	File name
	Psupertime	exp2_psupertime_toyDataset.py
	Random forest	exp2_randomForest_toyDataset.py
	Calderon22	exp2_science2022_toyDataset.py
	Seurat	exp2_seurat_toyDataset.R
	Vanilla VAE with Linear regression	exp2_VAEwithLR_toyDataset.py
	TemporalVAE	exp2_temporalVAE_toyDataset.py

After run method's .py file, the result will save at *demo/Fig2_TemproalVAE_against_benchmark_methods/{**method**}_results/*, including *{**dataset**}_{**method**}_result.csv* and a prediction density image namd *{**dataset**}_labelsOverPsupertime.png*

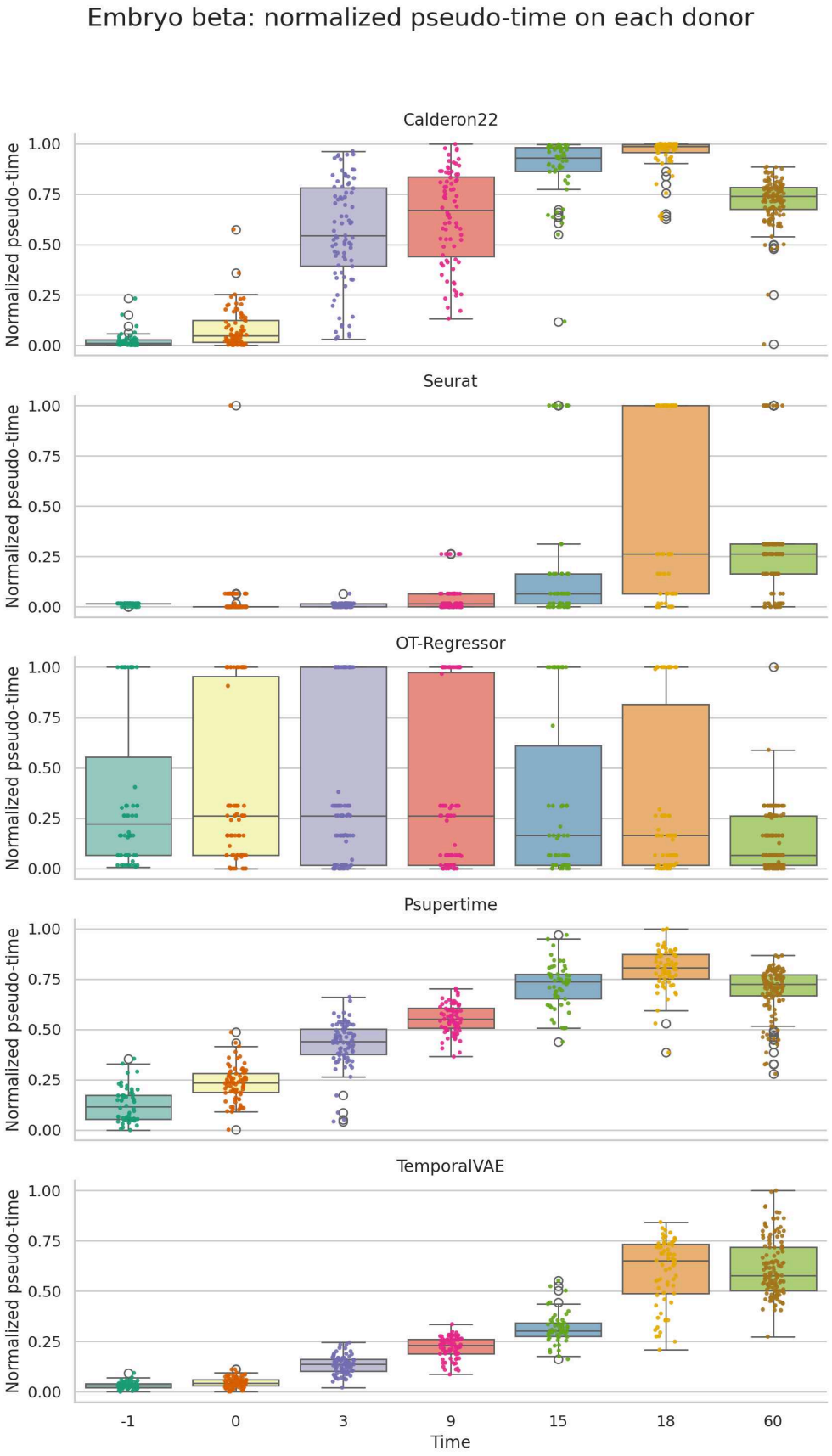
5.3 Run the **plotting code** to obtain comparison graphs of the performance of various methods across three datasets.
The result of each dataset as following:

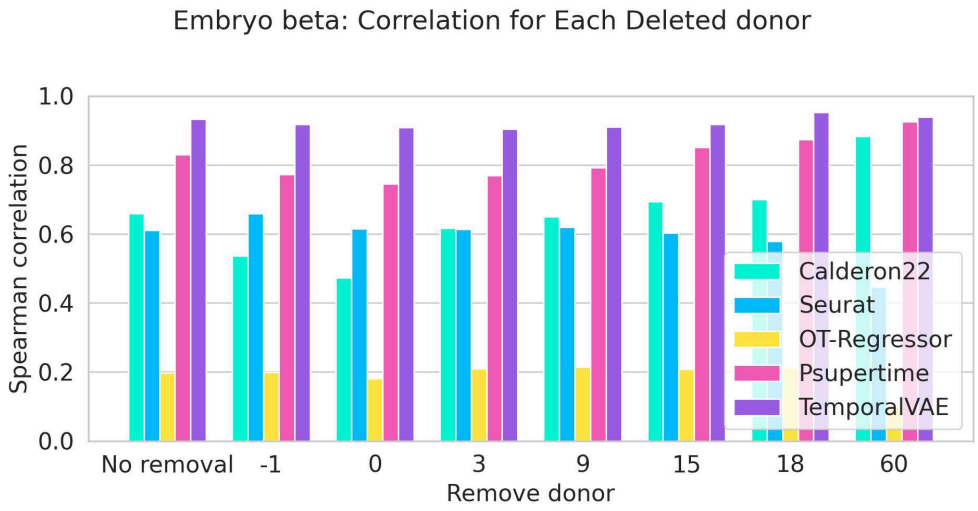
- **Acinar dataset**


Acinar: normalized pseudo-time on each donor
Calderon22
Seurat
OT-Regressor
Psupertime
TemporalVAE
Normalized pseudo-time
0.00
0.25
0.50
0.75
1.00
1
5
6
21
22
38
44
54
Time

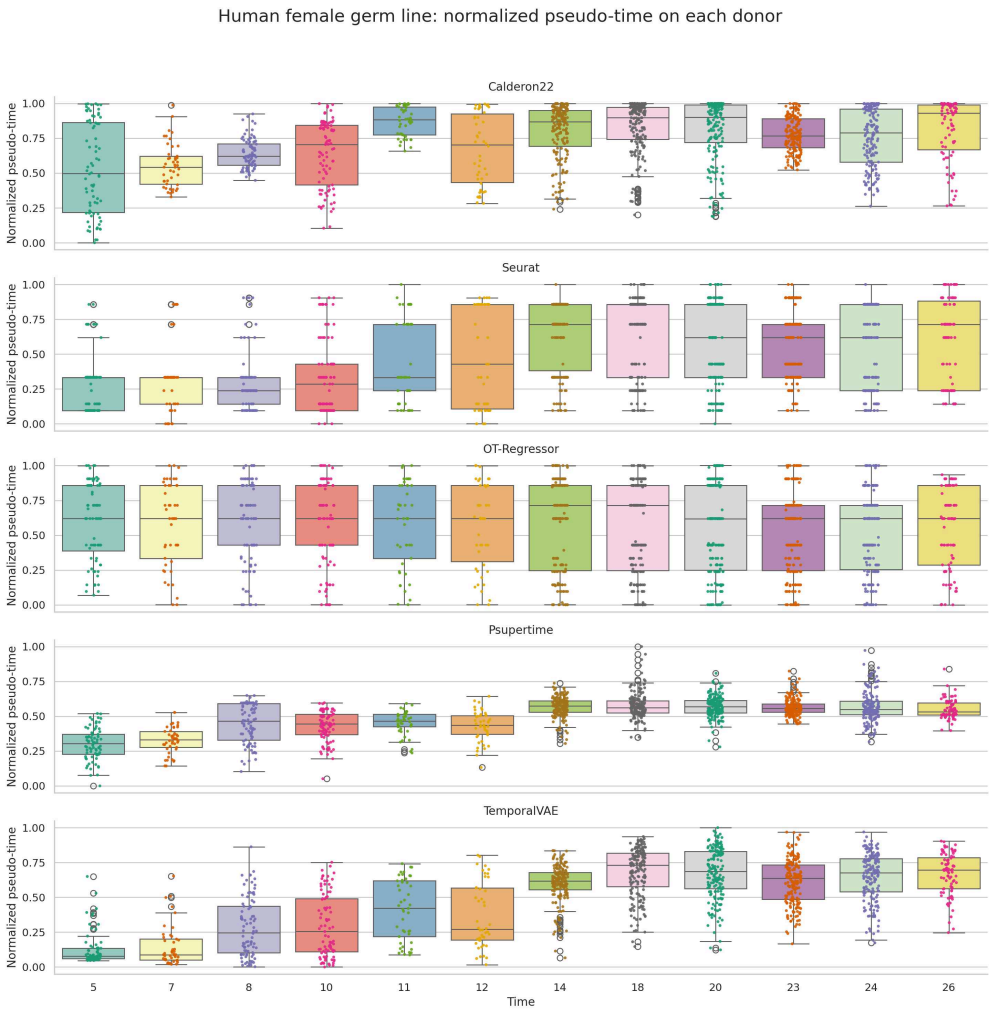


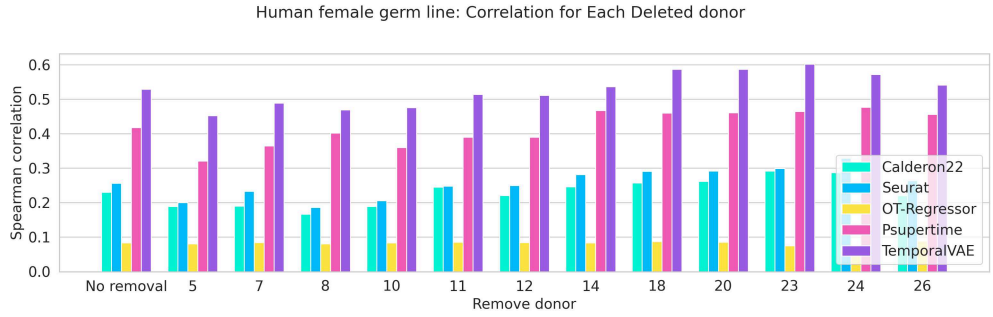
- **Embryonic beta cells**


Embryo beta: normalized pseudo-time on each donor
Calderon22
Seurat
OT-Regressor
Psupertime
TemporalVAE
Normalized pseudo-time
0.00
0.25
0.50
0.75
1.00
-1
0
3
9
15
18
60
Time



- **Human germline F**


Human female germ line: normalized pseudo-time on each donor
Calderon22
Seurat
OT-Regressor
Psupertime
TemporalVAE
Normalized pseudo-time
0.00
0.25
0.50
0.75
1.00
5
7
8
10
11
12
14
18
20
23
24
26
Time


Human female germ line: Correlation for Each Deleted donor
Spearman correlation
0.0
0.1
0.2
0.3
0.4
0.5
0.6
No removal
5
7
8
10
11
12
14
18
20
23
24
26
Remove donor
Calderon22
Seurat
OT-Regressor
Psupertime
TemporalVAE