Appendix

A. Learning OrganSync and benchmarks

We describe the learning procedure for OrganSync, as well as the hyperparameter search-ranges for each benchmark.

A.1. Optimisation procedure (OrganSync)

OrganSync's optimisation procedure is quite straightforward. We have included a formal description of the optimisation procedure, both at train-time (Algorithm 1) as well as at test-time (Algorithm 2). While Algorithm 1 will prepare $f(\cdot)$ through simple supervised learning, Algorithm 2 will output only $\mathbf{a}(\mathbf{X}, \mathbf{O})$. Using $\mathbf{a}(\mathbf{X}, \mathbf{O})$, we can infer point estimates of Y as well as survival estimates S, using the training data $\mathcal{D}_t$, as is described in our main text. Note that all code implementing Algorithm 1 and Algorithm 2 as well as the allocation mechanism described in the main text, is available at `https://github.com/vanderschaarlab/mlforhealthlab`. All hyperparameters and search ranges (for every discussed model) are described below in Section A.2.

Algorithm 1 *OrganSync optimisation procedure at train-time.*

```
Given, Untrained $f_\theta(\cdot)$ and $\beta$,
Given, Train-data $\mathcal{D}_t \subset \mathcal{D}$,
for $e$ in epochs do
    $\mathbf{X}, \mathbf{O}, Y \sim \mathcal{D}_t$
    $\hat{Y} = \beta^\top f_\theta(\mathbf{X}, \mathbf{O})$
    $l = \|\hat{Y} - Y\|_2^2$
    GradientUpdate$(f, l)$
end for
```

Algorithm 2 *OrganSync inference procedure at test-time.*

```
Given, Trained $f_\theta(\cdot)$,
Given, Test-data $\mathcal{D}_{te} \subset \mathcal{D}$,
Given, Train-data $\mathcal{D}_t \subset \mathcal{D}$,
Infer: $Y(\mathbf{O})$, given $\mathbf{X}, \mathbf{O} \sim \mathcal{D}_{te}$.
$\mathbf{u} = f(\mathbf{X}, \mathbf{O})$
$\mathcal{X}_{te}, \mathcal{O}_{te} \sim \mathcal{D}_{te}$
$\mathbf{U} = f(\mathcal{X}_{te}, \mathcal{O}_{te})$
$\mathbf{a}(\mathbf{X}, \mathbf{O}) = \min_{\mathbf{a} \in [0,1]^N} \|\mathbf{U}\mathbf{a} - f(\mathbf{X}, \mathbf{O})\|_2^2 + \lambda \|\mathbf{a}\|_1$
output $\mathbf{a}(\mathbf{X}, \mathbf{O})$
```

A.2. Hyperparameters

In Table 5, we describe the neural network based hyperparameter search ranges, as well as the eventually chosen hyperparameters, for each dataset (the ranges were kept the same throughout searches). Note that for TransplantBenefit, there are no hyperparameters as this boils down to simple CoxPH regressors; and for ConfidentMatch there are none as the ConfidentMatch algorithm, in essence, is a search algorithm over various regressors.

Hyperparameters for the allocation policies are: for OrganITE we have $\alpha = \beta = \gamma = 1$, as was also used in Berrevoets et al. (2020); and for OrganSync we have $K = 20$, empirically this resulted in the best results. Besides OrganSync and OrganITE there are no hyperparameter settings required for the other allocation strategies as they amount to selecting inferred maxima, i.e. the only hyperparameters associated with those allocation policies are those associated with the used models.

B. Assumptions

We make two assumptions w.r.t. the data-generating process: Assumption 1 (*Overlap*), and Assumption 2 (*Unconfoundedness*). While these are standard assumptions in the potential outcomes literature (Rubin, 2005; Hirano & Imbens, 2001; 2004; Shalit et al., 2017; Alaa & van der Schaar, 2017; Athey & Imbens, 2016; Berrevoets et al., 2020), we provide some discussion into how realistic these assumptions are in the organ transplantation setting. Naturally, confirming these assumptions require domain knowledge to verify.

Overlap. With Assumption 1 we assume that every patient on the wait-list has a non-zero probability of receiving the available organ. Despite obvious restrictions such as blood-group type, having only a restricted patient subset available at the organ-arrival time, encourages less likely organ-to-patient allocation in light of patient survival. Furthermore, the data is collected over several years (e.g. the UKReg dataset spans 26 years of transplantation), under several different allocation policies. Having different allocation polices collecting the data benefits the overlap assumption as this by definition changes allocation probability in the queue.

Unconfoundedness. With Assumption 2 we assume that there are no unobserved confounders (variables that influence both treatment assignment, and outcome; $\mathbf{O} \leftarrow Z \rightarrow Y$).

We argue that, in the context of medical data, every variable clinicians in the past relate to organ failure and transplantation success has been highly debated and sought after. After consensus, these data are registered and present in medical datasets. As such, we argue that in the context of medicine, assuming unconfoundedness is more realistic than say marketing, or economics. Should there, in the future, be more variables thought to be informative, than OrganSync (as well as other allocation models) can easily be retrained to incorporate those variables also.

Table 5. **Neural network based hyperparameters**. Using Bayesian optimisation, we searched for optimal hyperparameters using below values. We report for each dataset the optimal (and used) hyperparameters in our experiments.

Parameter	Range	UNOS	UNOS synth.	UKReg	UKReg synth.
		OrganSync			
lr	.001:.05	.004	.004	.0095	.004
lr-decay	.7:.999	.88	.87	.87	.87
weight decay	.00001:.01	.0004	.0003	.00001	.0003
hidden-layers	0:13	4	4	3	4
hidden-dim	8:64	30	30	50	50
$\mathcal{U}$-dim	2:32	8	7	19	8
dropout	0:.3	.01	.01	.006	.009
epochs	20, 30, ..., 70	70	40	50	50
batch size	128, 256, 512, 1024	256	1024	128	128
activation	ReLU, Leaky-ReLU	ReLU	ReLU	Leaky-ReLU	ReLU
		OrganITE			
lr	.001:.05	.006	.006	.007	.007
lr-decay	.7:.999	.79	.80	.79	.75
weight decay	.00001:.01	.00078	.0007	.0006	.0006
hidden-layers	0:64	3	3	4	4
representation width	2:32	6	5	6	5
dropout	0:.3	.1	.1	.11	.11
epochs	20, 30, ..., 70	20	20	30	20
batch size	128, 256, 512, 1024	128	128	128	128
activation	ReLU, Leaky-ReLU	Leaky-ReLU	Leaky-ReLU	Leaky-ReLU	Leaky-ReLU
		Multi-task			
lr	.001:.05	.008	.007	.006	.007
lr-decay	.7:.999	.85	.86	.80	.85
weight decay	.00001:.001	.0003	.0004	.0003	.0004
hidden-layers	0:5	3	2	2	3
cluster count	10:20	15	15	15	15
dropout	0:.3	.17	.16	.15	.17
epochs	20, 30, ..., 70	40	40	40	40
batch size	128, 256, 512, 1024	256	256	256	256
activation	ReLU, Leaky-ReLU	Leaky-ReLU	Leaky-ReLU	ReLU	ReLU

C. Simulation

Correctly evaluating organ-allocation policies is difficult, especially when based on real data. Collected data is naturally heavily biased by existing allocation policies. When testing alternative polices— than those which collected the data — there is no ground-truth as an alternative policy will almost immediately deviate from the data, as organs are possibly matched with different recipients. Furthermore, not only does the available organ determine the match, but also the patients currently in the queue. As such, allocating an organ differently, will completely change the queue onward.

For this, testing new allocation polices (such as OrganSync) requires some approximation. Currently, allocation polices are evaluated using simple linear outcome models (Kilambi et al., 2018; Thompson et al., 2004), such as Transplant-Benefit. Whenever the new allocation policy provides an allocation suggestion, it is evaluated with poor estimates, as is shown in Table 3.

We argue that any simulation should be agnostic to its outcome model, as well as input data. For this we provide an `Inference`, and `OrganDataModule` interface. These two interfaces are combined in a `Sim` class. When a `Sim` is initialised, the wait-list is filled with `initial_waitlist_size` patients, selected randomly from `OrganDataModule`, and their time-to-live is estimated using the given `Inference` module.

The `Sim` then iterates over a given number of `days`, where at each day new patients and organs are sampled. The amount of patients per day is a given `patient_count / days`; and the amount of organs is `round(patient_count`

Table 6. **Used variables for UNOS and UKReg.** All used data is public upon request. Either through (Cecka, 2000) for UNOS, or `https://www.odt.nhs.uk/statistics-and-reports/access-data/` for UKReg.

Variable	UNOS		UKReg	
	patient	organ	patient	organ
Primary liver disease	✓	✗	✓	✗
Age	✓	✓	✓	✓
Registration year	✓	✗	✓	✗
Serum Creatinine	✓	✗	✓	✗
Serum Bilirubin	✓	✗	✓	✗
INR	✓	✗	✓	✗
Serum Sodium	✓	✗	✓	✗
Gender	✓	✗	✓	✓
HCV	✓	✗	✓	✗
Potassium	✗	✗	✓	✗
Albumin	✓	✗	✓	✗
BMI	✗	✓	✓	✓
Donor type (deceased v. living)	✗	✓	✗	✓
Renal support	✓	✗	✓	✗
Ascites	✓	✗	✓	✗
Encephalopathy	✓	✗	✓	✗
Cause of death	✗	✗	✗	✓
RAB Surgery	✓	✗	✗	✗

`/ days * organ_deficit`), where `organ_deficit` is the amount of organs per patient. In our experiments, we set `days = 365`, `patient_count = 1000`, `organ_deficit = 0.7`, which corresponds to the UKReg data.

Whenever the new allocation policy assigns an organ to a patient, they are removed from the wait-list, there W, and their Y (estimated using `Inference`) are added to `population_life_years`. The patients that exceeded their estimated time-to-live on the waiting list are removed and their W is added to `population_life_years`.

Furthermore, we also provide a `Policy` interface. This interface can maintain an internal wait-list (as is the case for OrganSync), and for each available organ is asked a patient from its internal wait-list through `Policy.get_X(organ)`.

Code implementing the above is available at `https://github.com/vanderschaarlab/mlforhealthlab`

D. Data

We use two real datasets: UNOS (Cecka, 2000), and UKReg. Both datasets are publicly available, albeit upon request. We list the covariates used for each dataset below. Furthermore, we also employ a synthetic setup as was also used by Berrevoets et al. (2020).

D.1. Used covariates

Despite different variable naming across UNOS and UKReg, we report in Table 6 the variables used to test all models. We also indicate which variables were used to define $\mathbf{X}$ (patient) as well as $\mathbf{O}$ (organ). As both datasets are available upon request, we will also publicise our data pre-prosessing. We hope this also facilitates usage of our `Sim` through `OrganDataModule`.

D.2. Synthetic data-generation

While discussed in the main text, we provide some additional details on our semi-synthetic data-generation. Using data described in Table 6, we do any pre-processing regularly. This includes, one-hot-encoding categorical variables (e.g. cause of death, gender, etc.), imputation (we use MICE), and normalisation. The latter is not only important for principled learning, but also for semi-synthetic data-generation.

Specifically, we sample $\theta_1, \theta_2 \sim \mathcal{U}(0,1)^n$, where n corresponds to d for $\mathbf{X}$, and e for $\mathbf{O}$. Next, we split the data in a train, and test-set. We then remove Y from both (keeping W). We shuffle patients and organs in the test-set (forming new patient-organ pairs), but keep the allocation intact in the train-set. After shuffling, we compute a semi-synthetic Y for each patient-organ pair using $Y(\mathbf{O}) = \nu \exp\{\theta_1^\top \mathbf{X} + \theta_2^\top \mathbf{O} + \frac{1}{2}\} + \epsilon_Y$. We set $\nu = 1$, though it is easily changed should this be desired; $\epsilon_Y \sim \mathcal{N}(0, 0.1)$,

Table 7. ***Complete* example.** We report the contributors for one example's synthetic control. The absolute error in life-years for OrganSync was 243 days, and 532 days for TransplantBenefit.

Contrib.	Creat.	Bilir.	INR	Sodium	*Year*
Example	46	169	1.1	140	*2019*
	Past patients in synthetic control				
.399	254	238	1.6	134	*2019*
.261	71	35	1.4	135	*2017*
.241	72	20	1.0	142	*2011*
.063	88	100	1.9	123	*2015*
.052	67	70	1.1	141	*2005*
.018	155	44	1.3	132	*2017*

with 0.1 the standard deviation. We than scale the resulting Ys such that the standard deviation and mean matches the original, as well as clip the Ys such that the minimum and maximum values correspond with the original.

E. Complete example

In our main text we include a concrete example of a synthetic patient-organ pair, but restrict our report to top three contributors. We refer to Table 7 for a complete example, reporting more contributors. While the synthetic pair is based on the complete covariate set (through $\mathcal{U}$), we refrain from reporting the entire feature set as the data include sensitive medical information.