Mark van der Laan, Jeremy Coyle, Nima Hejazi, Ivana Malenica, Rachael Phillips, Alan Hubbard

Targeted Learning in R

Causal Data Science with the tlverse Software Ecosystem

Contents

List of Tables

List of Figures

About this book

Targeted Learning in R: Causal Data Science with the tlverse Software Ecosystem is an open source, reproducible electronic handbook for applying the Targeted Learning methodology in practice using the `tlverse` software ecosystem. This work is currently in an early draft phase and is available to facilitate input from the community. To view or contribute to the available content, consider visiting the GitHub repository.

0.0.1 Outline

The contents of this handbook are meant to serve as a reference guide for applied research as well as materials that can be taught in a series of short courses focused on the applications of Targeted Learning. Each section introduces a set of distinct causal questions, motivated by a case study, alongside statistical methodology and software for assessing the causal claim of interest. The (evolving) set of materials includes

- Motivation: Why we need a statistical revolution
- The Roadmap and introductory case study: the WASH Beneifits data
- Introduction to the `tlverse` software ecosystem

- Cross-validation with the `origami` package
- Ensemble machine learning with the `sl3` package
- Targeted learning for causal inference with the `tmle3` package
- Optimal treatments regimes and the `tmle3mopttx` package
- Stochastic treatment regimes and the `tmle3shift` package
- Causal mediation analysis with the `tmle3mediate` package (*work in progress*)
- *Coda*: Why we need a statistical revolution

What this book is not

The focus of this work is **not** on providing in-depth technical descriptions of current statistical methodology or recent advancements. Instead, the goal is to convey key details of state-of-the-art techniques in an manner that is both clear and complete, without burdening the reader with extraneous information. We hope that the presentations herein will serve as references for researchers – methodologists and domain specialists alike – that empower them to deploy the central tools of Targeted Learning in an efficient manner. For technical details and in-depth descriptions of both classical theory and recent advances in the field of Targeted Learning, the interested reader is invited to consult van der Laan and Rose (2011) and/or van der Laan and Rose (2018) as appropriate. The primary literature in statistical causal inference, machine learning, and non/semiparametric theory include many of the most recent advances in Targeted Learning and related areas.

About the authors

Mark van der Laan

Mark van der Laan, PhD, is Professor of Biostatistics and Statistics at UC Berkeley. His research interests include statistical methods in computational biology, survival analysis, censored data, adaptive designs, targeted maximum likelihood estimation, causal inference, data-adaptive loss-based learning, and multiple testing. His research group developed loss-based super learning in semiparametric models, based on cross-validation, as a generic optimal tool for the estimation of infinite-dimensional parameters, such as nonparametric density estimation and prediction with both censored and uncensored data. Building on this work, his research group developed targeted maximum likelihood estimation for a target parameter of the data-generating distribution in arbitrary semiparametric and nonparametric models, as a generic optimal methodology for statistical and causal inference. Most recently, Mark's group has

focused in part on the development of a centralized, principled set of software tools for targeted learning, the `tlverse`.

Jeremy Coyle

Jeremy Coyle, PhD, is a consulting data scientist and statistical programmer, currently leading the software development effort that has produced the `tlverse` ecosystem of R packages and related software tools. Jeremy earned his PhD in Biostatistics from UC Berkeley in 2016, primarily under the supervision of Alan Hubbard.

Nima Hejazi

Nima Hejazi is a PhD candidate in biostatistics, working under the collaborative direction of Mark van der Laan and Alan Hubbard. Nima is affiliated with UC Berkeley's Center for Computational Biology and NIH Biomedical Big Data training program, as well as with the Fred Hutchinson Cancer Research Center. Previously, he earned an MA in Biostatistics and a BA (with majors in Molecular and Cell Biology, Psychology, and Public Health), both at UC Berkeley. His research interests fall at the intersection of causal inference and machine learning, drawing on ideas from non/semi-parametric estimation in large, flexible statistical models to develop efficient and robust statistical procedures for evaluating complex target estimands in observational and randomized studies. Particular areas of current emphasis include mediation/path analysis, outcome-dependent sampling designs, targeted loss-based estimation, and vaccine efficacy trials. Nima is also passionate about statistical computing and open source software development for applied statistics.

Ivana Malenica

Ivana Malenica is a PhD student in biostatistics advised by Mark van der Laan. Ivana is currently a fellow at the Berkeley Institute for Data Science, after serving as a NIH Biomedical Big Data and Freeport-McMoRan Genomic Engine fellow. She earned her Master's in Biostatistics and Bachelor's in Mathematics, and spent some time at the Translational Genomics Research Institute. Very broadly, her research interests span non/semi-parametric theory, probability theory, machine learning, causal inference and high-dimensional statistics. Most of her current work involves complex dependent settings (dependence through time and network) and adaptive sequential designs.

Rachael Phillips

Rachael Phillips is a PhD student in biostatistics, advised by Alan Hubbard and Mark van der Laan. She has an MA in Biostatistics, BS in Biology, and BA in Mathematics. A student of targeted learning and causal inference; her research integrates personalized medicine, human-computer interaction, experimental design, and regulatory policy.

Alan Hubbard

Alan Hubbard is Professor of Biostatistics, former head of the Division of Biostatistics at UC Berkeley, and head of data analytics core at UC Berkeley's SuperFund research program. His current research interests include causal inference, variable importance analysis, statistical machine learning, estimation of and inference for data-adaptive statistical target parameters, and targeted minimum loss-based estimation. Research in his group is generally motivated by applications to problems in computational biology, epidemiology, and precision medicine.

0.0.2 Learning resources

To effectively utilize this handbook, the reader need not be a fully trained statistician to begin understanding and applying these methods. However, it is highly recommended for the reader to have an understanding of basic statistical concepts such as confounding, probability distributions, confidence intervals, hypothesis tests, and regression. Advanced knowledge of mathematical statistics may be useful but is not necessary. Familiarity with the `R` programming language will be essential. We also recommend an understanding of introductory causal inference.

For learning the `R` programming language we recommend the following (free) introductory resources:

- Software Carpentry's *Programming with `R`*
- Software Carpentry's *`R` for Reproducible Scientific Analysis*
- Garret Grolemund and Hadley Wickham's *`R` for Data Science*

For a general introduction to causal inference, we recommend

- Miguel A. Hernán and James M. Robins' *Causal Inference: What If*, 2021
- Jason A. Roy's *A Crash Course in Causality: Inferring Causal Effects from Observational Data* on Coursera

0.0.3 Setup instructions

0.0.3.1 R and RStudio

R and **RStudio** are separate downloads and installations. R is the underlying statistical computing environment. RStudio is a graphical integrated development environment (IDE) that makes using R much easier and more interactive. You need to install R before you install RStudio.

0.0.3.1.1 Windows

0.0.3.1.1.1 If you already have R and RStudio installed

- Open RStudio, and click on "Help" > "Check for updates". If a new version is available, quit RStudio, and download the latest version for RStudio.
- To check which version of R you are using, start RStudio and the first thing that appears in the console indicates the version of R you are running. Alternatively, you can type `sessionInfo()`, which will also display which version of R you are running. Go on the CRAN website and check whether a more recent version is available. If so, please download and install it. You can check here for more information on how to remove old versions from your system if you wish to do so.

0.0.3.1.1.2 If you don't have R and RStudio installed

- Download R from the CRAN website.
- Run the `.exe` file that was just downloaded
- Go to the RStudio download page
- Under *Installers* select **RStudio x.yy.zzz - Windows XP/Vista/7/8** (where x, y, and z represent version numbers)
- Double click the file to install it
- Once it's installed, open RStudio to make sure it works and you don't get any error messages.

0.0.3.1.2 macOS / Mac OS X

0.0.3.1.2.1 If you already have R and RStudio installed

- Open RStudio, and click on "Help" > "Check for updates". If a new version is available, quit RStudio, and download the latest version for RStudio.
- To check the version of R you are using, start RStudio and the first thing that appears on the terminal indicates the version of R you are running. Alternatively, you can type `sessionInfo()`, which will also display which version of R you are running. Go on the CRAN website and check whether a more recent version is available. If so, please download and install it.

0.0.3.1.2.2 If you don't have R and RStudio installed

- Download R from the CRAN website.
- Select the `.pkg` file for the latest R version
- Double click on the downloaded file to install R
- It is also a good idea to install XQuartz (needed by some packages)
- Go to the RStudio download page
- Under *Installers* select **RStudio x.yy.zzz - Mac OS X 10.6+ (64-bit)** (where x, y, and z represent version numbers)
- Double click the file to install RStudio
- Once it's installed, open RStudio to make sure it works and you don't get any error messages.

0.0.3.1.3 Linux

- Follow the instructions for your distribution from CRAN, they provide information to get the most recent version of R for common distributions. For most distributions, you could use your package manager (e.g., for Debian/Ubuntu run

`sudo apt-get install r-base`, and for Fedora `sudo yum install R`), but we don't recommend this approach as the versions provided by this are usually out of date. In any case, make sure you have at least R 3.3.1.
- Go to the RStudio download page
- Under *Installers* select the version that matches your distribution, and install it with your preferred method (e.g., with Debian/Ubuntu `sudo dpkg -i rstudio-x.yy.zzz-amd64.deb` at the terminal).
- Once it's installed, open RStudio to make sure it works and you don't get any error messages.

These setup instructions are adapted from those written for Data Carpentry: R for Data Analysis and Visualization of Ecological Data.

0.1 Robust Statistics and Reproducible Science

> "One enemy of robust science is our humanity — our appetite for being right, and our tendency to find patterns in noise, to see supporting evidence for what we already believe is true, and to ignore the facts that do not fit."
>
> — Anonymous (2015)

Scientific research is at a unique point in history. The need to improve rigor and reproducibility in our field is greater than ever; corroboration moves science forward, yet there is a growing alarm about results that cannot be reproduced and that report false discoveries (Baker, 2016). Consequences of not meeting this need will result in further decline in the rate of scientific progression, the reputation of the sciences, and the public's trust in its findings (Munafò et al., 2017; Editorial, 2015).

> "The key question we want to answer when seeing the results of any scientific study is whether we can trust the data analysis."
>
> — Peng (2015)

Unfortunately, at its current state the culture of data analysis and statistics actually enables human bias through improper model selection. All hypothesis tests and

estimators are derived from statistical models, so to obtain valid estimates and inference it is critical that the statistical model contains the process that generated the data. Perhaps treatment was randomized or only depended on a small number of baseline covariates; this knowledge should and can be incorporated in the model. Alternatively, maybe the data is observational, and there is no knowledge about the data-generating process (DGP). If this is the case, then the statistical model should contain *all* data distributions. In practice; however, models are not selected based on knowledge of the DGP, instead models are often selected based on (1) the p-values they yield, (2) their convenience of implementation, and/or (3) an analysts loyalty to a particular model. This practice of "cargo-cult statistics — the ritualistic miming of statistics rather than conscientious practice," (Stark and Saltelli, 2018) is characterized by arbitrary modeling choices, even though these choices often result in different answers to the same research question. That is, "increasingly often, [statistics] is used instead to aid and abet weak science, a role it can perform well when used mechanically or ritually," as opposed to its original purpose of safeguarding against weak science (Stark and Saltelli, 2018). This presents a fundamental drive behind the epidemic of false findings that scientific research is suffering from (van der Laan and Starmans, 2014).

> "We suggest that the weak statistical understanding is probably due to inadequate"statistics lite" education. This approach does not build up appropriate mathematical fundamentals and does not provide scientifically rigorous introduction into statistics. Hence, students' knowledge may remain imprecise, patchy, and prone to serious misunderstandings. What this approach achieves, however, is providing students with false confidence of being able to use inferential tools whereas they usually only interpret the p-value provided by black box statistical software. While this educational problem remains unaddressed, poor statistical practices will prevail regardless of what procedures and measures may be favored and/or banned by editorials."
>
> — Szucs and Ioannidis (2017)

Our team at The University of California, Berkeley, is uniquely positioned to provide such an education. Spearheaded by Professor Mark van der Laan, and spreading rapidly by many of his students and colleagues who have greatly enriched the field, the aptly named "Targeted Learning" methodology targets the scientific question at hand and is counter to the current culture of "convenience statistics" which opens the door to biased estimation, misleading results, and false discoveries. Targeted Learning restores the fundamentals that formalized the field of statistics, such as the

that facts that a statistical model represents real knowledge about the experiment that generated the data, and a target parameter represents what we are seeking to learn from the data as a feature of the distribution that generated it (van der Laan and Starmans, 2014). In this way, Targeted Learning defines a truth and establishes a principled standard for estimation, thereby inhibiting these all-too-human biases (e.g., hindsight bias, confirmation bias, and outcome bias) from infiltrating analysis.

> "The key for effective classical [statistical] inference is to have well-defined questions and an analysis plan that tests those questions."
>
> — Nosek et al. (2018)

The objective for this handbook is to provide training to students, researchers, industry professionals, faculty in science, public health, statistics, and other fields to empower them with the necessary knowledge and skills to utilize the sound methodology of Targeted Learning — a technique that provides tailored pre-specified machines for answering queries, so that each data analysis is completely reproducible, and estimators are efficient, minimally biased, and provide formal statistical inference.

Just as the conscientious use of modern statistical methodology is necessary to ensure that scientific practice thrives, it remains critical to acknowledge the role that robust software plays in allowing practitioners direct access to published results. We recall that "an article. . . in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures," thus making the availability and adoption of robust statistical software key to enhancing the transparency that is an inherent aspect of science (Buckheit and Donoho, 1995).

For a statistical methodology to be readily accessible in practice, it is crucial that it is accompanied by robust user-friendly software (Pullenayegum et al., 2016; Stromberg et al., 2004). The `tlverse` software ecosystem was developed to fulfill this need for the Targeted Learning methodology. Not only does this software facilitate computationally reproducible and efficient analyses, it is also a tool for Targeted Learning education since its workflow mirrors that of the methodology. In particular, the `tlverse` paradigm does not focus on implementing a specific estimator or a small set of related estimators. Instead, the focus is on exposing the statistical framework of Targeted Learning itself — all `R` packages in the `tlverse` ecosystem directly model the key objects defined in the mathematical and theoretical framework of Targeted

Learning. What's more, the `tlverse` `R` packages share a core set of design principles centered on extensibility, allowing for them to be used in conjunction with each other and built upon one other in a cohesive fashion. For an introduction to Targeted Learning, we recommend the recent review paper from Coyle et al. (2021).

In this handbook, the reader will embark on a journey through the `tlverse` ecosystem. Guided by `R` programming exercises, case studies, and intuitive explanation readers will build a toolbox for applying the Targeted Learning statistical methodology, which will translate to real-world causal inference analyses. Some preliminaries are required prior to this learning endeavor – we have made available a list of recommended learning resources.

0.2 The Roadmap for Targeted Learning

Learning Objectives

By the end of this chapter you will be able to:

1. Translate scientific questions to statistical questions.
2. Define a statistical model based on the knowledge of the experiment that generated the data.
3. Identify a causal parameter as a function of the observed data distribution.
4. Explain the following causal and statistical assumptions and their implications: i.i.d., consistency, interference, positivity, SUTVA.

Introduction

The roadmap of statistical learning is concerned with the translation from real-world data applications to a mathematical and statistical formulation of the relevant estimation problem. This involves data as a random variable having a probability distribution, scientific knowledge represented by a statistical model, a statistical target parameter representing an answer to the question of interest, and the notion of an estimator and sampling distribution of the estimator.

0.2.1 The Roadmap

Following the roadmap is a process of five stages.

1. Data as a random variable with a probability distribution, $O \sim P_0$.
2. The statistical model $\mathcal{M}$ such that $P_0 \in \mathcal{M}$.
3. The statistical target parameter Ψ and estimand $\Psi(P_0)$.
4. The estimator $\hat{\Psi}$ and estimate $\hat{\Psi}(P_n)$.
5. A measure of uncertainty for the estimate $\hat{\Psi}(P_n)$.

(1) Data: A random variable with a probability distribution, $O \sim P_0$

The data set we're confronted with is the result of an experiment and we can view the data as a random variable, O, because if we repeat the experiment we would have a different realization of this experiment. In particular, if we repeat the experiment many times we could learn the probability distribution, P_0, of our data. So, the observed data O with probability distribution P_0 are n independent identically distributed (i.i.d.) observations of the random variable $O; O_1, \ldots, O_n$. Note that while not all data are i.i.d., there are ways to handle non-i.i.d. data, such as establishing conditional independence, stratifying data to create sets of identically distributed data, etc. It is crucial that researchers be absolutely clear about what they actually know about the data-generating distribution for a given problem of interest. Unfortunately, communication between statisticians and researchers is often fraught with misinterpretation. The roadmap provides a mechanism by which to ensure clear communication between research and statistician – it truly helps with this communication!

0.2.1.0.1 The empirical probability measure, P_n

Once we have n of such i.i.d. observations we have an empirical probability measure, P_n. The empirical probability measure is an approximation of the true probability measure P_0, allowing us to learn from our data. For example, we can define the empirical probability measure of a set, A, to be the proportion of observations which end up in A. That is,

$$P_n(A) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(O_i \in A)$$

In order to start learning something, we need to ask *"What do we know about the probability distribution of the data?"* This brings us to Step 2.

(2) The statistical model $\mathcal{M}$ such that $P_0 \in \mathcal{M}$

The statistical model $\mathcal{M}$ is defined by the question we asked at the end of Step 1. It is defined as the set of possible probability distributions for our observed data. Often $\mathcal{M}$ is very large (possibly infinite-dimensional), to reflect the fact that statistical knowledge is limited. In the case that $\mathcal{M}$ is infinite-dimensional, we deem this a nonparametric statistical model.

Alternatively, if the probability distribution of the data at hand is described by a finite number of parameters, then the statistical model is parametric. In this case, we subscribe to the belief that the random variable O being observed has, for example, a normal distribution with mean μ and variance σ^2. Formally, a parametric model may be defined

$$\mathcal{M} = \{P_\theta : \theta \in \mathbb{R}^d\}$$

Sadly, the assumption that the data-generating distribution has a specific, parametric form is all too common, especially since this is a leap of faith or an assumption made of convenience. This practice of oversimplification in the current culture of data analysis typically derails any attempt at trying to answer the scientific question at hand; alas, such statements as the ever-popular quip of Box that "All models are wrong but some are useful" encourage the data analyst to make arbitrary choices even when such a practice often forces starkly different answers to the same estimation problem. The Targeted Learning paradigm does not suffer from this bias since it defines the statistical model through a representation of the true data-generating distribution corresponding to the observed data.

Now, on to Step 3: *"What are we trying to learn from the data?"*

(3) The statistical target parameter Ψ and estimand $\Psi(P_0)$

The statistical target parameter, Ψ, is defined as a mapping from the statistical model, $\mathcal{M}$, to the parameter space (i.e., a real number) $\mathbb{R}$. That is, $\Psi : \mathcal{M} \to \mathbb{R}$. The estimand may be seen as a representation of the quantity that we wish to learn from the data, the answer to a well-specified (often causal) question of interest. In contrast to purely statistical estimands, causal estimands require *identification from the observed data*, based on causal models that include several untestable assumptions, described in more detail in the section on causal target parameters.

For a simple example, consider a data set which contains observations of a survival time on every subject, for which our question of interest is "What's the probability that someone lives longer than five years?" We have,

$$\Psi(P_0) = \mathbb{P}(O > 5)$$

This answer to this question is the **estimand,** $\Psi(P_0)$, which is the quantity we're trying to learn from the data. Once we have defined O, $\mathcal{M}$ and $\Psi(P_0)$ we have formally defined the statistical estimation problem.

(4) The estimator $\hat{\Psi}$ and estimate $\hat{\Psi}(P_n)$

To obtain a good approximation of the estimand, we need an estimator, an *a priori*-specified algorithm defined as a mapping from the set of possible empirical distributions, P_n, which live in a non-parametric statistical model, $\mathcal{M}_{NP}$ ($P_n \in \mathcal{M}_{NP}$), to the parameter space of the parameter of interest. That is, $\hat{\Psi} : \mathcal{M}_{NP} \rightarrow \mathbb{R}^d$. The estimator is a function that takes as input the observed data, a realization of P_n, and gives as output a value in the parameter space, which is the **estimate,** $\hat{\Psi}(P_n)$.

Where the estimator may be seen as an operator that maps the observed data and corresponding empirical distribution to a value in the parameter space, the numerical output that produced such a function is the estimate. Thus, it is an element of the parameter space based on the empirical probability distribution of the observed data. If we plug in a realization of P_n (based on a sample size n of the random variable O), we get back an estimate $\hat{\Psi}(P_n)$ of the true parameter value $\Psi(P_0)$.

In order to quantify the uncertainty in our estimate of the target parameter (i.e., to construct statistical inference), an understanding of the sampling distribution of our estimator will be necessary. This brings us to Step 5.

(5) A measure of uncertainty for the estimate $\hat{\Psi}(P_n)$

Since the estimator $\hat{\Psi}$ is a function of the empirical distribution P_n, the estimator itself is a random variable with a sampling distribution. So, if we repeat the experiment of drawing n observations we would every time end up with a different realization of our estimate and our estimator has a sampling distribution. The sampling distribution of some estimators can be theoretically validated to be approximately normally distributed by a Central Limit Theorem (CLT).

A **Central Limit Theorem** (CLTs) is a statement regarding the convergence of the **sampling distribution of an estimator** to a normal distribution. In general, we will construct estimators whose limit sampling distributions may be shown to be approximately normal distributed as sample size increases. For large enough n we have,

$$\hat{\Psi}(P_n) \sim N\left(\Psi(P_0), \frac{\sigma^2}{n}\right),$$

permitting statistical inference. Now, we can proceed to quantify the uncertainty of our chosen estimator by construction of hypothesis tests and confidence intervals. For example, we may construct a confidence interval at level $(1 - \alpha)$ for our estimand, $\Psi(P_0)$:

$$\hat{\Psi}(P_n) \pm z_{1-\frac{\alpha}{2}} \left(\frac{\sigma}{\sqrt{n}} \right),$$

where $z_{1-\frac{\alpha}{2}}$ is the $(1 - \frac{\alpha}{2})^{\text{th}}$ quantile of the standard normal distribution. Often, we will be interested in constructing 95% confidence intervals, corresponding to mass $\alpha = 0.05$ in either tail of the limit distribution; thus, we will typically take $z_{1-\frac{\alpha}{2}} \approx 1.96$.

Note: we will typically have to estimate the standard error, $\frac{\sigma}{\sqrt{n}}$.

A 95% confidence interval means that if we were to take 100 different samples of size n and compute a 95% confidence interval for each sample, then approximately 95 of the 100 confidence intervals would contain the estimand, $\Psi(P_0)$. More practically, this means that there is a 95% probability that the confidence interval procedure generates intervals containing the true estimand value (or 95% confidence of "covering" the true value). That is, any single estimated confidence interval either will contain the true estimand or will not (also called "coverage").

0.2.2 Summary of the Roadmap

Data, O, is viewed as a random variable that has a probability distribution. We often have n units of independent identically distributed units with probability distribution P_0, such that $O_1, \ldots, O_n \sim P_0$. We have statistical knowledge about the experiment that generated this data. In other words, we make a statement that the true data distribution P_0 falls in a certain set called a statistical model, $\mathcal{M}$. Often these sets are very large because statistical knowledge is very limited - hence, these statistical models are often infinite dimensional models. Our statistical query is, "What are we trying to learn from the data?" denoted by the statistical target parameter, Ψ, which maps the P_0 into the estimand, $\Psi(P_0)$. At this point the statistical estimation problem is formally defined and now we will need statistical theory to guide us in the construction of estimators. There's a lot of statistical theory we will review in this course that, in particular, relies on the Central Limit Theorem, allowing us to come up with estimators that are approximately normally distributed and also allowing us to come with statistical inference (i.e., confidence intervals and hypothesis tests).

0.2.3 Causal Target Parameters

In many cases, we are interested in problems that ask questions regarding the effect of an intervention on a future outcome of interest. These questions can be represented as causal estimands.

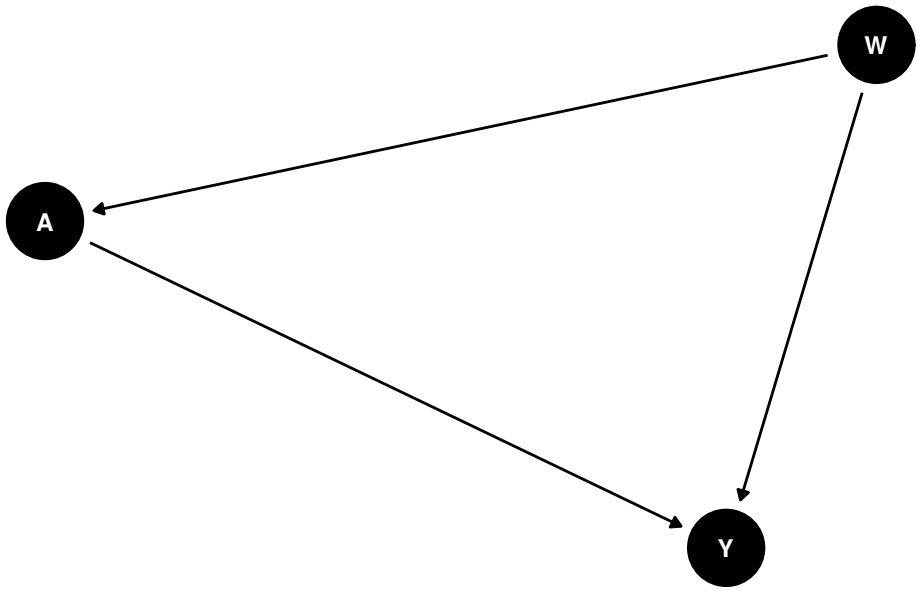
The Causal Model

After formalizing the data and the statistical model, we can define a causal model to express causal parameters of interest. Directed acyclic graphs (DAGs) are one useful tool to express what we know about the causal relations among variables. Ignoring exogenous U terms (explained below), we assume the following ordering of the variables in the observed data O. We do this below using `DAGitty` (Textor et al., 2011):

```
library(dagitty)
library(ggdag)

# make DAG by specifying dependence structure
dag <- dagitty(
  "dag {
    W -> A
    W -> Y
    A -> Y
    W -> A -> Y
  }"
)
exposures(dag) <- c("A")
outcomes(dag) <- c("Y")
tidy_dag <- tidy_dagitty(dag)

# visualize DAG
ggdag(tidy_dag) +
  theme_dag()
```



While directed acyclic graphs (DAGs) like above provide a convenient means by which to visualize causal relations between variables, the same causal relations among variables can be represented via a set of structural equations, which define the nonparametric structural equation model (NPSEM):

$$\begin{aligned} W &= f_W(U_W) \\ A &= f_A(W, U_A) \\ Y &= f_Y(W, A, U_Y), \end{aligned}$$

where U_W, U_A, and U_Y represent the unmeasured exogenous background characteristics that influence the value of each variable. In the NPSEM, f_W, f_A and f_Y denote that each variable (for W, A and Y, respectively) is a function of its parents and unmeasured background characteristics, but note that there is no imposition of any particular functional constraints(e.g., linear, logit-linear, only one interaction, etc.). For this reason, they are called non-parametric structural equation models (NPSEMs). The DAG and set of nonparametric structural equations represent exactly the same information and so may be used interchangeably.

The first hypothetical experiment we will consider is assigning exposure to the whole population and observing the outcome, and then assigning no exposure to the whole population and observing the outcome. On the nonparametric structural equations, this corresponds to a comparison of the outcome distribution in the population under two interventions:

1. A is set to 1 for all individuals, and
2. A is set to 0 for all individuals.

These interventions imply two new nonparametric structural equation models. For the case $A = 1$, we have

$$\begin{aligned} W &= f_W(U_W) \\ A &= 1 \\ Y(1) &= f_Y(W, 1, U_Y), \end{aligned}$$

and for the case $A = 0$,

$$\begin{aligned} W &= f_W(U_W) \\ A &= 0 \\ Y(0) &= f_Y(W, 0, U_Y). \end{aligned}$$

In these equations, A is no longer a function of W because we have intervened on the system, setting A deterministically to either of the values 1 or 0. The new symbols $Y(1)$ and $Y(0)$ indicate the outcome variable in our population if it were generated by the respective NPSEMs above; these are often called *counterfactuals* (since they run contrary-to-fact). The difference between the means of the outcome under these two interventions defines a parameter that is often called the "average treatment effect" (ATE), denoted

$$ATE = \mathbb{E}_X(Y(1) - Y(0)), \tag{0.1}$$

where $\mathbb{E}_X$ is the mean under the theoretical (unobserved) full data $X = (W, Y(1), Y(0))$.

Note, we can define much more complicated interventions on NPSEM's, such as interventions based upon rules (themselves based upon covariates), stochastic rules, etc. and each results in a different targeted parameter and entails different identifiability assumptions discussed below.

Identifiability

Because we can never observe both $Y(0)$ (the counterfactual outcome when $A = 0$) and $Y(1)$ (similarly, the counterfactual outcome when $A = 1$), we cannot estimate the quantity in Equation (0.1) directly. Instead, we have to make assumptions under which this quantity may be estimated from the observed data $O \sim P_0$ under the data-generating distribution P_0. Fortunately, given the causal model specified in the NPSEM above, we can, with a handful of untestable assumptions, estimate the ATE, even from observational data. These assumptions may be summarized as follows.

1. The causal graph implies $Y(a) \perp A$ for all $a \in \mathcal{A}$, which is the *randomization* assumption. In the case of observational data, the analogous assumption is *strong ignorability* or *no unmeasured confounding* $Y(a) \perp A \mid W$ for all $a \in \mathcal{A}$;
2. Although not represented in the causal graph, also required is the assumption of no interference between units, that is, the outcome for unit i Y_i is not affected by exposure for unit j A_j unless $i = j$;
3. *Consistency* of the treatment mechanism is also required, i.e., the outcome for unit i is $Y_i(a)$ whenever $A_i = a$, an assumption also known as "no other versions of treatment";
4. It is also necessary that all observed units, across strata defined by W, have a bounded (non-deterministic) probability of receiving treatment – that is, $0 < \mathbb{P}(A = a \mid W) < 1$ for all a and W). This assumption is referred to as *positivity* or *overlap*.

Remark: Together, (2) and (3), the assumptions of no interference and consistency, respectively, are jointly referred to as the *stable unit treatment value assumption* (SUTVA).

Given these assumptions, the ATE may be re-written as a function of P_0, specifically

$$ATE = \mathbb{E}_0(Y(1) - Y(0)) = \mathbb{E}_0\left(\mathbb{E}_0[Y \mid A = 1, W] - \mathbb{E}_0[Y \mid A = 0, W]\right). \quad (0.2)$$

In words, the ATE is the difference in the predicted outcome values for each subject, under the contrast of treatment conditions ($A = 0$ versus $A = 1$), in the population, averaged over all observations. Thus, a parameter of a theoretical "full" data distribution can be represented as an estimand of the observed data distribution. Significantly, there is nothing about the representation in Equation (0.2) that requires parameteric assumptions; thus, the regressions on the right hand side may be estimated freely with machine learning. With different parameters, there will be potentially different identifiability assumptions and the resulting estimands can be functions of different components of P_0. We discuss several more complex estimands in later sections of this handbook.

0.3 Welcome to the `tlverse`

Learning Objectives

1. Understand the `tlverse` ecosystem conceptually

2. Identify the core components of the `tlverse`
3. Install `tlverse` `R` packages
4. Understand the Targeted Learning roadmap
5. Learn about the WASH Benefits example data

What is the `tlverse`?

The `tlverse` is a new framework for doing Targeted Learning in R, inspired by the `tidyverse` ecosystem of R packages.

By analogy to the `tidyverse`:

> The `tidyverse` is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

So, the `tlverse` is

- an opinionated collection of R packages for Targeted Learning
- sharing an underlying philosophy, grammar, and set of data structures

Anatomy of the `tlverse`

These are the main packages that represent the **core** of the `tlverse`:

- `sl3`: Modern Super Learning with Pipelines
  - *What?* A modern object-oriented re-implementation of the Super Learner algorithm, employing recently developed paradigms for `R` programming.
  - *Why?* A design that leverages modern tools for fast computation, is forward-looking, and can form one of the cornerstones of the `tlverse`.
- `tmle3`: An Engine for Targeted Learning
  - *What?* A generalized framework that simplifies Targeted Learning by identifying and implementing a series of common statistical estimation procedures.
  - *Why?* A common interface and engine that accommodates current algorithmic approaches to Targeted Learning and is still flexible enough to remain the engine even as new techniques are developed.

In addition to the engines that drive development in the `tlverse`, there are some supporting packages – in particular, we have two...

- `origami`: A Generalized Framework for Cross-Validation
  - *What?* A generalized framework for flexible cross-validation
  - *Why?* Cross-validation is a key part of ensuring error estimates are honest and preventing overfitting. It is an essential part of the both the Super Learner algorithm and Targeted Learning.
- `delayed`: Parallelization Framework for Dependent Tasks
  - *What?* A framework for delayed computations (futures) based on task dependencies.
  - *Why?* Efficient allocation of compute resources is essential when deploying large-scale, computationally intensive algorithms.

A key principle of the `tlverse` is extensibility. That is, we want to support new Targeted Learning estimators as they are developed. The model for this is new estimators are implemented in additional packages using the core packages above. There are currently two featured examples of this:

- `tmle3mopttx`: Optimal Treatments in `tlverse`
  - *What?* Learn an optimal rule and estimate the mean outcome under the rule
  - *Why?* Optimal Treatment is a powerful tool in precision healthcare and other settings where a one-size-fits-all treatment approach is not appropriate.
- `tmle3shift`: Shift Interventions in `tlverse`
  - *What?* Shift interventions for continuous treatments
  - *Why?* Not all treatment variables are discrete. Being able to estimate the effects of continuous treatment represents a powerful extension of the Targeted Learning approach.

0.3.1 Installation

The `tlverse` ecosystem of packages are currently hosted at `https://github.com/tlverse`, not yet on CRAN. You can use the `usethis` package to install them:

```
install.packages("devtools")
devtools::install_github("tlverse/tlverse")
```

The `tlverse` depends on a large number of other packages that are also hosted on GitHub. Because of this, you may see the following error:

```
Error: HTTP error 403.
  API rate limit exceeded for 71.204.135.82. (But here's the good news:
  Authenticated requests get a higher rate limit. Check out the documentation
  for more details.)

  Rate limit remaining: 0/60
  Rate limit reset at: 2019-03-04 19:39:05 UTC

  To increase your GitHub API rate limit
  - Use `usethis::browse_github_pat()` to create a Personal Access Token.
  - Use `usethis::edit_r_environ()` and add the token as `GITHUB_PAT`.
```

This just means that R tried to install too many packages from GitHub in too short of a window. To fix this, you need to tell R how to use GitHub as your user (you'll need a GitHub user account). Follow these two steps:

1. Type `usethis::browse_github_pat()` in your R console, which will direct you to GitHub's page to create a New Personal Access Token (PAT).
2. Create a PAT simply by clicking "Generate token" at the bottom of the page.
3. Copy your PAT, a long string of lowercase letters and numbers.
4. Type `usethis::edit_r_environ()` in your R console, which will open your `.Renviron` file in the source window of RStudio.
    a. If your `.Renviron` file does not pop-up after calling `usethis::edit_r_environ()`; then try inputting `Sys.setenv(GITHUB_PAT = "yourPAT")`, replacing your PAT with inside the quotes. If this does not error, then skip to step 8.
5. In your `.Renviron` file, type `GITHUB_PAT=` and then paste your PAT after the equals symbol with no space.

6. In your `.Renviron` file, press the enter key to ensure that your `.Renviron` ends with a new line.

7. Save your `.Renviron` file. The example below shows how this syntax should look.

```
GITHUB_PAT=yourPAT
```

8. Restart R. You can restart R via the drop-down menu on RStudio's "Session" tab, which is located at the top of the RStudio interface. You have to restart R for the changes to take effect!

After following these steps, you should be able to successfully install the package which threw the error above.

0.4 Meet the Data

0.4.1 WASH Benefits Example Dataset

The data come from a study of the effect of water quality, sanitation, hand washing, and nutritional interventions on child development in rural Bangladesh (WASH Benefits Bangladesh): a cluster randomized controlled trial (Tofail et al., 2018). The study enrolled pregnant women in their first or second trimester from the rural villages of Gazipur, Kishoreganj, Mymensingh, and Tangail districts of central Bangladesh, with an average of eight women per cluster. Groups of eight geographically adjacent clusters were block randomized, using a random number generator, into six intervention groups (all of which received weekly visits from a community health promoter for the first 6 months and every 2 weeks for the next 18 months) and a double-sized control group (no intervention or health promoter visit). The six intervention groups were:

1. chlorinated drinking water;
2. improved sanitation;
3. hand-washing with soap;
4. combined water, sanitation, and hand washing;
5. improved nutrition through counseling and provision of lipid-based nutrient supplements; and

6. combined water, sanitation, handwashing, and nutrition.

In the handbook, we concentrate on child growth (size for age) as the outcome of interest. For reference, this trial was registered with ClinicalTrials.gov as NCT01590095.

```
library(readr)
# read in data via readr::read_csv
dat <- read_csv(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-data/master/",
    "wash-benefits/washb_data.csv"
  )
)
```

For the purposes of this handbook, we start by treating the data as independent and identically distributed (i.i.d.) random draws from a very large target population. We could, with available options, account for the clustering of the data (within sampled geographic units), but, for simplification, we avoid these details in the handbook, although modifications of our methodology for biased samples, repeated measures, and related complications, are available.

We have 28 variables measured, of which a single variable is set to be the outcome of interest. This outcome, Y, is the weight-for-height Z-score (`whz` in `dat`); the treatment of interest, A, is the randomized treatment group (`tr` in `dat`); and the adjustment set, W, consists simply of *everything else*. This results in our observed data structure being n i.i.d. copies of $O_i = (W_i, A_i, Y_i)$, for $i = 1, \ldots, n$.

Using the `skimr` package, we can quickly summarize the variables measured in the WASH Benefits data set:

skim_type	skim_variable	n_missing	complete_rate	character.min	character.max	character.e
character	tr	0	1.00000	3	15	
character	fracode	0	1.00000	2	6	
character	sex	0	1.00000	4	6	
character	momedu	0	1.00000	12	15	
character	hfiacat	0	1.00000	11	24	
numeric	whz	0	1.00000	NA	NA	
numeric	month	0	1.00000	NA	NA	
numeric	aged	0	1.00000	NA	NA	
numeric	momage	18	0.99617	NA	NA	
numeric	momheight	31	0.99340	NA	NA	
numeric	Nlt18	0	1.00000	NA	NA	
numeric	Ncomp	0	1.00000	NA	NA	
numeric	watmin	0	1.00000	NA	NA	
numeric	elec	0	1.00000	NA	NA	
numeric	floor	0	1.00000	NA	NA	
numeric	walls	0	1.00000	NA	NA	
numeric	roof	0	1.00000	NA	NA	
numeric	asset_wardrobe	0	1.00000	NA	NA	
numeric	asset_table	0	1.00000	NA	NA	
numeric	asset_chair	0	1.00000	NA	NA	
numeric	asset_khat	0	1.00000	NA	NA	
numeric	asset_chouki	0	1.00000	NA	NA	
numeric	asset_tv	0	1.00000	NA	NA	
numeric	asset_refrig	0	1.00000	NA	NA	
numeric	asset_bike	0	1.00000	NA	NA	
numeric	asset_moto	0	1.00000	NA	NA	
numeric	asset_sewmach	0	1.00000	NA	NA	
numeric	asset_mobile	0	1.00000	NA	NA	

A convenient summary of the relevant variables is given just above, complete with a small visualization describing the marginal characteristics of each covariate. Note that the *asset* variables reflect socio-economic status of the study participants. Notice also the uniform distribution of the treatment groups (with twice as many controls); this is, of course, by design.

0.4.2 International Stroke Trial Example Dataset

The International Stroke Trial database contains individual patient data from the International Stroke Trial (IST), a multi-national randomized trial conducted between 1991 and 1996 (pilot phase between 1991 and 1993) that aimed to assess whether early administration of aspirin, heparin, both aspirin and heparin, or neither influenced the clinical course of acute ischaemic stroke (Sandercock et al., 1997). The IST dataset includes data on 19,435 patients with acute stroke, with 99% complete follow-up. De-identified data are available for download at https://datashare.is.ed.ac.uk/handle/10283/128. This study is described in more detail in Sandercock et al. (2011). The example data for this handbook considers a sample of 5,000 patients and the binary outcome of recurrent ischemic stroke within 14 days after randomization. Also in this example data, we ensure that we have subjects with a missing outcome.

```
# read in data
ist <- read_csv(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-handbook/master/",
    "data/ist_sample.csv"
  )
)
```

We have 26 variables measured, and the outcome of interest, Y, indicates recurrent ischemic stroke within 14 days after randomization (`DRSISC` in `ist`); the treatment of interest, A, is the randomized aspirin vs. no aspirin treatment allocation (`RXASP` in `ist`); and the adjustment set, W, consists of all other variables measured at baseline. In this data, the outcome is occasionally missing, but there is no need to create a variable indicating this missingness (such as Δ) for analyses in the `tlverse`, since it is automatically detected when `NA` are present in the outcome. This observed data structure can be denoted as n i.i.d. copies of $O_i = (W_i, A_i, \Delta_i, \Delta Y_i)$, for $i = 1, \ldots, n$, where Δ denotes the binary indicator that the outcome is observed.

Like before, we can summarize the variables measured in the IST sample data set with `skimr`:

skim_type	skim_variable	n_missing	complete_rate	character.min	character.ma
character	RCONSC	0	1.000	1	
character	SEX	0	1.000	1	
character	RSLEEP	0	1.000	1	
character	RATRIAL	0	1.000	1	
character	RCT	0	1.000	1	
character	RVISINF	0	1.000	1	
character	RHEP24	0	1.000	1	
character	RASP3	0	1.000	1	
character	RDEF1	0	1.000	1	
character	RDEF2	0	1.000	1	
character	RDEF3	0	1.000	1	
character	RDEF4	0	1.000	1	
character	RDEF5	0	1.000	1	
character	RDEF6	0	1.000	1	
character	RDEF7	0	1.000	1	
character	RDEF8	0	1.000	1	
character	STYPE	0	1.000	3	
character	RXHEP	0	1.000	1	
character	REGION	0	1.000	10	2
numeric	RDELAY	0	1.000	NA	N
numeric	AGE	0	1.000	NA	N
numeric	RSBP	0	1.000	NA	N
numeric	MISSING_RATRIAL_RASP3	0	1.000	NA	N
numeric	MISSING_RHEP24	0	1.000	NA	N
numeric	RXASP	0	1.000	NA	N
numeric	DRSISC	10	0.998	NA	N

0.4.3 NHANES I Epidemiologic Follow-up Study (NHEFS)

This data is from the National Health and Nutrition Examination Survey (NHANES) Data I Epidemiologic Follow-up Study. More coming soon.

```
# read in data
nhefs_data <- read_csv(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-handbook/master/",
```

```
    "data/NHEFS.csv"
  )
)
```

A snapshot of the data set is shown below:

skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0
numeric	seqn	0	1.00000	16552.36464	7498.91820	233.00000
numeric	qsmk	0	1.00000	0.26274	0.44026	0.00000
numeric	death	0	1.00000	0.19521	0.39649	0.00000
numeric	yrdth	1311	0.19521	87.56918	2.65941	83.00000
numeric	modth	1307	0.19767	6.25776	3.61530	1.00000
numeric	dadth	1307	0.19767	15.87267	8.90549	1.00000
numeric	sbp	77	0.95273	128.70941	19.05156	87.00000
numeric	dbp	81	0.95028	77.74483	10.63486	47.00000
numeric	sex	0	1.00000	0.50952	0.50006	0.00000
numeric	age	0	1.00000	43.91529	12.17043	25.00000
numeric	race	0	1.00000	0.13198	0.33858	0.00000
numeric	income	62	0.96194	17.94767	2.66328	11.00000
numeric	marital	0	1.00000	2.50338	1.08237	2.00000
numeric	school	0	1.00000	11.13505	3.08960	0.00000
numeric	education	0	1.00000	2.70350	1.19010	1.00000
numeric	ht	0	1.00000	168.74096	9.05313	142.87500
numeric	wt71	0	1.00000	71.05213	15.72959	36.17000
numeric	wt82	63	0.96133	73.46922	16.15805	35.38020
numeric	wt82_71	63	0.96133	2.63830	7.87991	-41.28047
numeric	birthplace	92	0.94352	31.59532	14.50050	1.00000
numeric	smokeintensity	0	1.00000	20.55126	11.80375	1.00000
numeric	smkintensity82_71	0	1.00000	-4.73788	13.74136	-80.00000
numeric	smokeyrs	0	1.00000	24.87109	12.19807	1.00000
numeric	asthma	0	1.00000	0.04850	0.21488	0.00000
numeric	bronch	0	1.00000	0.08533	0.27946	0.00000
numeric	tb	0	1.00000	0.01412	0.11802	0.00000
numeric	hf	0	1.00000	0.00491	0.06993	0.00000
numeric	hbp	0	1.00000	1.05095	0.95821	0.00000
numeric	pepticulcer	0	1.00000	0.10374	0.30502	0.00000
numeric	colitis	0	1.00000	0.03376	0.18067	0.00000
numeric	hepatitis	0	1.00000	0.01719	0.13001	0.00000
numeric	chroniccough	0	1.00000	0.05402	0.22613	0.00000
numeric	hayfever	0	1.00000	0.08963	0.28573	0.00000
numeric	diabetes	0	1.00000	0.97974	0.99579	0.00000
numeric	polio	0	1.00000	0.01412	0.11802	0.00000
numeric	tumor	0	1.00000	0.02333	0.15099	0.00000
numeric	nervousbreak	0	1.00000	0.02885	0.16744	0.00000
numeric	alcoholpy	0	1.00000	0.87600	0.33887	0.00000
numeric	alcoholfreq	0	1.00000	1.92020	1.30714	0.00000
numeric	alcoholtype	0	1.00000	2.47575	1.20816	1.00000
numeric	alcoholhowmuch	417	0.74401	3.28713	2.98470	1.00000
numeric	pica	0	1.00000	0.97545	0.99785	0.00000
numeric	headache	0	1.00000	0.62983	0.48300	0.00000
numeric	otherpain	0	1.00000	0.24616	0.43091	0.00000
numeric	weakheart	0	1.00000	0.02210	0.14705	0.00000
numeric	allergies	0	1.00000	0.06200	0.24123	0.00000

0.5 Cross-validation

Ivana Malenica

Based on the `origami R` package by *Jeremy Coyle, Nima Hejazi, Ivana Malenica and Rachael Phillips.*

Updated: 2021-04-19

0.5.1 Learning Objectives

1. Differentiate between training, validation and test sets.
2. Understand the concept of a loss function, risk and cross-validation.
3. Select a loss function that is appropriate for the functional parameter to be estimated.
4. Understand and contrast different cross-validation schemes for i.i.d. data.
5. Understand and contrast different cross-validation schemes for time dependent data.
6. Setup the proper fold structure, build custom fold-based function, and cross-validate the proposed function using the `origami R` package.
7. Setup the proper cross-validation structure for the use by the Super Learner using the the `origami R` package.

0.5.2 Introduction

In this chapter, we start elaborating on the estimation step outlined in the introductory chapter, which discussed the *Roadmap for Targeted Learning*. In order to generate an initial estimate of our target parameter – which is the focus of the following chapter on Super Learning, we first need to translate, and incorporate, our knowledge about the data generating process into the estimation procedure, and decide how to evaluate our estimation performance.

The performance, or error, of any algorithm used in the estimation procedure directly relates to its generalizability on the independent data. The proper assessment of the performance of proposed algorithms is extremely important; it guides the choice of the final learning method, and it gives us a quantitative assessment of how good the chosen algorithm is doing. In order to assess the performance of an algorithm, we introduce the concept of a **loss** function, which helps us define the **risk**, also referred

to as the **expected prediction error**. Our goal, as further specified in the next chapter, will be to estimate the true risk of the proposed statistical learning method. Our goal(s) consist of:

1. Estimating the performance of different algorithms in order to choose the best one.
2. Having chosen a winner, try to estimate the true risk of the proposed statistical learning method.

In the following, we propose a method to do so using the observed data and **cross-validation** procedure using the `origami` package (Coyle and Hejazi, 2018).

0.5.3 Background

Ideally, in a data-rich scenario, we would split our dataset into three parts:

1. training set,
2. validation set,
3. test set.

The training set is used to fit algorithm(s) of interest; we evaluate the performance of the fit(s) on a validation set, which can be used to estimate prediction error (e.g., for tuning and model selection). The final error of the chosen algorithm(s) is obtained by using the test set, which is kept separately, and doesn't see the data before the final evaluation. One might wonder, with training data readily available, why not use the training error to evaluate the proposed algorithm's performance? Unfortunately, the training error is not a good estimate of the true risk; it consistently decreases with model complexity, resulting in a possible overfit to the training data and low generalizability.

Since data are often scarce, separating it into training, validation and test set is usually not possible. In the absence of a large data set and a designated test set, we must resort to methods that estimate the true risk by efficient sample re-use. Re-sampling methods, in great generality, involve repeatedly sampling from the training set and fitting proposed algorithms on the new samples. While often computationally intensive, re-sampling methods are particularly useful for model selection and estimation of the true risk. In addition, they might provide more insight on variability and robustness of the algorithm fit then fitting an algorithm only once on all the training data.

0.5.3.1 Introducing: cross-validation

In this chapter, we focus on **cross-validation** – an essential tool for evaluating how any given algorithm extends from a sample to the target population from which the sample is derived. It has seen widespread application in all facets of statistics, perhaps most notably statistical machine learning. The cross-validation procedure can be used for model selection, as well as for estimation of the true risk associated with any statistical learning method in order to evaluate its performance. It particular, cross-validation directly estimates the true risk when the estimate is applied to an independent sample from the joint distribution of the predictors and outcome. When used for model selection, cross-validation has powerful optimality properties. The asymptotic optimality results state that the cross-validated selector performs (in terms of risk) asymptotically as well as an optimal oracle selector based on the true, unknown data generating distribution. For further details on the theoretical results, we suggest van der Laan et al. (2004), Dudoit and van der Laan (2005) and Van der Vaart et al. (2006).

In great generality, cross-validation works by partitioning a sample into complementary subsets, applying a particular algorithm(s) on a subset (the training set), and evaluating the method of choice on the complementary subset (the validation/test set). This procedure is repeated across multiple partitions of the data. A variety of different partitioning schemes exist, depending on the problem of interest, data size, prevalence of the outcome, and dependence structure. The `origami` package provides a suite of tools that generalize the application of cross-validation to arbitrary data analytic procedures. In the the following, we describe different types of cross-validation schemes readily available in `origami`, introduce the general structure of the `origami` package, and show their use in applied settings.

0.5.4 Estimation Roadmap: how does it all fit together?

Similarly to how we defined the *Roadmap for Targeted Learning*, we can define the **Estimation Roadmap** to guide the estimation process. In particular, we have developed a unified loss-based cross-validation methodology for estimator construction, selection, and performance assessment in a series of articles (e.g., see van der Laan et al. (2004), Dudoit and van der Laan (2005), Van der Vaart et al. (2006), and van der Laan et al. (2007)) that follow three main steps:

1. **The loss funtion**: Define the target parameter as the minimizer of the expected loss (risk) for a full data loss function chosen to represent the desired performance measure. Map the full data loss function into an observed data loss function, having the same expected value and leading to an efficient estimator of risk.

2. **The algorithms**: Construct a finite collection of candidate estimators for the parameter of interest.

3. **The cross-validation scheme**: Apply appropriate cross-validation to select an optimal estimator among the candidates, and assess the overall performance of the resulting estimator.

Step 1 of the Estimation Roadmap allows us to unify a broad range of problems that are traditionally treated separately in the statistical literature, including density estimation, prediction of polychotomous and continuous outcomes. For example, if we are interested in estimating the full joint conditional density, we could use the negative log-likelihood loss. If instead we are interested in the conditional mean with continuous outcome, one could use the squared error loss; had the outcome been binary, one could resort to the indicator (0-1) loss. The unified loss-based framework also reconciles censored and full data estimation methods, as full data estimators are recovered as special cases of censored data estimators.

0.5.5 Example: cross-validation and prediction

Now that we introduced the Estimation Roadmap, we can define our objective with more mathematical notation, using prediction as an example. Let the observed data be defined as $X = (W, Y)$, where a unit specific data can be written as $X_i = (W_i, Y_i)$, for $i = 1, \ldots, n$. For each of the n samples, we denote Y_i as the outcome of interest (polychotomous or continuous), and W_i as a p-dimensional set of covariates. Let $\psi_0(W)$ denote the target parameter of interest we want to estimate; for this example, we are interested in estimating the conditional expectation of the outcome given the covariates, $\psi_0(W) = E(Y \mid W)$. Following the Estimation Roadmap, we chose the appropriate loss function, L, such that $\psi_0(W) = \text{argmin}_\psi E[L(X, \psi(W))]$. But how do we know how each ψ is doing? In order to pick the optimal estimator among the candidates, and assess the overall performance of the resulting estimator, use cross-validation – dividing the available data into the training set and validation set. Observations in the training set are used to fit (or train) the estimator, while the validation set is used to assess the risk of (or validate) it.

To derive a general representation for cross-validation, we define a **split vector**, $B_n = (B_n(i) : i = 1, \ldots, n) \in \{0, 1\}^n$. Note that split vector is independent of the empirical distribution, P_n. A realization of B_n defines a random split of the data into a training and validation set such that if

$$B_n(i) = 0, \quad \text{i sample is in the training set}$$

$$B_n(i) = 1, \quad \text{i sample is in the validation set.}$$

We can further define P^0_{n,B_n} and P^1_{n,B_n} as the empirical distributions of the training and validation sets, respectively. Then $n_0 = \sum_i 1 - B_n(i)$ and $n_1 = \sum_i B_n(i)$ denote the number of samples in each set. The particular distribution of the split vector B_n defines the type of cross-validation scheme, tailored to the problem and data set in hand.

0.5.6 Cross-validation schemes in `origami`

As we specified earlier, the particular distribution of the split vector B_n defines the type of cross-validation method. In the following, we describe different types of cross-validation schemes available in `origami` package, and show their use in the sequel.

WASH Benefits Study Example

In order to illustrate different cross-validation schemes, we will be using the WASH data. Detailed information on the WASH Benefits Example Dataset can be found in Chapter 3. In particular, we are interested in predicting weight-for-height z-score `whz` using the available covariate data. For this illustration, we will start by treating the data as independent and identically distributed (i.i.d.) random draws. To see what each cross-validation scheme is doing, we will subset the data to only $n = 30$. Note that each row represents an i.i.d. sample, indexed by the row number.

```
library(data.table)
library(origami)
library(knitr)
library(kableExtra)

# load data set and take a peek
washb_data <- fread(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-data/master/",
```

```
    "wash-benefits/washb_data.csv"
  ),
  stringsAsFactors = TRUE
)
```

whz	tr	fracode	month	aged	sex	momage	momedu	momheight	hfiacat
0.00	Control	N05265	9	268	male	30	Primary (1-5y)	146.40	Food Se
-1.16	Control	N05265	9	286	male	25	Primary (1-5y)	148.75	Modera
-1.05	Control	N08002	9	264	male	25	Primary (1-5y)	152.15	Food Se
-1.26	Control	N08002	9	252	female	28	Primary (1-5y)	140.25	Food Se
-0.59	Control	N06531	9	336	female	19	Secondary (¿5y)	150.95	Food Se
-0.51	Control	N06531	9	304	male	20	Secondary (¿5y)	154.20	Severely

Above is a look at the first 30 of the data.

0.5.6.1 Cross-validation for i.i.d. data

0.5.6.1.1 Re-substitution

The re-substitution method is the simplest strategy for estimating the risk associated with fitting a proposed algorithm on a set of observations. Here, all observed data is used for both training and validation set.

We illustrate the usage of the re-substitution method with `origami` package below; we will use the function `folds_resubstitution(n)`. In order to setup `folds_resubstitution(n)`, we just need the total number of samples we want to allocate to training and validation sets; remember that each row of data is a unique i.i.d. sample. Notice the structure of the `origami` output:

1. v: the cross-validation fold
2. training_set: the indexes of the samples in the training set
3. validation_set: the indexes of the samples in the training set.

This structure of the `origami` output (fold(s)) will persist for each of the cross-validation schemes we present in this chapter. Below, we show the fold generated by the re-substitution method:

```
folds_resubstitution(nrow(washb_data))
[[1]]
$v
[1] 1

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30

$validation_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30

attr(,"class")
[1] "fold"
```

0.5.6.1.2 Holdout method

The holdout method, or the validation set approach, consists of randomly dividing the available data into the training set and validation set (holdout set). The model is then fitted on the training set, and further evaluated on the observations in the validation set. Typically, the data is split into 60/40, 70/30 or 80/20 splits.

The holdout method is intuitive, conceptually easy, and computationally not too demanding. However, if we repeat the process of randomly splitting the data into the training and validation set, we might get a different validation loss (e.g., MSE). In particular, the loss over the validation sets might be highly variable, depending on which samples were included in the training/validation split. For classification problems, there is a possibility of an uneven distribution of different classes in the training and validation set unless data is stratified. Finally, note that we are not using all of the data to train and evaluate the performance of the proposed algorithm, which might result in bias.

0.5.6.1.3 Leave-one-out

The leave-one-out cross-validation scheme is closely related to the holdout method.

In particular, it also involves splitting the data into the training and validation set; however, instead of partitioning the observed data into sets of similar size, a single observation is used as a validation set. With that, majority of the units are employed for training (fitting) the proposed algorithm. Since only one unit (for example $x_1 = (w_1, y_1)$) is not used in the fitting process, leave-one-out cross-validation results in a possibly less biased estimate of the true risk; typically, leave-one-out approach will not overestimate the risk as much as the holdout method. On the other hand, since the estimate of risk is based on a single sample, it is typically a highly variable estimate.

We can repeat the process of spiting the data into training and validation set until all samples are part of the validation set at some point. For example, next iteration of the cross-validation might have $x_2 = (w_2, y_2)$ as the validation set and all the rest of $n - 1$ samples as the training set. Repeating this approach n times results in, for example, n squared errors $MSE_1, MSE_2, \ldots, MSE_n$. The estimate of the true risk is the average over the n squared errors. While the leave-one-out cross-validation results in a less biased (albeit, more variable) estimate of risk than the holdout method, it could be expensive to implement if n is large.

We illustrate the usage of the leave-one-out cross-validation with `origami` package below; we will use the function `folds_loo(n)`. In order to setup `folds_loo(n)`, similarly to the re-substitution method, we just need the total number of samples we want to cross-validate. We show the first two folds generated by the leave-one-out cross-validation below.

```
folds <- folds_loo(nrow(washb_data))
folds[[1]]
$v
[1] 1

$training_set
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
[26] 27 28 29 30

$validation_set
[1] 1

attr(,"class")
[1] "fold"
```

```
folds[[2]]
$v
[1] 2

$training_set
 [1]  1  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
[26] 27 28 29 30

$validation_set
[1] 2

attr(,"class")
[1] "fold"
```

0.5.6.1.4 V-fold

An alternative to leave-one-out is V-fold cross-validation. This cross-validation scheme randomly divides the data into v sets (folds) of equal size; for each fold, the number of samples in the validation set are the same. For V-fold cross-validation, one of the folds is treated as a validation set, whereas the proposed algorithm is fit on the remaining $v - 1$ folds in the training set. The loss, for example MSE, is computed on the samples in the validation set. With the proposed algorithm trained and its performance evaluated on the first fold, we repeat this process v times; each time, a different group of samples is treated as a validation set. Note that with V-fold cross-validation we effectively use all of the data to train and evaluate the proposed algorithm without overfitting to the training data. In the end, the V-fold cross-validation results in v estimates of validation error. The final V-fold CV estimate is computed as an average over all the validation losses.

For a dataset with n samples, V-fold cross-validation with $v = n$ is just leave-one-out; similarly, if we set $n = 1$, we can get the holdout method's estimate of algorithm's performance. Despite the obvious computational advantages, V-fold cross-validation often gives more accurate estimates of the true risk. The reason for this comes from the bias-variance trade-off that comes from employing both methods; while leave-one-out might be less biased, it has higher variance. This difference becomes more obvious as $v << n$ (but not too small, as then we increase bias). With V-fold cross-validation, we end up averaging output from v fits that are typically less correlated

than the outputs from leave-one-out fits. Since the mean of many highly correlated quantities has higher variance, leave-one-out estimate of the risk will also have higher variance than the estimate based on V-fold cross-validation.

Let's see V-fold cross-validation with `origami` in action! In the next chapter we will study the Super Learner, an actual algorithm that we fit and evaluate its performance, that uses V-fold as default cross-validation scheme. In order to set up V-fold CV, we need to call function `folds_vfold(n, V)`. Arguments for `folds_vfold(n, V)` require the total number of samples to be cross-validated, and the number of folds we want to get.

At $V = 2$, we get 2 folds with $n/2$ number of samples in both training and validation set.

```
folds <- folds_vfold(nrow(washb_data), V = 2)
folds[[1]]
$v
[1] 1

$training_set
 [1]  2  3  4  6  7  8 11 12 14 15 19 22 23 24 28

$validation_set
 [1]  1  5  9 10 13 16 17 18 20 21 25 26 27 29 30

attr(,"class")
[1] "fold"
folds[[2]]
$v
[1] 2

$training_set
 [1]  1  5  9 10 13 16 17 18 20 21 25 26 27 29 30

$validation_set
 [1]  2  3  4  6  7  8 11 12 14 15 19 22 23 24 28

attr(,"class")
[1] "fold"
```

0.5.6.1.5 Monte Carlo

With Monte Carlo cross-validation, we randomly select some fraction of the data (without replacement) to form the training set; we assign the rest of the samples to the validation set. With that, the data is repeatedly and randomly divided into two sets, a training set of $n_0 = n \cdot (1 - p)$ observations and a validation set of $n_1 = n \cdot p$ observations. This process is then repeated multiple times, generating (at random) new training and validation partitions each time.

Since the partitions are independent across folds, the same sample can appear in the validation set multiple times – note that this is a stark difference between Monte Carlo and V-fold cross-validation. With Monte Carlo cross-validation, one is able to explore many more available partitions than with V-fold cross-validation – resulting in a possibly less variable estimate of the risk, at a cost of an increase in bias.

We illustrate the usage of the Monte Carlo cross-validation with `origami` package below using the function `folds_montecarlo(n, V, pvalidation)`. In order to setup `folds_montecarlo(n, V, pvalidation)`, we need:

1. the total number of samples we want to cross-validate;
2. the number of folds;
3. the proportion of observations to be placed in the validation set.

At $V = 2$ and $pvalidation = 0.2$, we obtain 2 folds with approximately 6 samples in validation set per fold.

```
folds <- folds_montecarlo(nrow(washb_data), V = 2, pvalidation = 0.2)
folds[[1]]
$v
[1] 1

$training_set
 [1] 19 27 16 29 23 12  1  3 18 11  5  7  8  6  9 22 10 25 20 28 15  2 24 26

$validation_set
[1]  4 13 14 17 21 30

attr(,"class")
```

```
[1] "fold"
folds[[2]]
$v
[1] 2

$training_set
 [1] 19 15 28 25 29 11 20 17 14  4  9 12 30  8 27 18 16 10 13  6 24  3 26  1

$validation_set
[1]  2  5  7 21 22 23

attr(,"class")
[1] "fold"
```

0.5.6.1.6 Bootstrap

The bootstrap cross-validation also consists of randomly selecting samples, with replacement, for the training set. The rest of the samples not picked for the training set are allocated to the validation set. This process is then repeated multiple times, generating (at random) new training and validation partitions each time. In contract to the Monte Carlo cross-validation, the total number of samples in a training and validation size across folds is not constant. We also sample with replacement, hence the same samples can be in multiple training sets. The proportion of observations in the validation sets is a random variable, with expectation ~ 0.368.

We illustrate the usage of the bootstrap cross-validation with `origami` package below using the function `folds_bootstrap(n, V)`. In order to setup `folds_bootstrap(n, V)`, we need:

1. the total number of samples we want to cross-validate;
2. the number of folds.

At $V = 2$, we obtain 2 folds with different number of samples in the validation set across folds.

```
folds <- folds_bootstrap(nrow(washb_data), V = 2)
folds[[1]]
$v
[1] 1

$training_set
 [1]  2  5 30  1 29 16 10 11  8 25 28  2 11  2 16 28 15 28  1 27  9 19 20 30 18
[26] 11 13  2 18 12

$validation_set
 [1]  3  4  6  7 14 17 21 22 23 24 26

attr(,"class")
[1] "fold"
folds[[2]]
$v
[1] 2

$training_set
 [1] 12 16 10 29 22 15 27  9 27 16 12 28 10 28 26  1 14  6 23 14 21 16  5 20  8
[26] 23 25  8 27  5

$validation_set
 [1]  2  3  4  7 11 13 17 18 19 24 30

attr(,"class")
[1] "fold"
```

0.5.6.2 Cross-validation for dependent data

The `origami` package also supports numerous cross-validation schemes for time-series data, for both single and multiple time-series with arbitrary time and network dependence.

AirPassenger Example

In order to illustrate different cross-validation schemes for time-series, we will be using the AirPassenger data; this is a widely used, freely available dataset. The

AirPassenger dataset in R provides monthly totals of international airline passengers from 1949 to 1960. This dataset is already of a time series class therefore no further class or date manipulation is required.

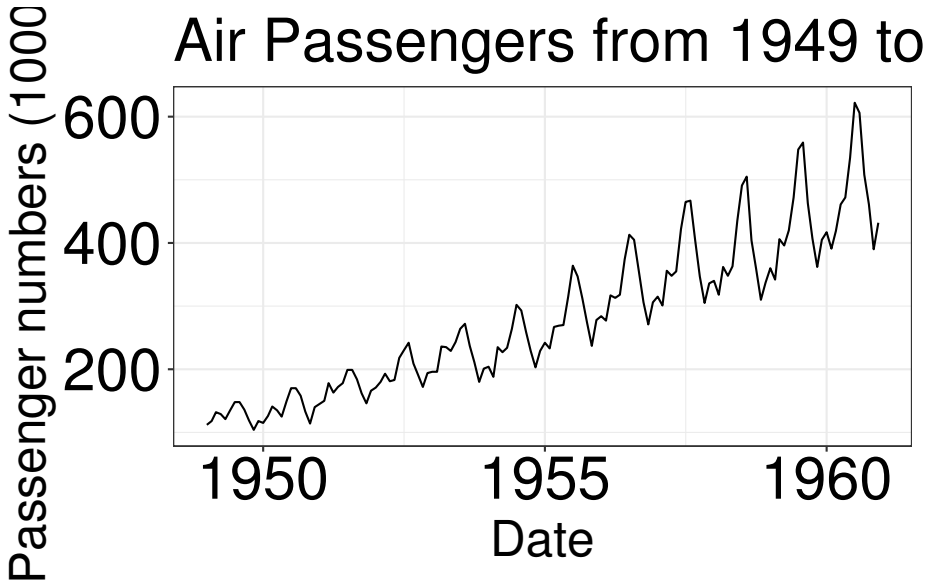
Goal: we want to forecast the number of airline passengers at time h horizon using the historical data from 1949 to 1960.

```
library(ggfortify)

data(AirPassengers)
AP <- AirPassengers

autoplot(AP) +
  labs(
    x = "Date",
    y = "Passenger numbers (1000's)",
    title = "Air Passengers from 1949 to 1961"
  )

t <- length(AP)
```



0.5.6.2.1 Rolling origin

Rolling origin cross-validation scheme lends itself to "online" algorithms, where large streams of data have to be fit continually, and the final fit is constantly updated with more data acquired. In general, the rolling origin scheme defines an initial training set, and with each iteration the size of the training set grows by m observations until we reach time t for a particular fold. The time points included in the training set are always behind the validation set time points; in addition, there might be a gap between training and validation times of size h.

To further illustrate rolling origin cross-validation, we show below an example with 3 folds. Here, the first window size is 15 time points, on which we first train the proposed algorithm. We then evaluate its performance on 10 time points, with a gap of size 5 between the training and validation time points. For the following fold, we train the algorithm on a longer stream of data, 25 time points, including the original 15 we started with. We then evaluate its performance on 10 time points in the future.

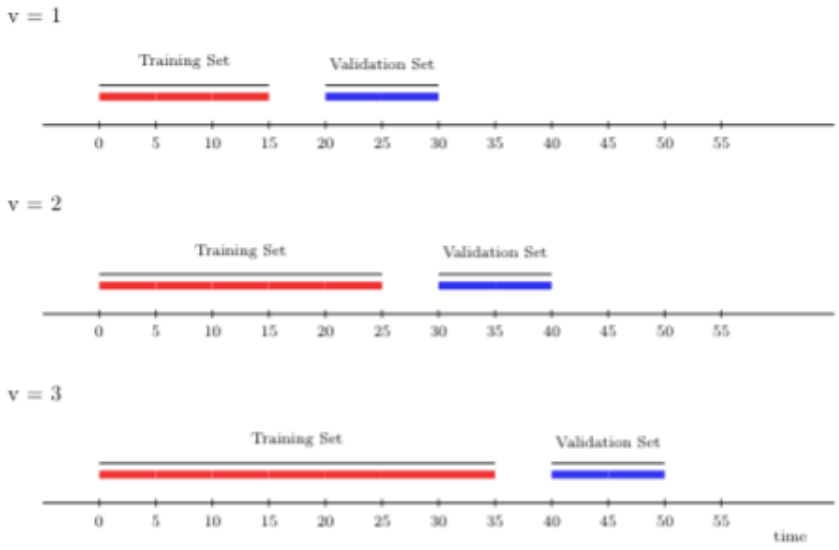


Figure 1: Rolling origin CV

We illustrate the usage of the rolling origin cross-validation with `origami` package below using the function `folds_rolling_origin(n, first_window, validation_size, gap, batch)`. In order to setup `folds_rolling_origin(n, first_window, validation_size, gap, batch)`, we need:

1. the total number of time points we want to cross-validate

2. the size of the first training set
3. the size of the validation set
4. the gap between training and validation set
5. the size of the update on the training set per each iteration of CV

Our time-series has $t = 144$ time points. Setting the `first_window` to 50, `validation_size` to 10, `gap` to 5 and `batch` to 20, we get 4 time-series folds; we show the first two below.

```
folds <- folds_rolling_origin(
  t,
  first_window = 50, validation_size = 10, gap = 5, batch = 20
)
folds[[1]]
$v
[1] 1

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

$validation_set
 [1] 56 57 58 59 60 61 62 63 64 65

attr(,"class")
[1] "fold"
folds[[2]]
$v
[1] 2

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
[51] 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70

$validation_set
 [1] 76 77 78 79 80 81 82 83 84 85
```

```
attr(,"class")
[1] "fold"
```

0.5.6.2.2 Rolling window

Instead of adding more time points to the training set per each iteration, the rolling window cross-validation scheme "rolls" the training sample forward by m time units. The rolling window scheme might be considered in parametric settings when one wishes to guard against moment or parameter drift that is difficult to model explicitly; it is also more efficient for computationally demanding settings such as streaming data, in which large amounts of training data cannot be stored. In contrast to rolling origin CV, the training sample for each iteration of the rolling window scheme is always the same.

To illustrate the rolling window cross-validation with 3 time-series folds below. The first window size is 15 time points, on which we first train the proposed algorithm. As in the previous illustration, we evaluate its performance on 10 time points, with a gap of size 5 between the training and validation time points. However, for the next fold, we train the algorithm on time points further away from the origin (here, 10 time points). Note that the size of the training set in the new fold is the same as in the first fold (15 time points). This setup keeps the training sets comparable over time (and fold) as compared to the rolling origin CV. We then evaluate the performance of the proposed algorithm on 10 time points in the future.

We illustrate the usage of the rolling window cross-validation with `origami` package below using the function `folds_rolling_window(n, window_size, validation_size, gap, batch)`. In order to setup `folds_rolling_window(n, window_size, validation_size, gap, batch)`, we need:

1. the total number of time points we want to cross-validate
2. the size of the training sets
3. the size of the validation set
4. the gap between training and validation set
5. the size of the update on the training set per each iteration of CV

Setting the `window_size` to 50, `validation_size` to 10, `gap` to 5 and `batch` to 20, we also get 4 time-series folds; we show the first two below.

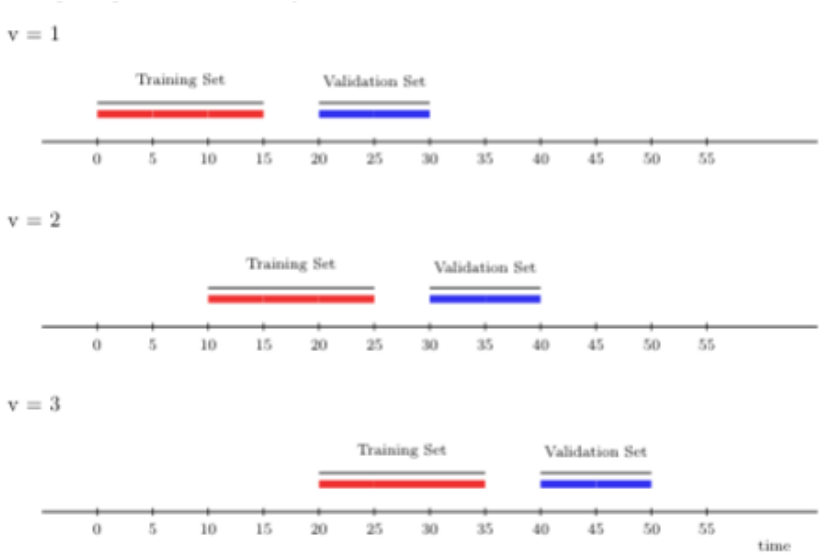


Figure 2: Rolling window CV

```
folds <- folds_rolling_window(
  t,
  window_size = 50, validation_size = 10, gap = 5, batch = 20
)
folds[[1]]
$v
[1] 1

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

$validation_set
 [1] 56 57 58 59 60 61 62 63 64 65

attr(,"class")
[1] "fold"
folds[[2]]
$v
[1] 2
```

```
$training_set
 [1] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
[26] 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70

$validation_set
 [1] 76 77 78 79 80 81 82 83 84 85

attr(,"class")
[1] "fold"
```

0.5.6.2.3 Rolling origin with V-fold

A variant of rolling origin scheme which accounts for sample dependence is the rolling-origin-V-fold cross-validation. In contrast to the canonical rolling origin CV, samples in the training and validation set are not the same, as the variant encompasses V-fold CV in addition to the time-series setup. The predictions are evaluated on the future times of time-series units not seen during the training step, allowing for dependence in both samples and time. One can use the rolling-origin-v-fold cross-validation with `origami` package using the function `folds_vfold_rolling_origin_pooled(n, t, id, time, V, first_window, validation_size, gap, batch)`. In the figure below, we show $V = 2$ V-folds, and 2 time-series CV folds.

0.5.6.2.4 Rolling window with v-fold

Analogous to the previous section, we can extend rolling window CV to support multiple time-series with arbitrary sample dependence. One can use the rolling-window-V-fold cross-validation with `origami` package using the function `folds_vfold_rolling_window_pooled(n, t, id, time, V, window_size, validation_size, gap, batch)`. In the figure below, we show $V = 2$ V-folds, and 2 time-series CV folds.

0.5.7 General workflow of `origami`

Before we dive into more details, let's take a moment to review some of the basic functionality in `origami` R package. The main function in the `origami` is

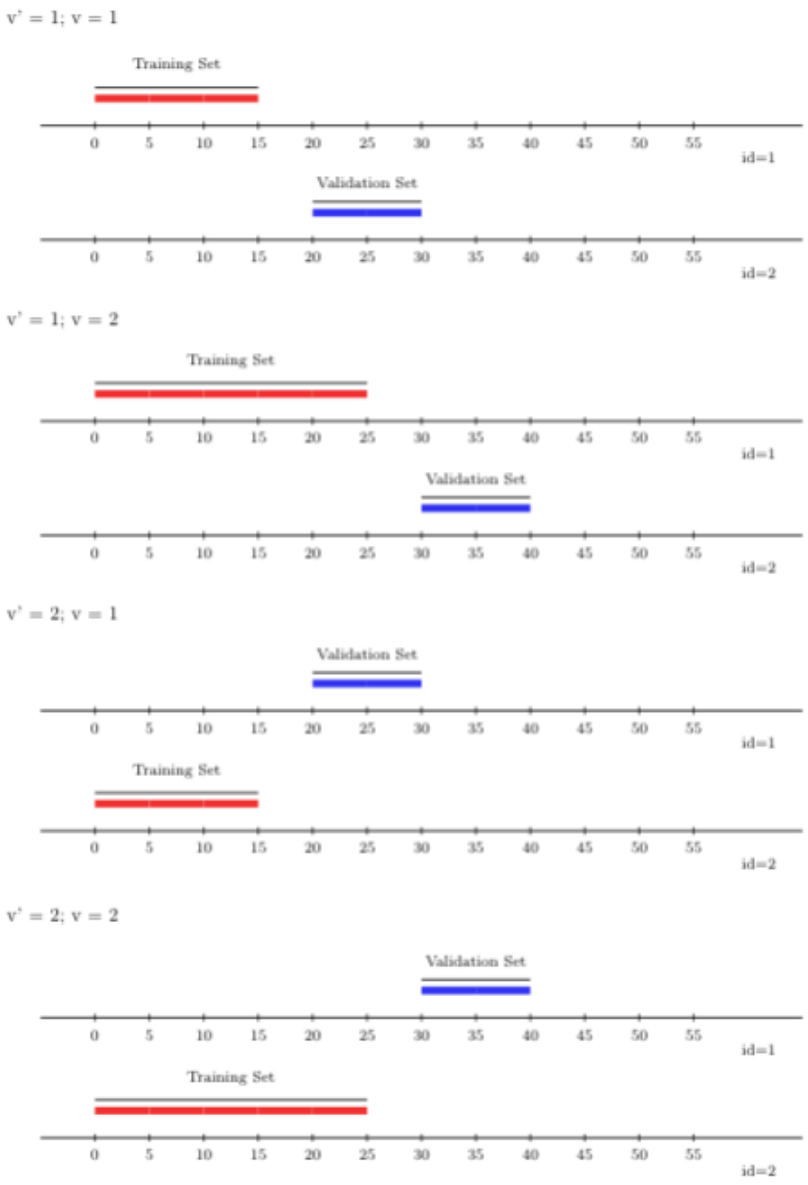


Figure 3: Rolling origin V-fold CV

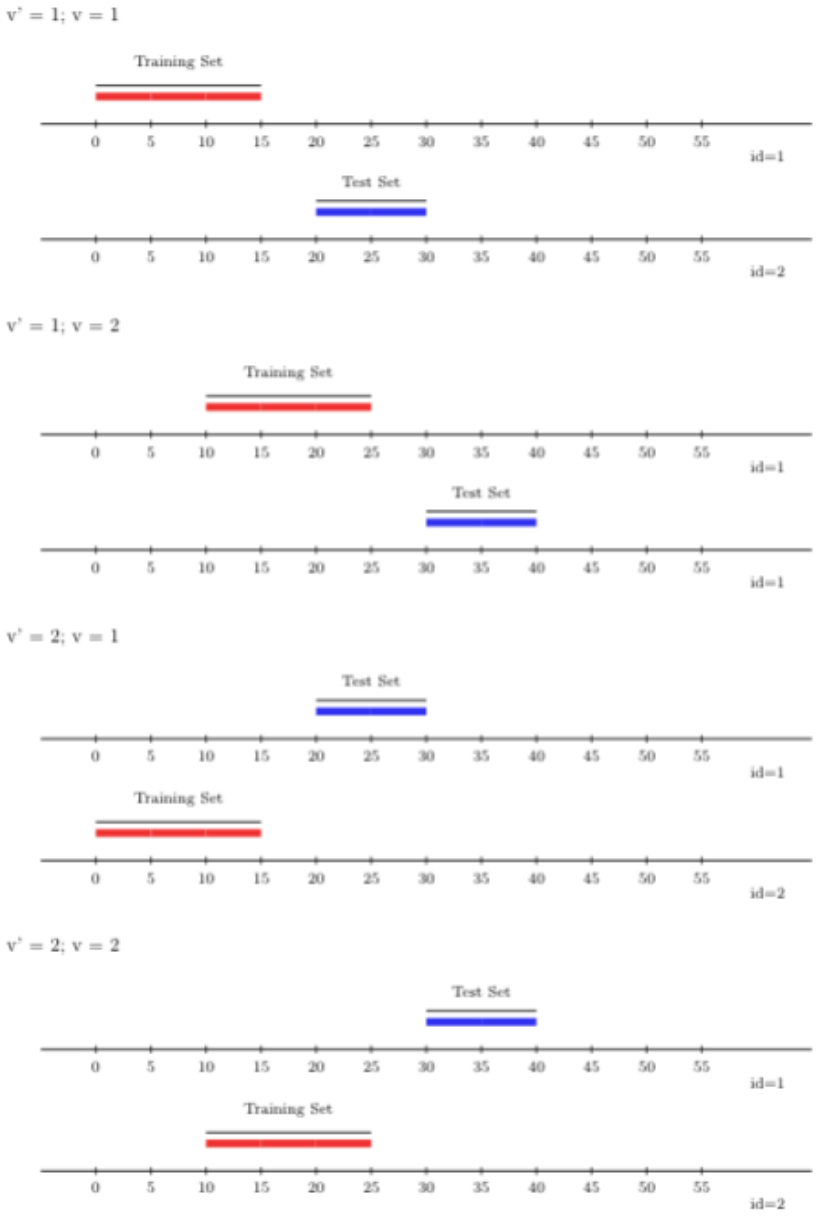


Figure 4: Rolling window V-fold CV

`cross_validate`. To start off, the user must define folds and a function that operates on each fold. Once these are passed to `cross_validate`, this function will map the fold-specific function across the folds, combining the results in a reasonable way. We will see this in action in later sections; for now, we provide specific details on each each step of this process below.

0.5.7.1 (1) Define folds

The `folds` object passed to `cross_validate` is a list of folds; such lists can be generated using the `make_folds` function. Each fold consists of a list with a `training` index vector, a `validation` index vector, and a `fold_index` (its order in the list of folds). This function supports a variety of cross-validation schemes we describe in the following section. The `make_folds` can balance across levels of a variable (`strata_ids`), and it can also keep all observations from the same independent unit together (`cluster`).

0.5.7.2 (2) Define fold function

The `cv_fun` argument to `cross_validate` is a function that will perform some operation on each fold. The first argument to this function must be `fold`, which will receive an individual fold object to operate on. Additional arguments can be passed to `cv_fun` using the `...` argument to `cross_validate`. Within this function, the convenience functions `training`, `validation` and `fold_index` can return the various components of a fold object. If `training` or `validation` is passed an object, it will index into it in a sensible way. For instance, if it is a vector, it will index the vector directly. If it is a `data.frame` or `matrix`, it will index rows. This allows the user to easily partition data into training and validation sets. The fold function must return a named list of results containing whatever fold-specific outputs are generated.

0.5.7.3 (3) Apply `cross_validate`

After defining folds, `cross_validate` can be used to map the `cv_fun` across the `folds` using `future_lapply`. This means that it can be easily parallelized by specifying a parallelization scheme (i.e., a `plan` from the future parallelization framework for `R` (Bengtsson, 2020)). The application of `cross_validate` generates a list of results. As described above, each call to `cv_fun` itself returns a list of results, with different elements for each type of result we care about. The main loop generates a list of these individual lists of results (a sort of "meta-list"). This "meta-list" is then inverted such that there is one element per result type (this too is a list of the results for each fold).

By default, `combine_results` is used to combine these results type lists in a sensible manner. How results are combined is determined automatically by examining the data types of the results from the first fold. This can be modified by specifying a list of arguments to `.combine_control`.

0.5.8 Cross-validation in action

Let's see `origami` in action! In the following chapter we will learn how to use cross-validation with the Super Learner, and how we can utilize the power of cross-validation to build optimal ensembles of algorithms, not just its use on a single statistical learning method.

0.5.8.1 Cross-validation with linear regression

First, we will load the relevant `R` packages, set a seed, and load the full WASH data once again. In order to illustrate cross-validation with `origami` and linear regression, we will focus on predicting the weight-for-height Z-score `whz` using all of the available covariate data. As stated previously, we will assume the data is independent and identically distributed, ignoring the cluster structure imposed by the clinical trial design. For the sake of illustration, we will work with a subset of data, and remove all samples with missing data from the dataset; we will learn in the next chapter how to deal with missingness.

```
library(stringr)
library(dplyr)
library(tidyr)

# load data set and take a peek
washb_data <- fread(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-data/master/",
    "wash-benefits/washb_data.csv"
  ),
  stringsAsFactors = TRUE
)

# Remove missing data, then pick just the first 500 rows
washb_data <- washb_data %>%
```

```
  drop_na() %>%
  slice(1:500)

outcome <- "whz"
covars <- colnames(washb_data)[-which(names(washb_data) == outcome)]
```

Here's a look at the data:

whz	tr	fracode	month	aged	sex	momage	momedu	momheight	hfiacat
0.00	Control	N05265	9	268	male	30	Primary (1-5y)	146.40	Food Se
-1.16	Control	N05265	9	286	male	25	Primary (1-5y)	148.75	Modera
-1.05	Control	N08002	9	264	male	25	Primary (1-5y)	152.15	Food Se
-1.26	Control	N08002	9	252	female	28	Primary (1-5y)	140.25	Food Se
-0.59	Control	N06531	9	336	female	19	Secondary (¿5y)	150.95	Food Se
-0.51	Control	N06531	9	304	male	20	Secondary (¿5y)	154.20	Severely

We can see the covariates used in the prediction:

```
outcome
[1] "whz"
covars
 [1] "tr"             "fracode"        "month"          "aged"
 [5] "sex"            "momage"         "momedu"         "momheight"
 [9] "hfiacat"        "Nlt18"          "Ncomp"          "watmin"
[13] "elec"           "floor"          "walls"          "roof"
[17] "asset_wardrobe" "asset_table"    "asset_chair"    "asset_khat"
[21] "asset_chouki"   "asset_tv"       "asset_refrig"   "asset_bike"
[25] "asset_moto"     "asset_sewmach"  "asset_mobile"
```

Next, we fit a linear model on the full data, with the goal of predicting the weight-for-height Z-score **whz** using all of the available covariate data. Let's try it out:

```
lm_mod <- lm(whz ~ ., data = washb_data)
summary(lm_mod)

Call:
lm(formula = whz ~ ., data = washb_data)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-2.8890 -0.6799 -0.0169  0.6595  3.1005

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -1.89006    1.72022   -1.10   0.2725
trHandwashing                 -0.25276    0.17032   -1.48   0.1385
trNutrition                   -0.09695    0.15696   -0.62   0.5371
trNutrition + WSH             -0.09587    0.16528   -0.58   0.5622
trSanitation                  -0.27702    0.15846   -1.75   0.0811 .
trWSH                         -0.02846    0.15967   -0.18   0.8586
trWater                       -0.07148    0.15813   -0.45   0.6515
fracodeN05160                  0.62355    0.30719    2.03   0.0430 *
fracodeN05265                  0.38762    0.31011    1.25   0.2120
fracodeN05359                  0.10187    0.31329    0.33   0.7452
fracodeN06229                  0.30933    0.29766    1.04   0.2993
fracodeN06453                  0.08066    0.30006    0.27   0.7882
fracodeN06458                  0.43707    0.29970    1.46   0.1454
fracodeN06473                  0.45406    0.30912    1.47   0.1426
fracodeN06479                  0.60994    0.31463    1.94   0.0532 .
fracodeN06489                  0.25923    0.31901    0.81   0.4169
fracodeN06500                  0.07539    0.35794    0.21   0.8333
fracodeN06502                  0.36748    0.30504    1.20   0.2290
fracodeN06505                  0.20038    0.31560    0.63   0.5258
fracodeN06516                  0.55455    0.29807    1.86   0.0635 .
fracodeN06524                  0.49429    0.31423    1.57   0.1164
fracodeN06528                  0.75966    0.31060    2.45   0.0148 *
fracodeN06531                  0.36856    0.30155    1.22   0.2223
fracodeN06862                  0.56932    0.29293    1.94   0.0526 .
fracodeN08002                  0.36779    0.26846    1.37   0.1714
month                          0.17161    0.10865    1.58   0.1149
aged                          -0.00336    0.00112   -3.00   0.0029 **
sexmale                        0.12352    0.09203    1.34   0.1802
momage                        -0.01379    0.00973   -1.42   0.1570
momeduPrimary (1-5y)          -0.13214    0.15225   -0.87   0.3859
momeduSecondary (>5y)          0.12632    0.16041    0.79   0.4314
momheight                      0.00512    0.00919    0.56   0.5776
```

```
hfiacatMildly Food Insecure      0.05804    0.19341    0.30   0.7643
hfiacatModerately Food Insecure -0.01362    0.12887   -0.11   0.9159
hfiacatSeverely Food Insecure   -0.13447    0.25418   -0.53   0.5970
Nlt18                           -0.02557    0.04060   -0.63   0.5291
Ncomp                            0.00179    0.00762    0.23   0.8145
watmin                           0.01347    0.00861    1.57   0.1182
elec                             0.08906    0.10700    0.83   0.4057
floor                           -0.17763    0.17734   -1.00   0.3171
walls                           -0.03001    0.21445   -0.14   0.8888
roof                            -0.03716    0.49214   -0.08   0.9399
asset_wardrobe                  -0.05754    0.13736   -0.42   0.6755
asset_table                     -0.22079    0.12276   -1.80   0.0728 .
asset_chair                      0.28012    0.13750    2.04   0.0422 *
asset_khat                       0.02306    0.11766    0.20   0.8447
asset_chouki                    -0.13943    0.14084   -0.99   0.3227
asset_tv                         0.17723    0.12972    1.37   0.1726
asset_refrig                     0.12613    0.23162    0.54   0.5863
asset_bike                      -0.02568    0.10083   -0.25   0.7990
asset_moto                      -0.32094    0.19944   -1.61   0.1083
asset_sewmach                    0.05090    0.17795    0.29   0.7750
asset_mobile                     0.01420    0.14972    0.09   0.9245
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.984 on 447 degrees of freedom
Multiple R-squared:  0.129, Adjusted R-squared:  0.0277
F-statistic: 1.27 on 52 and 447 DF,  p-value: 0.104
```

We can assess how well the model fits the data by comparing the predictions of the linear model to the true outcomes observed in the data set. This is the well known (and standard) mean squared error. We can extract that from the `lm` model object like so:

```
(err <- mean(resid(lm_mod)^2))
[1] 0.86568
```

The mean squared error is 0.86568. There is an important problem that arises when

we assess the model in this way - that is, we have trained our linear regression model on the full data set and assessed the error on the full data set, using up all of our data. We, of course, are generally not interested in how well the model explains variation in the observed data; rather, we are interested in how the explanation provided by the model generalizes to a target population from which the sample is presumably derived. Having used all of our available data, we cannot honestly evaluate how well the model fits (and thus explains) variation at the population level.

To resolve this issue, cross-validation allows for a particular procedure (e.g., linear regression) to be implemented over subsets of the data, evaluating how well the procedure fits on a testing ("validation") set, thereby providing an honest evaluation of the error.

We can easily add cross-validation to our linear regression procedure using `origami`. First, let us define a new function to perform linear regression on a specific partition of the data (called a "fold"):

```
cv_lm <- function(fold, data, reg_form) {
  # get name and index of outcome variable from regression formula
  out_var <- as.character(unlist(str_split(reg_form, " "))[1])
  out_var_ind <- as.numeric(which(colnames(data) == out_var))

  # split up data into training and validation sets
  train_data <- training(data)
  valid_data <- validation(data)

  # fit linear model on training set and predict on validation set
  mod <- lm(as.formula(reg_form), data = train_data)
  preds <- predict(mod, newdata = valid_data)
  valid_data <- as.data.frame(valid_data)

  # capture results to be returned as output
  out <- list(
    coef = data.frame(t(coef(mod))),
    SE = (preds - valid_data[, out_var_ind])^2
  )
  return(out)
}
```

Our `cv_lm` function is rather simple: we merely split the available data into a training

and validation sets, using the eponymous functions provided in `origami`, fit the linear model on the training set, and evaluate the model on the testing set. This is a simple example of what `origami` considers to be `cv_fun` – functions for using cross-validation to perform a particular routine over an input data set. Having defined such a function, we can simply generate a set of partitions using `origami`'s `make_folds` function, and apply our `cv_lm` function over the resultant `folds` object. Below, we replicate the re-substitution estimate of the error – we did this "by hand" above – using the functions `make_folds` and `cv_lm`.

```
# re-substitution estimate
resub <- make_folds(washb_data, fold_fun = folds_resubstitution)[[1]]
resub_results <- cv_lm(fold = resub, data = washb_data, reg_form = "whz ~ .")
mean(resub_results$SE, na.rm = TRUE)
[1] 0.86568
```

This (nearly) matches the estimate of the error that we obtained above.

We can more honestly evaluate the error by V-fold cross-validation, which partitions the data into v subsets, fitting the model on $v-1$ of the subsets and evaluating on the subset that was held out for testing. This is repeated such that each subset is used for testing. We can easily apply our `cv_lm` function using `origami`'s `cross_validate` (n.b., by default this performs 10-fold cross-validation):

```
# cross-validated estimate
folds <- make_folds(washb_data)
cvlm_results <- cross_validate(
  cv_fun = cv_lm, folds = folds, data = washb_data, reg_form = "whz ~ .",
  use_future = FALSE
)
mean(cvlm_results$SE, na.rm = TRUE)
[1] 1.35
```

Having performed 10-fold cross-validation, we quickly notice that our previous estimate of the model error (by resubstitution) was a bit optimistic. The honest estimate of the error is larger.

0.5.8.2 Cross-validation with random forests

To examine `origami` further, let us return to our example analysis using the WASH data set. Here, we will write a new `cv_fun` type object. As an example, we will use Breiman's `randomForest` (Breiman, 2001):

```
# make sure to load the package!
library(randomForest)

cv_rf <- function(fold, data, reg_form) {
  # get name and index of outcome variable from regression formula
  out_var <- as.character(unlist(str_split(reg_form, " "))[1])
  out_var_ind <- as.numeric(which(colnames(data) == out_var))

  # define training and validation sets based on input object of class "folds"
  train_data <- training(data)
  valid_data <- validation(data)

  # fit Random Forest regression on training set and predict on holdout set
  mod <- randomForest(formula = as.formula(reg_form), data = train_data)
  preds <- predict(mod, newdata = valid_data)
  valid_data <- as.data.frame(valid_data)

  # define output object to be returned as list (for flexibility)
  out <- list(
    coef = data.frame(mod$coefs),
    SE = ((preds - valid_data[, out_var_ind])^2)
  )
  return(out)
}
```

Above, in writing our `cv_rf` function to cross-validate `randomForest`, we used our previous function `cv_lm` as an example. For now, individual `cv_fun` must be written by hand; however, in future releases, a wrapper may be available to support auto-generating `cv_fun`s to be used with `origami`.

Below, we use `cross_validate` to apply our new `cv_rf` function over the `folds` object generated by `make_folds`.

```
# now, let's cross-validate...
folds <- make_folds(washb_data)
cvrf_results <- cross_validate(
  cv_fun = cv_rf, folds = folds, data = washb_data, reg_form = "whz ~ .",
  use_future = FALSE
)
mean(cvrf_results$SE)
[1] 1.0271
```

Using 10-fold cross-validation (the default), we obtain an honest estimate of the prediction error of random forests. From this, we gather that the use of `origami`'s `cross_validate` procedure can be generalized to arbitrary estimation techniques, given availability of an appropriate `cv_fun` function.

0.5.8.3 Cross-validation with arima

Cross-validation can also be used for forecast model selection in a time series setting. Here, the partitioning scheme mirrors the application of the forecasting model: we'll train the data on past observations (either all available or a recent subset), and then use the model fit to predict the next few observations. We consider the `AirPassengers` dataset again, a monthly time series of passenger air traffic in thousands of people.

```
data(AirPassengers)
print(AirPassengers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Suppose we want to pick between two forecasting models with different `arima` configurations. We can do that by evaluating their forecasting performance. First, we set up the appropriate cross-validation scheme for time-series.

```
folds <- make_folds(AirPassengers,
  fold_fun = folds_rolling_origin,
  first_window = 36, validation_size = 24, batch = 10
)

# How many folds where generated?
length(folds)
[1] 9

# Examine the first 2 folds.
folds[[1]]
$v
[1] 1

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36

$validation_set
 [1] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60

attr(,"class")
[1] "fold"
folds[[2]]
$v
[1] 2

$training_set
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46

$validation_set
 [1] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

```
attr(,"class")
[1] "fold"
```

By default, `folds_rolling_origin` will increase the size of the training set by one time point each fold. Had we followed the default option, we would have 85 folds to train! Luckily, we can pass the `batch` as option to `folds_rolling_origin` that tells it to increase the size of the training set by 10 points each iteration. Since we want to forecast the immediate next point, `gap` argument remains the default (0).

```
# make sure to load the package!
library(forecast)

# function to calculate cross-validated squared error
cv_forecasts <- function(fold, data) {
  # Get training and validation data
  train_data <- training(data)
  valid_data <- validation(data)
  valid_size <- length(valid_data)

  train_ts <- ts(log10(train_data), frequency = 12)

  # First arima model
  arima_fit <- arima(train_ts, c(0, 1, 1),
    seasonal = list(
      order = c(0, 1, 1),
      period = 12
    )
  )
  raw_arima_pred <- predict(arima_fit, n.ahead = valid_size)
  arima_pred <- 10^raw_arima_pred$pred
  arima_MSE <- mean((arima_pred - valid_data)^2)

  # Second arima model
  arima_fit2 <- arima(train_ts, c(5, 1, 1),
    seasonal = list(
      order = c(0, 1, 1),
      period = 12
    )
```

```
  )
  raw_arima_pred2 <- predict(arima_fit2, n.ahead = valid_size)
  arima_pred2 <- 10^raw_arima_pred2$pred
  arima_MSE2 <- mean((arima_pred2 - valid_data)^2)

  out <- list(mse = data.frame(
    fold = fold_index(),
    arima = arima_MSE, arima2 = arima_MSE2
  ))
  return(out)
}

mses <- cross_validate(
  cv_fun = cv_forecasts, folds = folds, data = AirPassengers,
  use_future = FALSE
)
mses$mse
  fold   arima  arima2
1    1   68.21  137.28
2    2  319.68  313.15
3    3  578.35  713.36
4    4  428.69  505.31
5    5  407.33  371.27
6    6  281.82  250.99
7    7  827.56  910.12
8    8 2099.59 2213.15
9    9  398.37  293.38
colMeans(mses$mse[, c("arima", "arima2")])
 arima arima2
601.07 634.22
```

The arima model with no AR component seems to be a better fit for this data.

0.5.9 Exercises

0.5.9.1 Review of Key Concepts

1. Compare and contrast V-fold cross-validation with resubstitution cross-validation. What are some of the differences between the two methods? How are they similar? Describe a scenario when you would use one over the other.

2. What are the advantages and disadvantages of v-fold CV relative to:

    a. holdout CV?
    b. leave-one-out CV?

3. Why can't we use V-fold cross-validation for time-series data?

4. Would you use rolling window or origin for non-stationary time-series? Why?

0.5.9.2 The Ideas in Action

1. Let Y be a binary variable with $P(Y = 1 \mid W) = 0.01$. What kind of cross-validation should be use for a rare outcome? How can we do this with the `origami` package?

2. Consider the WASH benefits dataset presented in this chapter. How can we include cluster information into cross-validation? How can we do this with the `origami` package?

0.5.9.3 Advanced Topics

1. Think about a dataset with arbitrary spatial dependence, where we know the extent of dependence, and groups formed by such dependence are clear with no spillover effects. What kind of cross-validation can we use?

2. Continuing on the last problem, what kind of procedure, and cross-validation method, can we use if the spatial dependence is not clearly defined as in the previous problem?

3. Consider a classification problem with a large number of predictors. A statistician proposes the following analysis:

    a. First screen the predictors, leaving only covariates with a strong correlation with the class labels.
    b. Fit some algorithm using only the subset of highly correlated covariates.

c. Use cross-validation to estimate the tuning parameters and the performance of the proposed algorithm.

Is this a correct application of cross-validation? Why?

0.6 The TMLE Framework

Jeremy Coyle

Based on the `tmle3` `R` package.

0.6.1 Learning Objectives

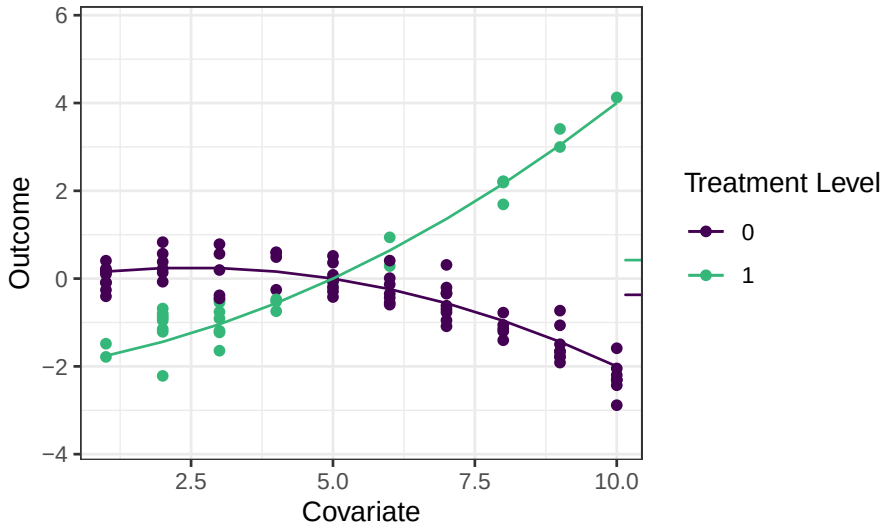
By the end of this chapter, you will be able to

1. Understand why we use TMLE for effect estimation.
2. Use `tmle3` to estimate an Average Treatment Effect (ATE).
3. Understand how to use `tmle3` "Specs" objects.
4. Fit `tmle3` for a custom set of target parameters.
5. Use the delta method to estimate transformations of target parameters.

0.6.2 Introduction

In the previous chapter on `sl3` we learned how to estimate a regression function like $\mathbb{E}[Y \mid X]$ from data. That's an important first step in learning from data, but how can we use this predictive model to estimate statistical and causal effects?

Going back to the roadmap for targeted learning, suppose we'd like to estimate the effect of a treatment variable A on an outcome Y. As discussed, one potential parameter that characterizes that effect is the Average Treatment Effect (ATE), defined as $\psi_0 = \mathbb{E}_W[\mathbb{E}[Y \mid A = 1, W] - \mathbb{E}[Y \mid A = 0, W]]$ and interpreted as the difference in mean outcome under when treatment $A = 1$ and $A = 0$, averaging over the distribution of covariates W. We'll illustrate several potential estimators for this parameter, and motivate the use of the TMLE (targeted maximum likelihood estimation; targeted minimum loss-based estimation) framework, using the following example data:



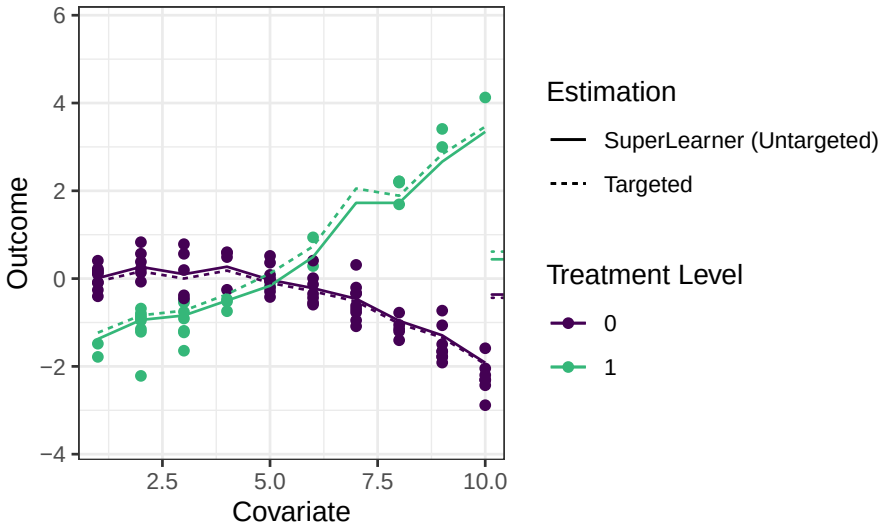
The small ticks on the right indicate the mean outcomes (averaging over W) under $A = 1$ and $A = 0$ respectively, so their difference is the quantity we'd like to estimate.

While we hope to motivate the application of TMLE in this chapter, we refer the interested reader to the two Targeted Learning books and associated works for full technical details.

0.6.3 Substitution Estimators

We can use `sl3` to fit a Super Learner or other regression model to estimate the outcome regression function $\mathbb{E}_0[Y \mid A, W]$, which we often refer to as $\overline{Q}_0(A, W)$ and whose estimate we denote $\overline{Q}_n(A, W)$. To construct an estimate of the ATE ψ_n, we need only "plug-in" the estimates of $\overline{Q}_n(A, W)$, evaluated at the two intervention contrasts, to the corresponding ATE "plug-in" formula: $\psi_n = \frac{1}{n} \sum(\overline{Q}_n(1, W) - \overline{Q}_n(0, W))$. This kind of estimator is called a *plug-in* or *substitution* estimator, since accurate estimates ψ_n of the parameter ψ_0 may be obtained by substituting estimates $\overline{Q}_n(A, W)$ for the relevant regression functions $\overline{Q}_0(A, W)$ themselves.

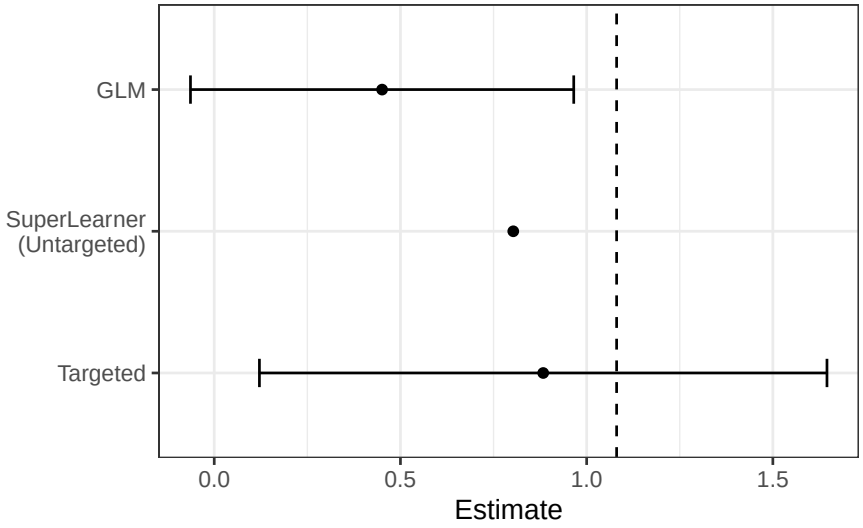
Applying `sl3` to estimate the outcome regression in our example, we can see that the ensemble machine learning predictions fit the data quite well:



The solid lines indicate the `sl3` estimate of the regression function, with the dotted lines indicating the `tmle3` updates (described below).

While substitution estimators are intuitive, naively using this approach with a Super Learner estimate of $\overline{Q}_0(A, W)$ has several limitations. First, Super Learner is selecting learner weights to minimize risk across the entire regression function, instead of "targeting" the ATE parameter we hope to estimate, leading to biased estimation. That is, `sl3` is trying to do well on the full regression curve on the left, instead of focusing on the small ticks on the right. What's more, the sampling distribution of this approach is not asymptotically linear, and therefore inference is not possible.

We can see these limitations illustrated in the estimates generated for the example data:



We see that Super Learner, estimates the true parameter value (indicated by the dashed vertical line) more accurately than GLM. However, it is still less accurate than TMLE, and valid inference is not possible. In contrast, TMLE achieves a less biased estimator and valid inference.

0.6.4 Targeted Maximum Likelihood Estimation

TMLE takes an initial estimate $\overline{Q}_n(A, W)$ as well as an estimate of the propensity score $g_n(A \mid W) = \mathbb{P}(A = 1 \mid W)$ and produces an updated estimate $\overline{Q}_n^\star(A, W)$ that is "targeted" to the parameter of interest. TMLE keeps the benefits of substitution estimators (it is one), but augments the original, potentially erratic estimates to *correct for bias* while also resulting in an *asymptotically linear* (and thus normally distributed) estimator that accommodates inference via asymptotically consistent Wald-style confidence intervals.

0.6.4.1 TMLE Updates

There are different types of TMLEs (and, sometimes, multiple for the same set of target parameters) – below, we give an example of the algorithm for TML estimation of the ATE. $\overline{Q}_n^\star(A, W)$ is the TMLE-augmented estimate $f(\overline{Q}_n^\star(A, W)) = f(\overline{Q}_n(A, W)) + \epsilon \cdot H_n(A, W)$, where $f(\cdot)$ is the appropriate link function (e.g., $\text{logit}(x) = \log(x/(1-x))$), and an estimate ϵ_n of the coefficient ϵ of the "clever covariate" $H_n(A, W)$ is computed. The form of the covariate $H_n(A, W)$ differs across

target parameters; in this case of the ATE, it is $H_n(A,W) = \frac{A}{g_n(A|W)} - \frac{1-A}{1-g_n(A,W)}$, with $g_n(A,W) = \mathbb{P}(A = 1 \mid W)$ being the estimated propensity score, so the estimator depends both on the initial fit (by `sl3`) of the outcome regression ($\overline{Q}_n$) and of the propensity score (g_n).

There are several robust augmentations that are used across the `tlverse`, including the use of an additional layer of cross-validation to avoid over-fitting bias (i.e., CV-TMLE) as well as approaches for more consistently estimating several parameters simultaneously (e.g., the points on a survival curve).

0.6.4.2 Statistical Inference

Since TMLE yields an **asymptotically linear** estimator, obtaining statistical inference is very convenient. Each TML estimator has a corresponding **(efficient) influence function** (often, "EIF", for short) that describes the asymptotic distribution of the estimator. By using the estimated EIF, Wald-style inference (asymptotically correct confidence intervals) can be constructed simply by plugging into the form of the EIF our initial estimates $\overline{Q}_n$ and g_n, then computing the sample standard error.

The following sections describe both a simple and more detailed way of specifying and estimating a TMLE in the `tlverse`. In designing `tmle3`, we sought to replicate as closely as possible the very general estimation framework of TMLE, and so each theoretical object relevant to TMLE is encoded in a corresponding software object/method. First, we will present the simple application of `tmle3` to the WASH Benefits example, and then go on to describe the underlying objects in greater detail.

0.6.5 Easy-Bake Example: `tmle3` for ATE

We'll illustrate the most basic use of TMLE using the WASH Benefits data introduced earlier and estimating an average treatment effect.

0.6.5.1 Load the Data

We'll use the same WASH Benefits data as the earlier chapters:

```
library(data.table)
library(dplyr)
library(tmle3)
library(sl3)
```

```
washb_data <- fread(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-data/master/",
    "wash-benefits/washb_data.csv"
  ),
  stringsAsFactors = TRUE
)
```

0.6.5.2 Define the variable roles

We'll use the common W (covariates), A (treatment/intervention), Y (outcome) data structure. `tmle3` needs to know what variables in the dataset correspond to each of these roles. We use a list of character vectors to tell it. We call this a "Node List" as it corresponds to the nodes in a Directed Acyclic Graph (DAG), a way of displaying causal relationships between variables.

```
node_list <- list(
  W = c(
    "month", "aged", "sex", "momage", "momedu",
    "momheight", "hfiacat", "Nlt18", "Ncomp", "watmin",
    "elec", "floor", "walls", "roof", "asset_wardrobe",
    "asset_table", "asset_chair", "asset_khat",
    "asset_chouki", "asset_tv", "asset_refrig",
    "asset_bike", "asset_moto", "asset_sewmach",
    "asset_mobile"
  ),
  A = "tr",
  Y = "whz"
)
```

0.6.5.3 Handle Missingness

Currently, missingness in `tmle3` is handled in a fairly simple way:

- Missing covariates are median- (for continuous) or mode- (for discrete) imputed, and additional covariates indicating imputation are generated, just as described in the `sl3` chapter.

- Missing treatment variables are excluded – such observations are dropped.
- Missing outcomes are efficiently handled by the automatic calculation (and incorporation into estimators) of *inverse probability of censoring weights* (IPCW); this is also known as IPCW-TMLE and may be thought of as a joint intervention to remove missingness and is analogous to the procedure used with classical inverse probability weighted estimators.

These steps are implemented in the `process_missing` function in `tmle3`:

```
processed <- process_missing(washb_data, node_list)
washb_data <- processed$data
node_list <- processed$node_list
```

0.6.5.4 Create a "Spec" Object

`tmle3` is general, and allows most components of the TMLE procedure to be specified in a modular way. However, most users will not be interested in manually specifying all of these components. Therefore, `tmle3` implements a `tmle3_Spec` object that bundles a set of components into a *specification* ("Spec") that, with minimal additional detail, can be run to fit a TMLE.

We'll start with using one of the specs, and then work our way down into the internals of `tmle3`.

```
ate_spec <- tmle_ATE(
  treatment_level = "Nutrition + WSH",
  control_level = "Control"
)
```

0.6.5.5 Define the learners

Currently, the only other thing a user must define are the `sl3` learners used to estimate the relevant factors of the likelihood: Q and g.

This takes the form of a list of `sl3` learners, one for each likelihood factor to be estimated with `sl3`:

```
# choose base learners
lrnr_mean <- make_learner(Lrnr_mean)
lrnr_rf <- make_learner(Lrnr_ranger)

# define metalearners appropriate to data types
ls_metalearner <- make_learner(Lrnr_nnls)
mn_metalearner <- make_learner(
  Lrnr_solnp, metalearner_linear_multinomial,
  loss_loglik_multinomial
)
sl_Y <- Lrnr_sl$new(
  learners = list(lrnr_mean, lrnr_rf),
  metalearner = ls_metalearner
)
sl_A <- Lrnr_sl$new(
  learners = list(lrnr_mean, lrnr_rf),
  metalearner = mn_metalearner
)
learner_list <- list(A = sl_A, Y = sl_Y)
```

Here, we use a Super Learner as defined in the previous chapter. In the future, we plan to include reasonable defaults learners.

0.6.5.6 Fit the TMLE

We now have everything we need to fit the tmle using `tmle3`:

```
tmle_fit <- tmle3(ate_spec, washb_data, node_list, learner_list)
print(tmle_fit)
A tmle3_Fit that took 1 step(s)
   type                                      param   init_est tmle_est       se
1:  ATE ATE[Y_{A=Nutrition + WSH}-Y_{A=Control}] -0.0031611 0.010044 0.050853
       lower   upper psi_transformed lower_transformed upper_transformed
1: -0.089626 0.10971        0.010044         -0.089626           0.10971
```

0.6.5.7 Evaluate the Estimates

We can see the summary results by printing the fit object. Alternatively, we can extra results from the summary by indexing into it:

```
estimates <- tmle_fit$summary$psi_transformed
print(estimates)
[1] 0.010044
```

0.6.6 tmle3 Components

Now that we've successfully used a spec to obtain a TML estimate, let's look under the hood at the components. The spec has a number of functions that generate the objects necessary to define and fit a TMLE.

0.6.6.1 tmle3_task

First is, a `tmle3_Task`, analogous to an `sl3_Task`, containing the data we're fitting the TMLE to, as well as an NPSEM generated from the `node_list` defined above, describing the variables and their relationships.

```
tmle_task <- ate_spec$make_tmle_task(washb_data, node_list)
```

```
tmle_task$npsem
$W
tmle3_Node: W
    Variables: month, aged, sex, momedu, hfiacat, Nlt18, Ncomp, watmin, elec, floor, wall
    Parents:

$A
tmle3_Node: A
    Variables: tr
    Parents: W

$Y
tmle3_Node: Y
    Variables: whz
    Parents: A, W
```

0.6.6.2 Initial Likelihood

Next, is an object representing the likelihood, factorized according to the NPSEM described above:

```
initial_likelihood <- ate_spec$make_initial_likelihood(
  tmle_task,
  learner_list
)
print(initial_likelihood)
W: Lf_emp
A: LF_fit
Y: LF_fit
```

These components of the likelihood indicate how the factors were estimated: the marginal distribution of W was estimated using NP-MLE, and the conditional distributions of A and Y were estimated using `sl3` fits (as defined with the `learner_list`) above.

We can use this in tandem with the `tmle_task` object to obtain likelihood estimates for each observation:

```
initial_likelihood$get_likelihoods(tmle_task)
                W       A        Y
   1: 0.00021299 0.34702 -0.32696
   2: 0.00021299 0.37305 -0.88218
   3: 0.00021299 0.34685 -0.79300
   4: 0.00021299 0.33625 -0.89157
   5: 0.00021299 0.34098 -0.63477
  ---
4691: 0.00021299 0.24334 -0.61095
4692: 0.00021299 0.24620 -0.21534
4693: 0.00021299 0.22401 -0.79223
4694: 0.00021299 0.27641 -0.94319
4695: 0.00021299 0.20158 -1.08201
```

0.6.6.3 Targeted Likelihood (updater)

We also need to define a "Targeted Likelihood" object. This is a special type of likelihood that is able to be updated using an `tmle3_Update` object. This object defines the update strategy (e.g., submodel, loss function, CV-TMLE or not).

```
targeted_likelihood <- Targeted_Likelihood$new(initial_likelihood)
```

When constructing the targeted likelihood, you can specify different update options. See the documentation for `tmle3_Update` for details of the different options. For example, you can disable CV-TMLE (the default in `tmle3`) as follows:

```
targeted_likelihood_no_cv <-
  Targeted_Likelihood$new(initial_likelihood,
    updater = list(cvtmle = FALSE)
  )
```

0.6.6.4 Parameter Mapping

Finally, we need to define the parameters of interest. Here, the spec defines a single parameter, the ATE. In the next section, we'll see how to add additional parameters.

```
tmle_params <- ate_spec$make_params(tmle_task, targeted_likelihood)
print(tmle_params)
[[1]]
Param_ATE: ATE[Y_{A=Nutrition + WSH}-Y_{A=Control}]
```

0.6.6.5 Putting it all together

Having used the spec to manually generate all these components, we can now manually fit a `tmle3`:

```
tmle_fit_manual <- fit_tmle3(
  tmle_task, targeted_likelihood, tmle_params,
  targeted_likelihood$updater
)
print(tmle_fit_manual)
```

```
A tmle3_Fit that took 1 step(s)
   type                                      param   init_est tmle_est       se
1:  ATE ATE[Y_{A=Nutrition + WSH}-Y_{A=Control}] -0.0062324 0.017515 0.050591
       lower   upper psi_transformed lower_transformed upper_transformed
1: -0.081641 0.11667        0.017515         -0.081641           0.11667
```

The result is equivalent to fitting using the `tmle3` function as above.

0.6.7 Fitting `tmle3` with multiple parameters

Above, we fit a `tmle3` with just one parameter. `tmle3` also supports fitting multiple parameters simultaneously. To illustrate this, we'll use the `tmle_TSM_all` spec:

```
tsm_spec <- tmle_TSM_all()
targeted_likelihood <- Targeted_Likelihood$new(initial_likelihood)
all_tsm_params <- tsm_spec$make_params(tmle_task, targeted_likelihood)
print(all_tsm_params)
[[1]]
Param_TSM: E[Y_{A=Control}]

[[2]]
Param_TSM: E[Y_{A=Handwashing}]

[[3]]
Param_TSM: E[Y_{A=Nutrition}]

[[4]]
Param_TSM: E[Y_{A=Nutrition + WSH}]

[[5]]
Param_TSM: E[Y_{A=Sanitation}]

[[6]]
Param_TSM: E[Y_{A=WSH}]

[[7]]
Param_TSM: E[Y_{A=Water}]
```

This spec generates a Treatment Specific Mean (TSM) for each level of the exposure variable. Note that we must first generate a new targeted likelihood, as the old one was targeted to the ATE. However, we can recycle the initial likelihood we fit above, saving us a super learner step.

0.6.7.1 Delta Method

We can also define parameters based on Delta Method Transformations of other parameters. For instance, we can estimate a ATE using the delta method and two of the above TSM parameters:

```
ate_param <- define_param(
  Param_delta, targeted_likelihood,
  delta_param_ATE,
  list(all_tsm_params[[1]], all_tsm_params[[4]])
)
print(ate_param)
Param_delta: E[Y_{A=Nutrition + WSH}] - E[Y_{A=Control}]
```

This can similarly be used to estimate other derived parameters like Relative Risks, and Population Attributable Risks

0.6.7.2 Fit

We can now fit a TMLE simultaneously for all TSM parameters, as well as the above defined ATE parameter

```
all_params <- c(all_tsm_params, ate_param)

tmle_fit_multiparam <- fit_tmle3(
  tmle_task, targeted_likelihood, all_params,
  targeted_likelihood$updater
)

print(tmle_fit_multiparam)
A tmle3_Fit that took 1 step(s)
   type                                    param   init_est tmle_est
1:  TSM                    E[Y_{A=Control}] -0.5953314 -0.61981
```

```
2:  TSM                          E[Y_{A=Handwashing}] -0.6179897 -0.66114
3:  TSM                            E[Y_{A=Nutrition}] -0.6119870 -0.60338
4:  TSM                     E[Y_{A=Nutrition + WSH}] -0.6015639 -0.60250
5:  TSM                          E[Y_{A=Sanitation}] -0.5866311 -0.58147
6:  TSM                                E[Y_{A=WSH}] -0.5213051 -0.45027
7:  TSM                              E[Y_{A=Water}] -0.5653576 -0.53554
8:  ATE E[Y_{A=Nutrition + WSH}] - E[Y_{A=Control}] -0.0062324  0.01731
         se     lower    upper psi_transformed lower_transformed
1: 0.030069 -0.678746 -0.56088        -0.61981         -0.678746
2: 0.041821 -0.743111 -0.57917        -0.66114         -0.743111
3: 0.041553 -0.684825 -0.52194        -0.60338         -0.684825
4: 0.040925 -0.682712 -0.52229        -0.60250         -0.682712
5: 0.042313 -0.664402 -0.49854        -0.58147         -0.664402
6: 0.045216 -0.538891 -0.36165        -0.45027         -0.538891
7: 0.039290 -0.612551 -0.45854        -0.53554         -0.612551
8: 0.050596 -0.081857  0.11648         0.01731         -0.081857
   upper_transformed
1:          -0.56088
2:          -0.57917
3:          -0.52194
4:          -0.52229
5:          -0.49854
6:          -0.36165
7:          -0.45854
8:           0.11648
```

0.6.8 Exercises

0.6.8.1 Estimation of the ATE with `tmle3`

Follow the steps below to estimate an average treatment effect using data from the Collaborative Perinatal Project (CPP), available in the `sl3` package. To simplify this example, we define a binary intervention variable, `parity01` – an indicator of having one or more children before the current child and a binary outcome, `haz01` – an indicator of having an above average height for age.

```
# load the data set
data(cpp)
cpp <- cpp %>%
  as_tibble() %>%
  dplyr::filter(!is.na(haz)) %>%
  mutate(
    parity01 = as.numeric(parity > 0),
    haz01 = as.numeric(haz > 0)
  )
```

1. Define the variable roles (W, A, Y) by creating a list of these nodes. Include the following baseline covariates in W: `apgar1`, `apgar5`, `gagebrth`, `mage`, `meducyrs`, `sexn`. Both A and Y are specified above. The missingness in the data (specifically, the missingness in the columns that are specified in the node list) will need to be taking care of. The `process_missing` function can be used to accomplish this, like the `washb_data` example above.
2. Define a `tmle3_Spec` object for the ATE, `tmle_ATE()`.
3. Using the same base learning libraries defined above, specify `sl3` base learners for estimation of $\overline{Q}_0 = \mathbb{E}_0(Y \mid A, Y)$ and $g_0 = \mathbb{P}(A = 1 \mid W)$.
4. Define the metalearner like below.

```
metalearner <- make_learner(
  Lrnr_solnp,
  loss_function = loss_loglik_binomial,
  learner_function = metalearner_logistic_binomial
)
```

5. Define one super learner for estimating $\overline{Q}_0$ and another for estimating g_0. Use the metalearner above for both super learners.
6. Create a list of the two super learners defined in the step above and call this object `learner_list`. The list names should be `A` (defining the super learner for estimation of g_0) and `Y` (defining the super learner for estimation of $\overline{Q}_0$).
7. Fit the TMLE with the `tmle3` function by specifying (1) the `tmle3_Spec`, which we defined in Step 2; (2) the data; (3) the list of nodes, which we specified in Step 1; and (4) the list of super learners for estimation of g_0 and $\overline{Q}_0$, which we defined in Step 6. *Note*: Like before, you will need to explicitly make a copy of the data (to work around `data.table` optimizations), e.g., (`cpp2 <- data.table::copy(cpp)`), then use the `cpp2` data going forward.

0.6.8.2 Estimation of Strata-Specific ATEs with `tmle3`

For this exercise, we will work with a random sample of 5,000 patients who participated in the International Stroke Trial (IST). This data is described in the Chapter 3.2 of the `tlverse` handbook. We included the data below and a summarized description that is relevant for this exercise.

The outcome, Y, indicates recurrent ischemic stroke within 14 days after randomization (`DRSISC`); the treatment of interest, A, is the randomized aspirin vs. no aspirin treatment allocation (`RXASP` in `ist`); and the adjustment set, W, consists simply of other variables measured at baseline. In this data, the outcome is occasionally missing, but there is no need to create a variable indicating this missingness (such as Δ) for analyses in the `tlverse`, since the missingness is automatically detected when `NA` are present in the outcome. Covariates with missing values (`RATRIAL`, `RASP3` and `RHEP24`) have already been imputed. Additional covariates were created (`MISSING_RATRIAL_RASP3` and `MISSING_RHEP24`), which indicate whether or not the covariate was imputed. The missingness was identical for `RATRIAL` and `RASP3`, which is why only one covariate indicating imputation for these two covariates was created.

1. Estimate the average effect of randomized asprin treatment (`RXASP` $= 1$) on recurrent ischemic stroke. Even though the missingness mechanism on Y, Δ, does not need to be specified in the node list, it does still need to be accounted for in the TMLE. In other words, for this estimation problem, Δ is a relevant factor of the likelihood. Thus, when defining the list of `sl3` learners for each likelihood factor, be sure to include a list of learners for estimation of Δ, say `sl_Delta`, and specify this in the learner list, like so `learner_list <- list(A = sl_A, delta_Y = sl_Delta, Y = sl_Y)`.
2. Recall that this RCT was conducted internationally. Suposse there is concern that the dose of asprin may have varied across geographical regions, and an average across all geographical regions may not be warranted. Calculate the strata specific ATEs according to geographical region (`REGION`).

```
ist_data <- fread(
  paste0(
    "https://raw.githubusercontent.com/tlverse/deming2019-workshop/",
    "master/data/ist_sample.csv"
  )
)
```

0.6.9 Summary

`tmle3` is a general purpose framework for generating TML estimates. The easiest way to use it is to use a predefined spec, allowing you to just fill in the blanks for the data, variable roles, and `sl3` learners. However, digging under the hood allows users to specify a wide range of TMLEs. In the next sections, we'll see how this framework can be used to estimate advanced parameters such as optimal treatments and stochastic shift interventions.

0.7 Causal Mediation Analysis

Nima Hejazi

Based on the `tmle3mediate` `R` package by *Nima Hejazi, James Duncan, and David McCoy.*

Updated: 2021-04-19

0.7.1 Introduction to Causal Mediation Analysis

A treatment often affects an outcome indirectly, through a particular pathway, by its effect on *intermediate variables* (mediators). Causal mediation analysis concerns the construction and evaluation of these *indirect effects* and their complementary *direct effects*. Generally, the indirect effect (IE) of a treatment on an outcome is the portion of the total effect that is found to work *through* mediator variables, while the direct effect often encompasses all other components of the total effect, including both the effect of the treatment on the outcome *and* the effect through all paths not explicitly involving the mediators). Identifying and quantifying the mechanisms underlying causal effects is an increasingly desirable endeavor in public health, medicine, and the social sciences, as such mechanistic knowledge improves understanding of both *why* and *how* treatments may be effective.

While the study of mediation analysis may be traced back quite far, the field only came into its modern form with the identification and careful study of the natural direct and indirect effects [Robins and Greenland (1992); pearl2001direct]. The natural direct effect (NDE) and the natural indirect effect (NIE) are based on a decomposition of the average treatment effect (ATE) in the presence of mediators

(VanderWeele, 2015); requisite theory for the construction of efficient estimators of these quantities only receiving attention relatively recently (Tchetgen Tchetgen and Shpitser, 2012).

0.7.2 Data Structure and Notation

Consider n observed units $O_1, \ldots, O_n$, where each observed data random variable takes the form $O = (W, A, Z, Y)$, for a vector of observed covariates W, a binary or continuous treatment A, possibly multivariate mediators Z, and a binary or continuous outcome Y. To avoid undue assumptions, we assume only that $O \sim \mathcal{P} \in \mathcal{M}$ where $\mathcal{M}$ is the nonparametric statistical model defined as all continuous densities on O with respect to an arbitrary dominating measure.

We formalize the definition of our counterfactual variables using the following nonparametric structural equation model (NPSEM):

$$W = f_W(U_W) \tag{0.3}$$
$$A = f_A(W, U_A) \tag{0.4}$$
$$Z = f_Z(W, A, U_M) \tag{0.5}$$
$$Y = f_Y(W, A, Z, U_Y). \tag{0.6}$$

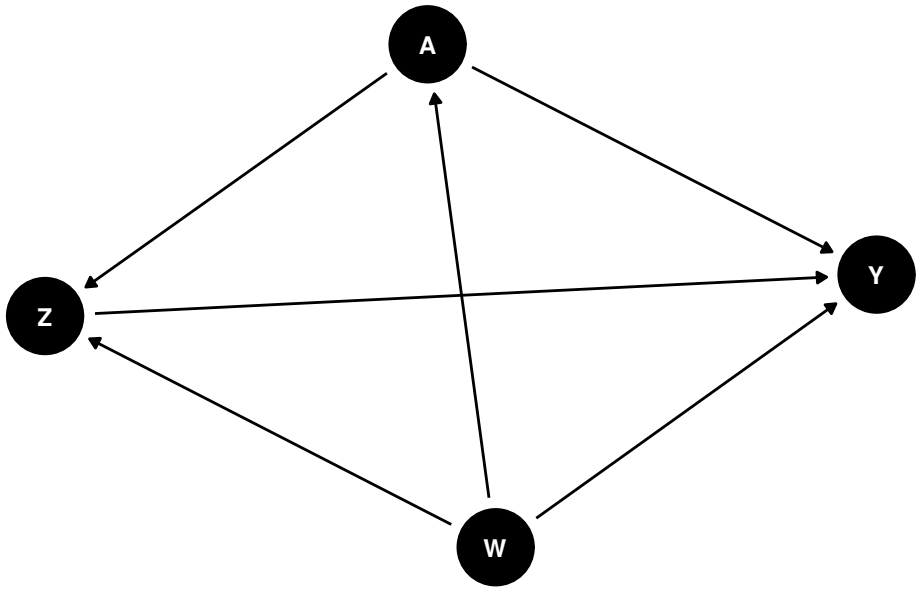
This set of equations represents a mechanistic model generating the observed data O; furthermore, the NPSEM encodes several fundamental assumptions. Firstly, there is an implicit temporal ordering: W occurs first, depending only on exogenous factors U_W; A happens next, based on both W and exogenous factors U_A; then come the mediators Z, which depend on A, W, and another set of exogenous factors U_Z; and finally appears the outcome Y. We assume neither access to the set of exogenous factors $\{U_W, U_A, U_Z, U_Y\}$ nor knowledge of the forms of the deterministic generating functions $\{f_W, f_A, f_Z, f_Y\}$. The NPSEM corresponds to the following DAG:

```
library(dagitty)
library(ggdag)

# make DAG by specifying dependence structure
dag <- dagitty(
  "dag {
    W -> A
    W -> Z
    W -> Y
```

```
    A -> Z
    A -> Y
    Z -> Y
    W -> A -> Y
    W -> A -> Z -> Y
  }"
)
exposures(dag) <- c("A")
outcomes(dag) <- c("Y")
tidy_dag <- tidy_dagitty(dag)

# visualize DAG
ggdag(tidy_dag) +
  theme_dag()
```



The likelihood of the data O admits a factorization, wherein, for p_0^O, the density of O with respect to the product measure, the density evaluated on a particular observation o may be a written

$$p_0^O(x) = q_{0,Y}^O(y \mid Z = zA = a, W = w) q_{0,Z}^O(z \mid A = a, W = w) q_{0,A}^O(a \mid W = w) q_{0,W}^O(w), \tag{0.7}$$

where $q_{0,Y}$ is the conditional density of Y given (Z, A, W), $q_{0,Z}$ is the conditional

density of Z given (A, W), $q_{0,A}$ is the conditional density of A given W, and $q_{0,W}$ is the density of W. Further, for ease of notation, we let $\bar{Q}_Y(Z, A, W) = \mathbb{E}[Y \mid Z, A, W]$, $\bar{Q}_Z(A, W) = \mathbb{P}[Z \mid A, W]$ (later re-parametrized to $e(A \mid Z, W) = \mathbb{P}(A \mid Z, W)$), $g(A \mid W) = \mathbb{P}(A \mid W)$, and q_W the marginal distribution of W.

Finally, note that we have explicitly excluded potential confounders of the mediator-outcome relationship affected by exposure (i.e., variables affected by A and affecting both Z and Y). Mediation analysis in the presence of such variables is exceptionally challenging (Avin et al., 2005); thus, most efforts to develop definitions of causal direct and indirect effects explicitly disallowed such a form of confounding. While we will not discuss the matter here, the interested reader may consult recent advances in the vast literature on causal mediation analysis, among them Díaz et al. (2020) and Hejazi et al. (2021).

0.7.3 Decomposing the Average Treatment Effect

The natural direct and indirect effects arise from a decomposition of the ATE:

$$\mathbb{E}[Y(1) - Y(0)] = \underbrace{\mathbb{E}[Y(1, Z(0)) - Y(0, Z(0))]}_{NDE} + \underbrace{\mathbb{E}[Y(1, Z(1)) - Y(1, Z(0))]}_{NIE}.$$

In particular, the natural indirect effect (NIE) measures the effect of the treatment $A \in \{0, 1\}$ on the outcome Y through the mediators Z, while the natural direct effect (NDE) measures the effect of the treatment on the outcome *through all other paths*. Identification of the natural direct and indirect effects requires the following non-testable causal assumptions:

1. *Exchangeability*: $Y(a, z) \perp\!\!\!\perp (A, Z) \mid W$, which further implies $\mathbb{E}\{Y(a, z) \mid A = a, W = w, Z = z\} = \mathbb{E}\{Y(a, z) \mid W = w\}$. This is a special case of the randomization assumption, extended to observational studies with mediators.
2. *Treatment positivity*: For any $a \in \mathcal{A}$ and $w \in \mathcal{W}$, $\xi < g(a \mid w) < 1 - \xi$, for $\xi > 0$. This mirrors the assumption required for static intervention, discussed previously.
3. *Mediator positivity*: For any $z \in \mathcal{Z}$, $a \in \mathcal{A}$, and $w \in \mathcal{W}$, $\epsilon < g(a \mid w)$, for $\epsilon > 0$. This only requires that the conditional density of the mediators be bounded away from zero for all (z, a, w) in their joint support $\mathcal{Z} \times \mathcal{A} \times \mathcal{W}$.
4. *Cross-world counterfactual independence*: For all $a \neq a'$, both contained in $\mathcal{A}$ and $z \in \mathcal{Z}$, $Y(a', z)$ is independent of $Z(a)$, given W. That is, the counterfactual outcome under the treatment contrast $a' \in \mathcal{A}$ and the counterfactual mediator $Z(a) in \mathcal{Z}$ (under a different contrast $a \in \mathcal{A}$) are independent. Note that the counterfactual outcome and mediator are defined under differing contrasts, hence the "cross-world" designation.

We note that many attempts have been made to weaken the last assumption, that of cross-world counterfactual independence, including work by Petersen et al. (2006) and Imai et al. (2010); however, importantly, Robins and Richardson (2010) established that this assumption cannot be satisfied in randomized experiments. Thus, the natural direct and indirect effects are not identifiable in randomized experiments, calling into question their utility. Despite this significant limitation, we will turn to considering estimation of the statistical functionals corresponding to these effects in observational studies.

0.7.4 The Natural Direct Effect

The NDE is defined as

$$\Psi_{NDE} = \mathbb{E}[Y(1, Z(0)) - Y(0, Z(0))] \overset{\text{rand.}}{=} \sum_w \sum_z [\underbrace{\mathbb{E}(Y \mid A = 1, z, w)}_{\bar{Q}_Y(A=1,z,w)} - \underbrace{\mathbb{E}(Y \mid A = 0, z, w)}_{\bar{Q}_Y(A=0,z,w)}] \times \underbrace{p(z \mid A = 0, w)}_{Q_Z(0,w))} \underbrace{p(w)}_{q_W}$$

where the likelihood factors $p(z \mid A = 0, w)$ and $p(w)$ (among other conditional densities) arise from a factorization of the joint likelihood:

$$p(w, a, z, y) = \underbrace{p(y \mid w, a, z)}_{Q_Y(A,W,Z)} \underbrace{p(z \mid w, a)}_{Q_Z(Z|A,W)} \underbrace{p(a \mid w)}_{g(A|W)} \underbrace{p(w)}_{Q_W}.$$

The process of estimating the NDE begins by constructing $\bar{Q}_{Y,n}$, an estimate of the outcome mechanism $\bar{Q}_Y(Z, A, W) = \mathbb{E}\{Y \mid Z, A, W\}$ (i.e., the conditional mean of Y, given Z, A, and W). With an estimate of this conditional expectation in hand, predictions of the counterfactual quantities $\bar{Q}_Y(Z, 1, W)$ (setting $A = 1$) and, likewise, $\bar{Q}_Y(Z, 0, W)$ (setting $A = 0$) can readily be obtained. We denote the difference of these counterfactual quantities $\bar{Q}_{\text{diff}}$, i.e., $\bar{Q}_{\text{diff}} = \bar{Q}_Y(Z, 1, W) - \bar{Q}_Y(Z, 0, W)$. $\bar{Q}_{\text{diff}}$ represents the difference in the conditional mean of Y attributable to changes in A while keeping Z and W at their *natural* (that is, observed) values.

The estimation procedure treats $\bar{Q}_{\text{diff}}$ itself as a nuisance parameter, regressing its estimate $\bar{Q}_{\text{diff},n}$ on W, among control observations only (i.e., those for whom $A = 0$ is observed); the goal of this step is to remove part of the marginal impact of Z on $\bar{Q}_{\text{diff}}$, since W is a parent of Z. Regressing this difference on W among the controls recovers the expected $\bar{Q}_{\text{diff}}$, had all individuals been set to the control condition $A = 0$. Any residual additive effect of Z on $\bar{Q}_{\text{diff}}$ is removed during the TML estimation step using the auxiliary (or "clever") covariate, which accounts for the mediators Z. This auxiliary covariate takes the form

$$C_Y(Q_Z, g)(O) = \left\{ \frac{\mathbb{I}(A = 1)}{g(1 \mid W)} \frac{Q_Z(Z \mid 0, W)}{Q_Z(Z \mid 1, W)} - \frac{\mathbb{I}(A = 0)}{g(0 \mid W)} \right\}.$$

Breaking this down, $\frac{\mathbb{I}(A=1)}{g(1|W)}$ is the inverse propensity score weight for $A = 1$ and, likewise, $\frac{\mathbb{I}(A=0)}{g(0|W)}$ is the inverse propensity score weight for $A = 0$. The middle term is the ratio of the conditional densities of the mediator under the control ($A = 0$) and treatment ($A = 1$) conditions.

This subtle appearance of a ratio of conditional densities is concerning – unfortunately, tools to estimate such quantities are sparse in the statistics literature, and the problem is still more complicated (and computationally taxing) when Z is high-dimensional. As only the ratio of these conditional densities is required, a convenient re-parametrization may be achieved, that is,

$$\frac{p(A = 0 \mid Z, W)g(0 \mid W)}{p(A = 1 \mid Z, W)g(1 \mid W)}.$$

Going forward, we will denote this re-parameterized conditional probability $e(A \mid Z, W) := p(A \mid Z, W)$. Similar re-parameterizations have been used in Zheng and van der Laan (2012) and Tchetgen Tchetgen (2013). This is particularly useful since this reformulation reduces the problem to one concerning only the estimation of conditional means, opening the door to the use of a wide range of machine learning algorithms (e.g., most of those in `sl3`).

Underneath the hood, the counterfactual outcome difference $\bar{Q}_{\text{diff}}$ and $e(A \mid Z, W)$, the conditional probability of A given Z and W, are used in constructing the auxiliary covariate for TML estimation. These nuisance parameters play an important role in the bias-correcting update step of the TMLE procedure.

0.7.5 The Natural Indirect Effect

Derivation and estimation of the NIE is analogous to that of the NDE. The NIE is the effect of A on Y *only through the mediator(s)* Z. This quantity – known as the natural indirect effect $\mathbb{E}(Y(Z(1), 1) - \mathbb{E}(Y(Z(0), 1)$ – corresponds to the difference of the conditional expectation of Y given $A = 1$ and $Z(1)$ (the values the mediator would take under $A = 1$) and the conditional expectation of Y given $A = 1$ and $Z(0)$ (the values the mediator would take under $A = 0$).

As with the NDE, the re-parameterization trick can be used to estimate $\mathbb{E}(A \mid Z, W)$, avoiding estimation of a possibly multivariate conditional density. However, in this case, the mediated mean outcome difference, denoted $\Psi_Z(Q)$, is instead estimated as follows

$$\Psi_{NIE}(Q) = \mathbb{E}_{QZ}(\Psi_{NIE,Z}(Q)(1, W) - \Psi_{NIE,Z}(Q)(0, W))$$

Here, $\bar{Q}_Y(Z, 1, W)$ (the predicted values for Y given Z and W when $A = 1$) is

regressed on W, among the treated units (for whom $A = 1$ is observed) to obtain the conditional mean $\Psi_{NIE,Z}(Q)(1, W)$. Performing the same procedure, but now regressing $\bar{Q}_Y(Z, 1, W)$ on W among the control units (for whom $A = 0$ is observed) yields $\Psi_{NIE,Z}(Q)(0, W)$. The difference of these two estimates is the NIE and can be thought of as the additive marginal effect of treatment on the conditional expectation of Y given W, $A = 1$, Z through its effects on Z. So, in the case of the NIE, our estimate ψ_n is slightly different, but the same quantity $e(A \mid Z, W)$ comes into play as the auxiliary covariate.

0.7.6 The Population Intervention (In)Direct Effects

At times, the natural direct and indirect effects may prove too limiting, as these effect definitions are based on *static interventions* (i.e., setting $A = 0$ or $A = 1$), which may be unrealistic for real-world interventions. In such cases, one may turn instead to the population intervention direct effect (PIDE) and the population intervention indirect effect (PIIE), which are based on decomposing the effect of the population intervention effect (PIE) of flexible stochastic interventions (Díaz and Hejazi, 2020).

A particular type of stochastic intervention well-suited to working with binary treatments is the *incremental propensity score intervention* (IPSI), first proposed by Kennedy et al. (2017). Such interventions do not deterministically set the treatment level of an observed unit to a fixed quantity (i.e., setting $A = 1$), but instead *alter the odds of receiving the treatment* by a fixed amount ($0 \leq \delta \leq \infty$) for each individual. In particular, this intervention takes the form

$$g_\delta(1 \mid w) = \frac{\delta g(1 \mid w)}{\delta g(1 \mid w) + 1 - g(1 \mid w)},$$

where the scalar $0 < \delta < \infty$ specifies a *change in the odds of receiving treatment*. As described by Díaz and Hejazi (2020), this stochastic intervention is a special case of exponential tilting, a framework that unifies post-intervention treatment values that are draws from an altered distribution.

Unlike the natural direct and indirect effects, the conditions required for identifiability of the population intervention direct and indirect effects are more lax. Most importantly, these differences involve a (1) treatment positivity assumption that only requires that the counterfactual treatment be in the observed support of the treatment $\mathcal{A}$, and (2) no requirement of the independence any cross-world counterfactuals.

0.7.7 Decomposing the Population Intervention Effect

We may decompose the population intervention effect (PIE) in terms of the *population intervention direct effect* (PIDE) and the *population intervention indirect effect* (PIIE):

$$\mathbb{E}\{Y(A_\delta)\} - \mathbb{E}Y = \overbrace{\mathbb{E}\{Y(A_\delta, Z(A_\delta)) - Y(A_\delta, Z)\}}^{\text{PIIE}} + \overbrace{\mathbb{E}\{Y(A_\delta, Z) - Y(A, Z)\}}^{\text{PIDE}}.$$

This decomposition of the PIE as the sum of the population intervention direct and indirect effects has an interpretation analogous to the corresponding standard decomposition of the average treatment effect. In the sequel, we will compute each of the components of the direct and indirect effects above using appropriate estimators as follows

- For $\mathbb{E}\{Y(A, Z)\}$, the sample mean $\frac{1}{n}\sum_{i=1}^{n} Y_i$ is consistent;
- for $\mathbb{E}\{Y(A_\delta, Z)\}$, a TML estimator for the effect of a joint intervention altering the treatment mechanism but not the mediation mechanism, based on the proposal in Díaz and Hejazi (2020); and,
- for $\mathbb{E}\{Y(A_\delta, Z_{A_\delta})\}$, an efficient estimator for the effect of a joint intervention altering both the treatment and mediation mechanisms, as proposed in Kennedy et al. (2017) and implemented in the `npcausal` R package.

0.7.8 Estimating the Effect Decomposition Term

As described by Díaz and Hejazi (2020), the statistical functional identifying the decomposition term that appears in both the PIDE and PIIE $\mathbb{E}\{Y(A_\delta, Z)\}$, which corresponds to altering the treatment mechanism while keeping the mediation mechanism fixed, is

$$\theta_0(\delta) = \int m_0(a, z, w) g_{0,\delta}(a \mid w) p_0(z, w) d\nu(a, z, w),$$

for which a TML estimator is available. The corresponding *efficient influence function* (EIF) with respect to the nonparametric model $\mathcal{M}$ is $D_{\eta,\delta}(o) = D^Y_{\eta,\delta}(o) + D^A_{\eta,\delta}(o) + D^{Z,W}_{\eta,\delta}(o) - \theta(\delta)$.

The TML estimator may be computed basd on the EIF estimating equation and may incorporate cross-validation (Zheng and van der Laan, 2011; Chernozhukov et al., 2018) to circumvent possibly restrictive entropy conditions (e.g., Donsker class). The resultant estimator is

$$\hat{\theta}(\delta) = \frac{1}{n}\sum_{i=1}^{n} D_{\hat{\eta}_{j(i)},\delta}(O_i) = \frac{1}{n}\sum_{i=1}^{n} \left\{ D^Y_{\hat{\eta}_{j(i)},\delta}(O_i) + D^A_{\hat{\eta}_{j(i)},\delta}(O_i) + D^{Z,W}_{\hat{\eta}_{j(i)},\delta}(O_i) \right\},$$

which is implemented in `tmle3mediate` (a one-step estimator is also avaialble, in the `medshift` R package). We demonstrate the use of `tmle3mediate` to obtain $\mathbb{E}\{Y(A_\delta, Z)\}$ via its TML estimator.

0.7.9 Evaluating the Direct and Indirect Effects

We now turn to estimating the natural direct and indirect effects, as well as the population intervention direct effect, using the WASH Benefits data, introduced in earlier chapters. Let's first load the data:

```
library(data.table)
library(sl3)
library(tmle3)
library(tmle3mediate)

# download data
washb_data <- fread(
  paste0(
    "https://raw.githubusercontent.com/tlverse/tlverse-data/master/",
    "wash-benefits/washb_data.csv"
  ),
  stringsAsFactors = TRUE
)

# make intervention node binary
washb_data[, tr := as.numeric(tr != "Control")]
```

We'll next define the baseline covariates W, treatment A, mediators Z, and outcome Y nodes of the NPSEM via a "Node List" object:

```
node_list <- list(
  W = c(
    "momage", "momedu", "momheight", "hfiacat", "Nlt18", "Ncomp", "watmin",
    "elec", "floor", "walls", "roof"
  ),
  A = "tr",
  Z = c("sex", "month", "aged"),
```

```
  Y = "whz"
)
```

Here the `node_list` encodes the parents of each node – for example, Z (the mediators) have parents A (the treatment) and W (the baseline confounders), and Y (the outcome) has parents Z, A, and W. We'll also handle any missingness in the data by invoking `process_missing`:

```
processed <- process_missing(washb_data, node_list)
washb_data <- processed$data
node_list <- processed$node_list
```

We'll now construct an ensemble learner using a handful of popular machine learning algorithms:

```
# SL learners used for continuous data (the nuisance parameter M)
enet_contin_learner <- Lrnr_glmnet$new(
  alpha = 0.5, family = "gaussian", nfolds = 5
)
lasso_contin_learner <- Lrnr_glmnet$new(
  alpha = 1, family = "gaussian", nfolds = 5
)
fglm_contin_learner <- Lrnr_glm_fast$new(family = gaussian())
mean_learner <- Lrnr_mean$new()
contin_learner_lib <- Stack$new(
  enet_contin_learner, lasso_contin_learner, fglm_contin_learner, mean_learner
)
sl_contin_learner <- Lrnr_sl$new(learners = contin_learner_lib)

# SL learners used for binary data (nuisance parameters G and E in this case)
enet_binary_learner <- Lrnr_glmnet$new(
  alpha = 0.5, family = "binomial", nfolds = 5
)
lasso_binary_learner <- Lrnr_glmnet$new(
  alpha = 1, family = "binomial", nfolds = 5
)
fglm_binary_learner <- Lrnr_glm_fast$new(family = binomial())
```

```
binary_learner_lib <- Stack$new(
  enet_binary_learner, lasso_binary_learner, fglm_binary_learner, mean_learner
)
sl_binary_learner <- Lrnr_sl$new(learners = binary_learner_lib)

# create list for treatment and outcome mechanism regressions
learner_list <- list(
  Y = sl_contin_learner,
  A = sl_binary_learner
)
```

0.7.10 Estimating the Natural Indirect Effect

We demonstrate calculation of the NIE below, starting by instantiating a "Spec" object that encodes exactly which learners to use for the nuisance parameters $e(A \mid Z, W)$ and Ψ_Z. We then pass our Spec object to the `tmle3` function, alongside the data, the node list (created above), and a learner list indicating which machine learning algorithms to use for estimating the nuisance parameters based on A and Y.

```
tmle_spec_NIE <- tmle_NIE(
  e_learners = Lrnr_cv$new(lasso_binary_learner, full_fit = TRUE),
  psi_Z_learners = Lrnr_cv$new(lasso_contin_learner, full_fit = TRUE),
  max_iter = 1
)
washb_NIE <- tmle3(
  tmle_spec_NIE, washb_data, node_list, learner_list
)
washb_NIE
A tmle3_Fit that took 1 step(s)
   type                 param  init_est  tmle_est       se     lower    upper
1:  NIE NIE[Y_{A=1} - Y_{A=0}] 0.0030972 0.0029694 0.015535 -0.027479 0.033418
   psi_transformed lower_transformed upper_transformed
1:       0.0029694         -0.027479          0.033418
```

Based on the output, we conclude that the indirect effect of the treatment through the mediators (sex, month, aged) is 0.00297.

0.7.11 Estimating the Natural Direct Effect

An analogous procedure applies for estimation of the NDE, only replacing the Spec object for the NIE with `tmle_spec_NDE` to define learners for the NDE nuisance parameters:

```
tmle_spec_NDE <- tmle_NDE(
  e_learners = Lrnr_cv$new(lasso_binary_learner, full_fit = TRUE),
  psi_Z_learners = Lrnr_cv$new(lasso_contin_learner, full_fit = TRUE),
  max_iter = 1
)
washb_NDE <- tmle3(
  tmle_spec_NDE, washb_data, node_list, learner_list
)
washb_NDE
A tmle3_Fit that took 1 step(s)
   type                  param init_est tmle_est      se    lower  upper
1:  NDE NDE[Y_{A=1} - Y_{A=0}] 0.028931 0.028931 0.31693 -0.59223 0.6501
   psi_transformed lower_transformed upper_transformed
1:        0.028931          -0.59223            0.6501
```

From this, we can draw the conclusion that the direct effect of the treatment (through all paths not involving the mediators (sex, month, aged)) is 0.02893. Note that, together, the estimates of the natural direct and indirect effects approximately recover the *average treatment effect*, that is, based on these estimates of the NDE and NIE, the ATE is roughly 0.0319.

0.7.12 Estimating the Population Intervention Direct Effect

As previously noted, the assumptions underlying the natural direct and indirect effects may be challenging to justify; moreover, the effect definitions themselves depend on the application of a static intervention to the treatment, sharply limiting their flexibility. When considering binary treatments, incremental propensity score shifts provide an alternative class of flexible, stochastic interventions. We'll now consider estimating the PIDE with an IPSI that modulates the odds of receiving treatment by $\delta = 3$. Such an intervention may be interpreted (hypothetically) as the effect of a design that encourages study participants to opt in to receiving the treatment, thus increasing their relative odds of receiving said treatment. To exemplify our approach,

we postulate a motivational intervention that *triples the odds* (i.e., $\delta = 3$) of receiving the treatment for each individual:

```
# set the IPSI multiplicative shift
delta_ipsi <- 3

# instantiate tmle3 spec for stochastic mediation
tmle_spec_pie_decomp <- tmle_medshift(
  delta = delta_ipsi,
  e_learners = Lrnr_cv$new(lasso_binary_learner, full_fit = TRUE),
  phi_learners = Lrnr_cv$new(lasso_contin_learner, full_fit = TRUE)
)

# compute the TML estimate
washb_pie_decomp <- tmle3(
  tmle_spec_pie_decomp, washb_data, node_list, learner_list
)
washb_pie_decomp
A tmle3_Fit that took 510 step(s)
   type         param init_est tmle_est       se   lower    upper
1: PIDE E[Y_{A=NULL}] -0.58163 -0.58385 0.016302 -0.6158 -0.55189
   psi_transformed lower_transformed upper_transformed
1:        -0.58385           -0.6158          -0.55189
```

Recall that, based on the decomposition outlined previously, the PIDE may be denoted $\beta_{\text{PIDE}}(\delta) = \theta_0(\delta) - \mathbb{E}Y$. Thus, an estimator of the PIDE, $\hat{\beta}_{\text{PIDE}}(\delta)$ may be expressed as a composition of estimators of its constituent parameters:

$$\hat{\beta}_{\text{PIDE}}(\delta) = \hat{\theta}(\delta) - \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

Based on the above, we may construct an estimator of the PIDE using the already estimated decomposition term and the empirical (marginal) mean of the outcome. Thus, our estimate of the PIDE is approximately 0.00223. Note that this is a straightforward application of the delta method and could equivalently be performed using the functionality exposed in the `tmle3` package.

0.8 A Primer on the R6 Class System

A central goal of the Targeted Learning statistical paradigm is to estimate scientifically relevant parameters in realistic (usually nonparametric) models.

The `tlverse` is designed using basic OOP principles and the `R6` OOP framework. While we've tried to make it easy to use the `tlverse` packages without worrying much about OOP, it is helpful to have some intuition about how the `tlverse` is structured. Here, we briefly outline some key concepts from OOP. Readers familiar with OOP basics are invited to skip this section.

0.8.1 Classes, Fields, and Methods

The key concept of OOP is that of an object, a collection of data and functions that corresponds to some conceptual unit. Objects have two main types of elements:

1. *fields*, which can be thought of as nouns, are information about an object, and
2. *methods*, which can be thought of as verbs, are actions an object can perform.

Objects are members of classes, which define what those specific fields and methods are. Classes can inherit elements from other classes (sometimes called base classes) – accordingly, classes that are similar, but not exactly the same, can share some parts of their definitions.

Many different implementations of OOP exist, with variations in how these concepts are implemented and used. R has several different implementations, including `S3`, `S4`, reference classes, and `R6`. The `tlverse` uses the `R6` implementation. In `R6`, methods and fields of a class object are accessed using the `$` operator. For a more thorough introduction to `R`'s various OOP systems, see `http://adv-r.had.co.nz/OO-essentials.html`, from Hadley Wickham's *Advanced R* (Wickham, 2014).

0.8.2 Object Oriented Programming: `Python` and `R`

OO concepts (classes with inherentence) were baked into Python from the first published version (version 0.9 in 1991). In contrast, `R` gets its OO "approach" from its predecessor, `S`, first released in 1976. For the first 15 years, `S` had no support for

classes, then, suddenly, S got two OO frameworks bolted on in rapid succession: informal classes with S3 in 1991, and formal classes with S4 in 1998. This process continues, with new OO frameworks being periodically released, to try to improve the lackluster OO support in R, with reference classes (R5, 2010) and R6 (2014). Of these, R6 behaves most like Python classes (and also most like OOP focused languages like C++ and Java), including having method definitions be part of class definitions, and allowing objects to be modified by reference.

Bibliography

Anonymous (2015). Let's think about cognitive bias. *Nature*, 526(7572).

Avin, C., Shpitser, I., and Pearl, J. (2005). Identifiability of path-specific effects. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 357–363.

Baker, M. (2016). Is there a reproducibility crisis? a nature survey lifts the lid on how researchers view the crisis rocking science and what they think will help. *Nature*, 533(7604):452–455.

Bengtsson, H. (2020). A unifying framework for parallel and distributed processing in r using futures.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buckheit, J. B. and Donoho, D. L. (1995). Wavelab and reproducible research. In *Wavelets and statistics*, pages 55–81. Springer.

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., and Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1).

Coyle, J. R. and Hejazi, N. S. (2018). origami: A generalized framework for cross-validation in r. *The Journal of Open Source Software*, 3(21).

Coyle, J. R., Hejazi, N. S., Malenica, I., Phillips, R. V., Arnold, B. F., Mertens, A., Benjamin-Chung, J., Cai, W., Dayal, S., Colford Jr., J. M., Hubbard, A. E., and van der Laan, M. J. (2021). Targeting Learning: Robust statistics for reproducible research. *arXiv*.

Díaz, I. and Hejazi, N. S. (2020). Causal mediation analysis for stochastic interventions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(3):661–683.

Díaz, I., Hejazi, N. S., Rudolph, K. E., and van der Laan, M. J. (2020). Nonparametric efficient causal mediation with intermediate confounders. *Biometrika*.

Dudoit, S. and van der Laan, M. J. (2005). Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. *Statistical Methodology*, 2(2):131–154.

Editorial, N. (2015). How scientists fool themselves — and how they can stop. *Nature*, 526(7572).

Hejazi, N. S., Rudolph, K. E., van der Laan, M. J., and Díaz, I. (2021). Nonparametric causal mediation analysis for stochastic interventional (in)direct effects.

Imai, K., Keele, L., and Yamamoto, T. (2010). Identification, inference and sensitivity analysis for causal mediation effects. *Statistical science*, pages 51–71.

Kennedy, E. H., Ma, Z., McHugh, M. D., and Small, D. S. (2017). Nonparametric methods for doubly robust estimation of continuous treatment effects. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):1229–1245.

Munafò, M. R., Nosek, B. A., Bishop, D. V., Button, K. S., Chambers, C. D., Du Sert, N. P., Simonsohn, U., Wagenmakers, E.-J., Ware, J. J., and Ioannidis, J. P. (2017). A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):0021.

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., and Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606.

Peng, R. (2015). The reproducibility crisis in science: A statistical counterattack. *Significance*, 12(3):30–32.

Petersen, M. L., Sinisi, S. E., and van der Laan, M. J. (2006). Estimation of direct causal effects. *Epidemiology*, pages 276–284.

Pullenayegum, E. M., Platt, R. W., Barwick, M., Feldman, B. M., Offringa, M., and Thabane, L. (2016). Knowledge translation in biostatistics: a survey of current practices, preferences, and barriers to the dissemination and uptake of new statistical methods. *Statistics in medicine*, 35(6):805–818.

Robins, J. M. and Greenland, S. (1992). Identifiability and exchangeability for direct and indirect effects. *Epidemiology*, pages 143–155.

Robins, J. M. and Richardson, T. S. (2010). Alternative graphical causal models and the identification of direct effects. *Causality and psychopathology: Finding the determinants of disorders and their cures*, pages 103–158.

Sandercock, P., Collins, R., Counsell, C., Farrell, B., Peto, R., Slattery, J., and Warlow, C. (1997). The international stroke trial (ist): a randomized trial of aspirin, subcutaneous heparin, both, or neither among 19,435 patients with acute ischemic stroke. *Lancet*, 349(9065):1569–1581.

Sandercock, P. A., Niewada, M., and Członkowska, A. (2011). The international stroke trial database. *Trials*, 12(1):101.

Stark, P. B. and Saltelli, A. (2018). Cargo-cult statistics and scientific crisis. *Significance*, 15(4):40–43.

Stromberg, A. et al. (2004). Why write statistical software? the case of robust statistical methods. *Journal of Statistical Software*, 10(5):1–8.

Szucs, D. and Ioannidis, J. (2017). When null hypothesis significance testing is unsuitable for research: a reassessment. *Frontiers in human neuroscience*, 11:390.

Tchetgen Tchetgen, E. J. (2013). Inverse odds ratio-weighted estimation for causal mediation analysis. *Statistics in Medicine*, 32(26):4567–4580.

Tchetgen Tchetgen, E. J. and Shpitser, I. (2012). Semiparametric theory for causal mediation analysis: efficiency bounds, multiple robustness, and sensitivity analysis. *Annals of Statistics*, 40(3):1816–1845.

Textor, J., Hardt, J., and Knüppel, S. (2011). Dagitty: a graphical tool for analyzing causal diagrams. *Epidemiology*, 22(5):745.

Tofail, F., Fernald, L. C., Das, K. K., Rahman, M., Ahmed, T., Jannat, K. K., Unicomb, L., Arnold, B. F., Ashraf, S., Winch, P. J., et al. (2018). Effect of water quality, sanitation, hand washing, and nutritional interventions on child development in rural bangladesh (wash benefits bangladesh): a cluster-randomised controlled trial. *The Lancet Child & Adolescent Health*, 2(4):255–268.

van der Laan, M. J., Dudoit, S., and Keles, S. (2004). Asymptotic optimality of likelihood-based cross-validation. *Statistical Applications in Genetics and Molecular Biology*, 3(1):1–23.

van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1).

van der Laan, M. J. and Rose, S. (2011). *Targeted learning: causal inference for observational and experimental data.* Springer Science & Business Media.

van der Laan, M. J. and Rose, S. (2018). *Targeted Learning in Data Science: Causal Inference for Complex Longitudinal Studies*. Springer Science & Business Media.

van der Laan, M. J. and Starmans, R. J. (2014). Entering the era of data science: Targeted learning and the integration of statistics and computational data analysis. *Advances in Statistics*, 2014.

Van der Vaart, A. W., Dudoit, S., and van der Laan, M. J. (2006). Oracle inequalities for multi-fold cross validation. *Statistics & Decisions*, 24(3):351–371.

VanderWeele, T. (2015). *Explanation in causal inference: methods for mediation and interaction*. Oxford University Press.

Wickham, H. (2014). *Advanced r*. Chapman and Hall/CRC.

Zheng, W. and van der Laan, M. J. (2011). Cross-validated targeted minimum-loss-based estimation. In *Targeted Learning*, pages 459–474. Springer.

Zheng, W. and van der Laan, M. J. (2012). Targeted maximum likelihood estimation of natural direct effects. *International Journal of Biostatistics*, 8(1).