A Ride in Targeted Learning Territory

David Benkeser (Emory University)
Antoine Chambaz (Université Paris Descartes)

2018-10-18

A Ride in Targeted Learning Territory

David Benkeser (Emory Univeristy)

Antoine Chambaz (Université Paris Descartes)

October 18, 2018

Long Mendocino Drive
(detail, Liana Steinmetz)

Contents

Welcome

This is either the website or the text called **"A Ride in Targeted Learning Territory"**. In the former case, the text can be downloaded by clicking on the dedicated button in the top part of the webpage. In the latter case, the website can be browsed here.

Long Mendocino Drive (detail, Liana Steinmetz)

Organization

The text takes the form of a series of brief sections. The main sections combine theoretical and computational developments. The code is written in the programming language `R`. `R` is widely used among statisticians and data scientists to develop statistical software and data analysis.

Regularly, a section is inserted that proposes exercizes. Each such section is indicated by the ⚙ symbol. The symbol ⪰ also indicates those sections whose content is more involved.

Overview

After a short introduction, we present the reproducible experiment that will play a central role throughout the text. Then, we introduce the main parameter of interest. We comment upon some of its properties that are useful from a statistical perspective. This paves the way to the presentation of several estimators that are increasingly more powerful statistically. The discussion takes us into *targeted learning territory*.

Audience

The text might be of interest to students in statistics and machine learning. It might also serve as a gentle introduction to targeted learning in both its theoretical and computational aspects before delving deeper into the literature [van der Laan and Rose, 2011], [van der Laan and Rose, 2018].

Technical details

The text is written in `RMarkdown` with `bookdown`. Assuming that the `knitr` package is installed, you can retrieve all the `R` code by running

```
knitr::purl("abcd.Rmd")
```

Section 1

A ride

1.1 Introduction

Our ambition is to present a gentle introduction to the inference of a causal quantity whose statistical analysis is typical and thus paves the way to more involved analyses. The introduction weaves together two main threads, one theoretical and the other computational.

1.1.1 A causal story

We focus on a causal story where a random reward (a real number between 0 and 1) depends on the action undertaken (one among two) and the random context where the action is performed (summarized by a real number between 0 and 1). The causal quantity of interest is the average difference of the two counterfactual rewards.

We will build several estimators and discuss their respective merits, theoretically and computationally. The construction of the most involved estimator will unfold in *targeted learning territory*, at the frontier of machine learning and semiparametrics, the statistical theory of inference based on semiparametric models.

1.1.2 The `tlrider` package

The computational illustrations will be developed based on the companion package `tlrider`. Make sure you have installed it, for instance by running the following chunk of code:

```
devtools::install_github("achambaz/tlride/tlrider")
```

Note that additional packages are required, among which `tidyverse` [Wickham and Grolemund, 2016], `caret` and `ggdag`. Assuming that these are installed too, we can run the next chunk of code:

```
set.seed(54321) ## because reproducibility matters...
library(tidyverse)
library(caret)
library(ggdag)
library(tlrider)
```

1.1.3 What we will discuss

...

1.2 A simulation study

1.2.1 Reproducible experiment as a law

We are interested in a reproducible experiment. Every time this experiment is run, it generates an observation that we call O. We view O as a random variable drawn from *the law of the experiment* that we denote by P_0.

We view P_0 as an element of *the model* $\mathcal{M}$. The model is a collection of laws. In particular, the model contains all laws that we think may plausibly describe the law of the experiment. Thus, the choice of model is based on our scientific knowledge of the experiment. The more we know about the experiment, the smaller is $\mathcal{M}$. In all our examples, we use large models that reflect a lack of knowledge about many aspects of the experiment.

1.2.2 A synthetic reproducible experiment

Instead of considering a real-life reproducible experiment, we focus for pedagogical purposes on a *synthetic* reproducible experiment. Thus we can from now on take on two different roles: that of an *oracle* knowing completely the nature of the experiment, and that of a *statistician* eager to know more about the experiment by observing some of its outputs.

Let us run the example built into the `tlrider` package:

```
example(tlrider)
```

A few objects have been defined:

```
ls()
#> [1] "another_experiment" "experiment"         "expit"
#> [4] "logit"              "sigma0"
```

Function `expit` implements the link function expit : $\mathbb{R} \to]0,1[$ given by $\text{expit}(x) \equiv (1 + e^{-x})^{-1}$. Function `logit` implements its inverse function logit $:]0,1[\to \mathbb{R}$ given by $\text{logit}(p) \equiv \log[p/(1-p)]$.

Let us take a look at `experiment`:

```
experiment
#> A law for (W,A,Y) in [0,1] x {0,1} x [0,1].
#>
#> If the law is fully characterized, you can use method
#> 'sample_from' to sample from it.
#>
#> If you built the law, or if you are an _oracle_, you can
#> also use methods 'reveal' to reveal its relevant features
#> (QW, Gbar, Qbar, qY -- see '?reveal'), and 'alter' to change
#> some of them.
#>
#> If all its relevant features are characterized, you can
#> use methods 'evaluate_psi' to obtain the value of 'Psi' at
#> this law (see '?evaluate_psi') and 'evaluate_eic' to obtain
#> the efficient influence curve of 'Psi' at this law (see '?
#> evaluate_eic').
```

The law P_0 of the synthetic experiment `experiment` built by us generates a generic observation O that decomposes as

$$O \equiv (W, A, Y) \in [0,1] \times \{0,1\} \times [0,1].$$

We interpret W as a real valued summary measure of a random context where an action A chosen among two is undertaken, leading to a real valued reward Y.

We can sample from the experiment (simply run `?sample_from` to see the man page of method `sample_from`). The next chunk of code runs the experiment five times, independently:

```
(five_obs <- sample_from(experiment, n = 5))
#>           W A     Y
#> [1,] 0.429 1 0.981
#> [2,] 0.454 1 0.855
#> [3,] 0.377 0 0.836
#> [4,] 0.461 1 0.582
#> [5,] 0.419 1 0.878
```

1.2.3 Revealing `experiment`

Acting as oracles, we can peek into `experiment` and *reveal* a selection of relevant features (simply run `?reveal` to see the man page of method `reveal`). Made by us, the selection exhibits features that will play an important role in the text.

```
relevant_features <- reveal(experiment)
names(relevant_features)
#> [1] "QW"          "Gbar"        "Qbar"        "qY"          "sample_from"
```

We have an oracular knowledge of `experiment` and can thus comment upon the features of P_0 revealed in `relevant_features`.

`QW`

The `QW` feature describes the marginal law of W, that we call $Q_{0,W}$.[1]

```
relevant_features$QW
#> function(W,
#>                      mixture_weights = c(1/10, 9/10, 0),
#>                      mins = c(0, 11/30, 0),
#>                      maxs = c(1, 14/30, 1)) {
#>         out <- sapply(1:length(mixture_weights),
#>                       function(ii){
#>                         mixture_weights[ii] *
#>                           stats::dunif(W,
#>                                        min = mins[ii],
#>                                        max = maxs[ii])
#>                       })
#>         return(rowSums(out))
#>       }
#> <environment: 0xcea19d0>
```

It appears that $Q_{0,W}$ is a mixture of the uniform laws over $[0, 1]$ (weight $1/10$) and $[11/30, 14/30]$ (weight $9/10$).[2]

`Gbar`

The `Gbar` feature describes the conditional probability of action $A = 1$ given W. For each $a \in \{0, 1\}$, we denote

[1] A summary of the notation used throughout the text is presented here.

[2] We fine-tuned the marginal law $Q_{0,W}$ of W to make it easier later on to drive home important messages.

$$\bar{G}_0(W) \equiv \Pr_{P_0}(A = 1|W),$$
$$\ell\bar{G}_0(a, W) \equiv \Pr_{P_0}(A = a|W).$$

Obviously,

$$\ell\bar{G}_0(A, W) \equiv A\bar{G}_0(W) + (1 - A)(1 - \bar{G}_0(W)).$$

```
relevant_features$Gbar
#> function(W) {
#>          expit(1 + 2 * W - 4 * sqrt(abs((W - 5/12))))
#>        }
#> <environment: 0xcea19d0>
```

Note how real numbers of the form $1 + 2W - 4 * \sqrt{|W - 5/12|})$ are mapped into the interval $[0, 1]$ by the expit link function.

qY

The `qY` feature describes the conditional density of Y given A and W. For each $y \in]0, 1[$, we denote by $q_{0,Y}(y, A, W)$ the conditional density evaluated at y of Y given A and W.

```
relevant_features$qY
#> function(obs, Qbar, shape10 = 2, shape11 = 3){
#>          A <- obs[, "A"]
#>          AW <- obs[, c("A", "W")]
#>          QAW <- Qbar(AW)
#>          shape1 <- ifelse(A == 0, shape10, shape11)
#>          stats::dbeta(Y,
#>                       shape1 = shape1,
#>                       shape2 = shape1 * (1 - QAW) / QAW)
#>        }
#> <environment: 0xcea19d0>
```

It appears that the conditional law of Y given A and W is the Beta law with conditional mean and variance characterized by the `Qbar` feature of `experiment` (see below) and the `shape10` and `shape11` parameters.

Qbar

As for the `Qbar` feature, it describes the conditional mean of Y given A and W.

```
relevant_features$Qbar
#> function(AW) {
#>         A <- AW[, "A"]
#>         W <- AW[, "W"]
#>         A * (cos((-1/2 + W) * pi) * 2/5 + 1/5 +
#>              (1/3 <= W & W <= 1/2) / 5 +
#>              (W >= 3/4) * (W - 3/4) * 2) +
#>           (1 - A) * (sin(4 * W^2 * pi) / 4 + 1/2)
#>       }
#> <bytecode: 0xda2b068>
#> <environment: 0xcea19d0>
```

We denote $\bar{Q}_0(A, W) = \mathrm{E}_{P_0}(Y|A, W)$ the conditional mean of Y given A and W. Note how $\bar{Q}_0(A, W)$ does depend heavily on A and W. We refer the reader to Section 1.3 for a visualization of $\bar{Q}_0$.

`sample_from`

Finally, the `sample_from` feature is the function called by method `sample_from` when it is applied to an object of class `LAW`, like `experiment`.

```
relevant_features$sample_from
#> function(n, ideal = FALSE) {
#>         ## preliminary
#>         n <- R.utils::Arguments$getInteger(n, c(1, Inf))
#>         ideal <- R.utils::Arguments$getLogical(ideal)
#>         ## ## 'Gbar' and 'Qbar' factors
#>         Gbar <- experiment$.Gbar
#>         Qbar <- experiment$.Qbar
#>         ## sampling
#>         ## ## context
#>         params <- formals(experiment$.QW)
#>         mixture_weights <- eval(params$mixture_weights)
#>         mins <- eval(params$mins)
#>         maxs <- eval(params$maxs)
#>         W <- sample_from_mixture_of_uniforms(n, mixture_weights,
#>                                              mins, maxs)
#>         ## ## counterfactual rewards
#>         zeroW <- cbind(A = 0, W)
#>         oneW <- cbind(A = 1, W)
#>         Qbar_zeroW <- Qbar(zeroW)
#>         Qbar_oneW <- Qbar(oneW)
```

```
#>         Yzero <- stats::rbeta(n,
#>                               shape1 = 2,
#>                               shape2 = 2 * (1 - Qbar_zeroW) / Qbar_zeroW)
#>         Yone <- stats::rbeta(n,
#>                              shape1 = 3,
#>                              shape2 = 3 * (1 - Qbar_oneW) / Qbar_oneW)
#>         ## ## action undertaken
#>         A <- stats::rbinom(n, size = 1, prob = Gbar(W))
#>         ## ## actual reward
#>         Y <- A * Yone + (1 - A) * Yzero
#>         ## ## observation
#>         if (ideal) {
#>           obs <- cbind(W = W, Yzero = Yzero, Yone = Yone, A = A, Y = Y)
#>         } else {
#>           obs <- cbind(W = W, A = A, Y = Y)
#>         }
#>         return(obs)
#>       }
#> <bytecode: 0xc186e50>
#> <environment: 0xcea19d0>
```

We will comment upon the `ideal` argument in the above `sample_from` feature in Section 2.1.

1.3 ⚙ Visualization

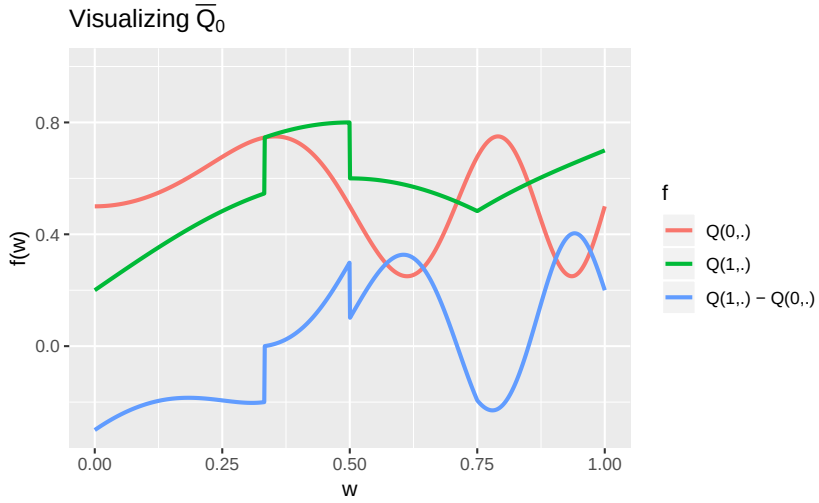
1. Run the following chunk of code. It visualizes the conditional mean $\bar{Q}_0$.

```
Gbar <- relevant_features$Gbar
Qbar <- relevant_features$Qbar
QW <- relevant_features$QW

features <- tibble(w = seq(0, 1, length.out = 1e3)) %>%
  mutate(Qw = QW(w),
         Gw = Gbar(w),
         Q1w = Qbar(cbind(A = 1, W = w)),
         Q0w = Qbar(cbind(A = 0, W = w)),
         blip_Qw = Q1w - Q0w)

features %>% select(-Qw, -Gw) %>%
  rename("Q(1,.)" = Q1w,
         "Q(0,.)" = Q0w,
```

```
        "Q(1,.) - Q(0,.)" = blip_Qw) %>%
  gather("f", "value", -w) %>%
  ggplot() +
  geom_line(aes(x = w, y = value, color = f), size = 1) +
  labs(y = "f(w)", title = bquote("Visualizing" ~ bar(Q)[0])) +
  ylim(NA, 1)
```



2. Adapt the above chunk of code to visualize the marginal density $Q_{0,W}$ and conditional probability $\bar{G}_0$.

1.4 ✪ Make your own experiment

You can easily make your own experiment.

1. Check out the man page of method `alter` by running `?alter`.
2. Run the following chunk of code:

```
my_experiment <- LAW() ## creates an object of class 'LAW'
alter(my_experiment,   ## characterize its relevant features
      QW = function(W) {
        out <- rep_len(0, length(W))
        out[W == 0] <- 1/4
        out[W == 1] <- 3/4
        return(out)
      },
      Gbar = function(W) {
        out <- rep_len(0, length(W))
        out[W == 0] <- 1/3
```

```
    out[W == 1] <- 3/5
    return(out)
  },
  Qbar = function(AW) {
    probs <- matrix(c(1/2, 2/3, 7/8, 4/5), ncol = 2,
                    dimnames = list(c("A=0", "A=1"),
                                    c("W=0", "W=1")))
    probs[cbind(AW[, "A"] + 1, AW[, "W"] + 1)]
  },
  qY = function(obs) {
    probs <- matrix(c(1/2, 2/3, 7/8, 4/5), ncol = 2,
                    dimnames = list(c("A=0", "A=1"),
                                    c("W=0", "W=1")))
    probs <- probs[cbind(obs[, "A"] + 1, obs[, "W"] + 1)]
    obs[, "Y"] * probs + (1 - obs[, "Y"]) * (1 - probs)
  },
  sample_from = function(n) {
    ## preliminary
    n <- R.utils::Arguments$getInteger(n, c(1, Inf))
    ## 'QW', 'Gbar' and 'Qbar' features
    QW <- my_experiment$.QW
    Gbar <- my_experiment$.Gbar
    Qbar <- my_experiment$.Qbar
    ## sampling
    W <- rbinom(n, size = 1, prob = QW(1))
    A <- rbinom(n, size = 1, prob = Gbar(W))
    AW <- cbind(W = W, A = A)
    Y <- rbinom(n, size = 1, Qbar(AW))
    return(cbind(AW, Y = Y))
  })
```

3. What does the next chunk do?

```
(sample_from(my_experiment, 3))
#>      W A Y
#> [1,] 0 0 1
#> [2,] 1 1 1
#> [3,] 1 0 1
```

4. Characterize entirely the law of `my_experiment`. Hint:

```
obs <- sample_from(my_experiment, 1e4)
obs %>% as.tibble %>% group_by(W, A, Y) %>% summarize(how_many = n())
#> # A tibble: 8 x 4
```

```
#> # Groups:   W, A [?]
#>       W     A     Y how_many
#>   <int> <int> <int>    <int>
#> 1     0     0     0      826
#> 2     0     0     1      808
#> 3     0     1     0      285
#> 4     0     1     1      556
#> 5     1     0     0      419
#> 6     1     0     1     2665
#> # ... with 2 more rows
obs %>% as.tibble %>% group_by(W, A) %>% summarize(prob = mean(Y))
#> # A tibble: 4 x 3
#> # Groups:   W [?]
#>       W     A  prob
#>   <int> <int> <dbl>
#> 1     0     0 0.494
#> 2     0     1 0.661
#> 3     1     0 0.864
#> 4     1     1 0.798
```

5. Now, make your own experiment.

Section 2

The parameter of interest

2.1 The parameter of interest

2.1.1 Definition

It happens that we especially care for a finite-dimensional feature of P_0 that we denote by ψ_0. Its definition involves two of the aforementioned infinite-dimensional features, the marginal law $Q_{0,W}$ of W and the conditional mean $\bar{Q}_0$ of Y given A and W:

$$\begin{aligned}\psi_0 &\equiv \int \left(\bar{Q}_0(1,w) - \bar{Q}_0(0,w)\right) dQ_{0,W}(w) \\ &= \mathrm{E}_{P_0}\left(\mathrm{E}_{P_0}(Y \mid A=1,W) - \mathrm{E}_{P_0}(Y \mid A=0,W)\right).\end{aligned} \tag{2.1}$$

Acting as oracles, we can compute explicitly the numerical value of ψ_0. The `evaluate_psi` method makes it very easy (simply run `?estimate_psi` to see the man page of the method):

```
(psi_zero <- evaluate_psi(experiment))
#> [1] 0.0832
```

2.1.2 A causal interpretation

Our interest in ψ_0 is of causal nature. Taking a closer look at the `sample_from` feature of `experiment` reveals indeed that the random making of an observation O drawn from P_0 can be summarized by the following causal graph:

```
dagify(
  Y ~ A + Y1 + Y0, A ~ W, Y1 ~ W, Y0 ~ W,
  labels = c(Y = "Actual reward",
```

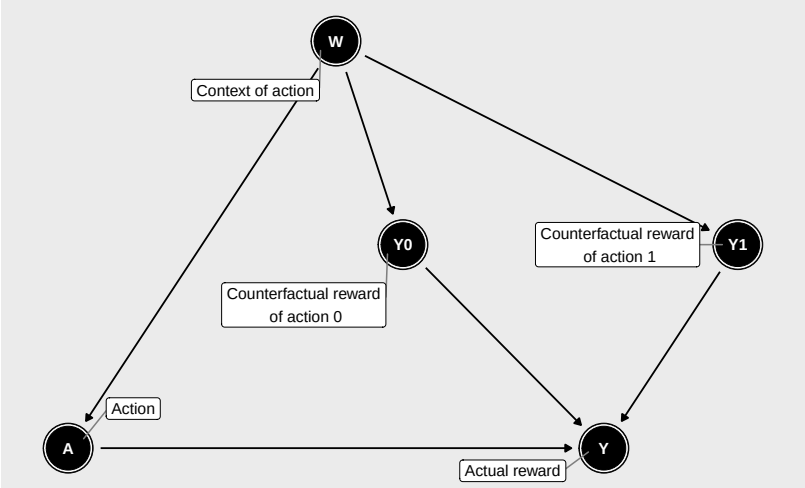


Figure 2.1: Causal graph summarizing the inner causal mechanism at play in `experiment`.

```
            A = "Action",
            Y1 = "Counterfactual reward\n of action 1",
            Y0 = "Counterfactual reward\n of action 0",
            W = "Context of action"),
  coords = list(
    x = c(W = 0, A = -1, Y1 = 1.5, Y0 = 0.25, Y = 1),
    y = c(W = 0, A = -1, Y1 = -0.5, Y0 = -0.5, Y = -1)),
  outcome = "Y",
  exposure = "A",
  latent = c("Y0", "Y1")) %>% tidy_dagitty %>%
  ggdag(text = TRUE, use_labels = "label") + theme_dag_grey()
```

In words, the experiment unfolds like this (see also Section 11.1):

1. a context of action $W \in [0, 1]$ is randomly generated;
2. two counterfactual rewards $Y_0 \in [0, 1]$ and $Y_1 \in [0, 1]$ are generated conditionally on W;
3. an action $A \in \{0, 1\}$ (among two possible actions called $a = 0$ and $a = 1$) is undertaken, *(i)* knowing the context but *not* the counterfactual rewards, and *(ii)* in such a way that both actions can always be considered;
4. the action yields a reward Y, which equals either Y_0 or Y_1 depending on whether action $a = 0$ or $a = 1$ has been undertaken;
5. summarize the course of the experiment with $O \equiv (W, A, Y)$, thus concealing Y_0 and Y_1.

The above description of the experiment is useful to reinforce what it means to run the "ideal" experiment by setting argument `ideal` to `TRUE` in a call to `sample_from` for `experiment` (see Section 2.1.3). Doing so triggers a modification of the nature of the experiment, enforcing that the counterfactual rewards Y_0 and Y_1 be part of the summary of the experiment eventually. In light of the above enumeration,

$$\mathbb{O} \equiv (W, Y_0, Y_1, A, Y)$$

is output, as opposed to its summary measure O. This defines another experiment and its law, that we denote $\mathbb{P}_0$.

It is straightforward to show that

$$\begin{aligned} \psi_0 &= \mathrm{E}_{\mathbb{P}_0}(Y_1 - Y_0) \\ &= \mathrm{E}_{\mathbb{P}_0}(Y_1) - \mathrm{E}_{\mathbb{P}_0}(Y_0). \end{aligned} \tag{2.2}$$

Thus, ψ_0 describes the average difference of the two counterfactual rewards. In other words, ψ_0 quantifies the difference in average of the reward one would get in a world where one would always enforce action $a = 1$ with the reward one would get in a world where one would always enforce action $a = 0$. This said, it is worth emphasizing that ψ_0 is a well-defined parameter beyond its causal interpretation, and that it describes a standardized association between the action A and reward Y.

2.1.3 A causal computation

We can use our position as oracles to sample observations from the ideal experiment. We call `sample_from` for `experiment` with its argument `ideal` set to `TRUE` in order to numerically approximate ψ_0. By the law of large numbers, the following code approximates ψ_0 and shows it approximate value.

```
B <- 1e5
ideal_obs <- sample_from(experiment, B, ideal = TRUE)
(psi_approx <- mean(ideal_obs[, "Yone"] - ideal_obs[, "Yzero"]))
#> [1] 0.0841
```

The object `psi_approx` contains an approximation to ψ_0 based on `B` observations from the ideal experiment. The random sampling of observations results in uncertainty in the numerical approximation of ψ_0. This uncertainty can be quantified by constructing a 95% confidence interval for ψ_0. The central limit theorem and Slutsky's lemma allow us to build such an interval as follows.

```r
sd_approx <- sd(ideal_obs[, "Yone"] - ideal_obs[, "Yzero"])
alpha <- 0.05
(psi_approx_CI <- psi_approx + c(-1, 1) *
   qnorm(1 - alpha / 2) * sd_approx / sqrt(B))
#> [1] 0.0822 0.0860
```

We note that the interpretation of this confidence interval is that in 95% of draws of size `B` from the ideal data generating experiment, the true value of ψ_0 will be contained in the generated confidence interval.

2.2 ✪ An alternative parameter of interest

Equality (2.2) shows that parameter ψ_0 (2.1) is the difference in average rewards if we enforce action $a = 1$ rather than $a = 0$. An alternative way to describe the rewards under different actions involves *quantiles* as opposed to *averages*.

Let

$$Q_{0,Y}(y, A, W) \equiv \int_0^y q_{0,Y}(u, A, W)du$$

be the conditional cumulative distribution of reward Y given A and W, evaluated at $y \in]0, 1[$, that is implied by P_0. For each action $a \in \{0, 1\}$ and $c \in]0, 1[$, introduce

$$\gamma_{0,a,c} \equiv \inf \left\{ y \in]0, 1[: \int Q_{0,Y}(y, a, w) dQ_{0,W}(w) \geq c \right\}. \tag{2.3}$$

It is not very difficult to check (see Problem 1 below) that

$$\gamma_{0,a,c} = \inf \left\{ y \in]0, 1[: \Pr_{\mathbb{P}_0}(Y_a \leq y) \geq c \right\}. \tag{2.4}$$

Thus, $\gamma_{0,a,c}$ can be interpreted as a covariate-adjusted c-th quantile reward when action a is enforced. The difference

$$\delta_{0,c} \equiv \gamma_{0,1,c} - \gamma_{0,0,c} \tag{2.5}$$

is the c-th quantile counterpart to parameter ψ_0 (2.1).

1. ⊇ Prove (2.4).
2. ⊇ Compute the numerical value of $\gamma_{0,a,c}$ for each $(a, c) \in \{0, 1\} \times \{1/4, 1/2, 3/4\}$ using the appropriate features of `experiment` (see `relevant_features`). Based on these results, report the numerical value of $\delta_{0,c}$ for each $c \in \{1/4, 1/2, 3/4\}$.

3. Approximate the numerical values of $\gamma_{0,a,c}$ for each $(a,c) \in \{0,1\} \times \{1/4, 1/2, 3/4\}$ by drawing a large sample from the "ideal" data experiment and using empirical quantile estimates. Deduce from these results a numerical approximation to $\delta_{0,c}$ for $c \in \{1/4, 1/2, 3/4\}$. Confirm that your results closely match those obtained in the previous problem.

2.3 The statistical mapping of interest

The noble way to define a statistical parameter is to view it as the value of a statistical mapping at the law of the experiment of interest. Beyond the elegance, this has paramount statistical implications.

2.3.1 Opening discussion

Oftentimes, the premise of a statistical analysis is presented like this. One assumes that the law P_0 of the experiment of interest belongs to a statistical model

$$\{P_\theta : \theta \in T\}.$$

The statistical model is identifiable, meaning that if two elements P_θ and $P_{\theta'}$ coincide, then necessarily $\theta = \theta'$. Therefore, there exists a unique $\theta_0 \in T$ such that $P_0 = P_{\theta_0}$, and one wishes to estimate θ_0.

For instance, each P_θ could be the Gaussian law with mean $\theta \in T \equiv \mathbb{R}$ and variance 1, and one could wish to estimate the mean θ_0 of P_0. To do so, one could rely on n observations X_1, ..., X_n drawn independently from P_0. The empirical mean

$$\theta_n \equiv \frac{1}{n} \sum_{i=1}^{n} X_i$$

estimates θ_0. If we assume that $\mathrm{Var}_{P_0}(X_1)$ is finite, then θ_n satisfies many useful properties. In particular, it can be used to construct confidence intervals.

Of course, the mean of a law is defined beyond the small model $\{P_\theta : \theta \in \mathbb{R}\}$. Let $\mathcal{M}$ be the set of laws P on $\mathbb{R}$ such that $\mathrm{Var}_P(X)$ is finite. In particular, $P_0 \in \mathcal{M}$. For every $P \in \mathcal{M}$, the mean $E_P(X)$ is well defined. Thus, we can introduce the *statistical mapping* $\Theta : \mathcal{M} \to \mathbb{R}$ given by

$$\Theta(P) \equiv E_P(X).$$

Interestingly, the empirical measure P_n[1] is an element of $\mathcal{M}$. Therefore, the statistical mapping Θ can be evaluated at P_n:

$$\Theta(P_n) = \frac{1}{n}\sum_{i=1}^{n} X_i = \theta_n.$$

We *recover* the empirical mean, and understand that it is a *substitution* estimator of the mean: in order to estimate $\Theta(P_0)$, we *substitute* P_n for P_0 within Θ.

Substitution-based estimators are particularly valuable notably because they, by construction, satisfy all the constraints to which the targeted parameter is subjected. Some of the estimators that we will build together are substitution-based, some are not.

2.3.2 The parameter as the value of a statistical mapping at the experiment

We now go back to our main topic of interest. Suppose we know beforehand that O drawn from P_0 takes its values in $\mathcal{O} \equiv [0,1] \times \{0,1\} \times [0,1]$ and that $\bar{G}_0(W) \equiv_P r_{P_0}(A = 1|W)$ is bounded away from zero and one $Q_{0,W}$-almost surely (this is the case indeed). Then we can define model $\mathcal{M}$ as the set of all laws P on $\mathcal{O}$ such that

$$\bar{G}(W) \equiv \Pr_P(A = 1|W)$$

is bounded away from zero and one Q_W-almost surely, where Q_W is the marginal law of W under P.

Let us also define generically $\bar{Q}$ as

$$\bar{Q}(A,W) \equiv \mathrm{E}_P(Y|A,W).$$

Note how we have suppressed the dependence of $\bar{G}$ and $\bar{Q}$ on P for notational simplicity.

Central to our approach is viewing ψ_0 as the value at P_0 of the statistical mapping Ψ from $\mathcal{M}$ to $[0,1]$ characterized by

$$\begin{aligned}\Psi(P) &\equiv \int \left(\bar{Q}(1,w) - \bar{Q}(0,w)\right) dQ_W(w) \\ &= \mathrm{E}_P\left(\bar{Q}(1,W) - \bar{Q}(0,W)\right),\end{aligned} \tag{2.6}$$

a clear extension of (2.1).

[1] The empirical measure P_n is the law such that *(i)* X drawn from P_n takes its values in $\{X_1, \ldots, X_n\}$, and *(ii)* $X = X_i$ with probability n^{-1}

2.3.3 The value of the statistical mapping at another experiment

When we ran `example(tlrider)` earlier, we created an object called `another_experiment`:

```
another_experiment
#> A law for (W,A,Y) in [0,1] x {0,1} x [0,1].
#>
#> If the law is fully characterized, you can use method
#> 'sample_from' to sample from it.
#>
#> If you built the law, or if you are an _oracle_, you can
#> also use methods 'reveal' to reveal its relevant features
#> (QW, Gbar, Qbar, qY -- see '?reveal'), and 'alter' to change
#> some of them.
#>
#> If all its relevant features are characterized, you can
#> use methods 'evaluate_psi' to obtain the value of 'Psi' at
#> this law (see '?evaluate_psi') and 'evaluate_eic' to obtain
#> the efficient influence curve of 'Psi' at this law (see '?
#> evaluate_eic').
reveal(another_experiment)
#> $QW
#> function (x, min = 1/10, max = 9/10)
#> {
#>     stats::dunif(x, min = min, max = max)
#> }
#> <environment: 0xdc9bbb0>
#>
#> $Gbar
#> function (W)
#> {
#>     sin((1 + W) * pi/6)
#> }
#> <environment: 0xdc9bbb0>
#>
#> $Qbar
#> function (AW, h)
#> {
#>     A <- AW[, "A"]
#>     W <- AW[, "W"]
#>     expit(logit(A * W + (1 - A) * W^2) + h * 10 * sqrt(W) * A)
#> }
#> <environment: 0xdc9bbb0>
```

```
#>
#> $qY
#> function (obs, Qbar, shape1 = 4)
#> {
#>     AW <- obs[, c("A", "W")]
#>     QAW <- Qbar(AW)
#>     stats::gdbeta(Y, shape1 = shape1, shape2 = shape1 * (1 -
#>         QAW)/QAW)
#> }
#> <environment: 0xdc9bbb0>
#>
#> $sample_from
#> function (n, h)
#> {
#>     n <- R.utils::Arguments$getInteger(n, c(1, Inf))
#>     h <- R.utils::Arguments$getNumeric(h)
#>     Gbar <- another_experiment$.Gbar
#>     Qbar <- another_experiment$.Qbar
#>     params <- formals(another_experiment$.QW)
#>     W <- stats::runif(n, min = eval(params$min), max = eval(params$max))
#>     A <- stats::rbinom(n, size = 1, prob = Gbar(W))
#>     params <- formals(another_experiment$.qY)
#>     shape1 <- eval(params$shape1)
#>     QAW <- Qbar(cbind(A = A, W = W), h = h)
#>     Y <- stats::rbeta(n, shape1 = shape1, shape2 = shape1 * (1 -
#>         QAW)/QAW)
#>     obs <- cbind(W = W, A = A, Y = Y)
#>     return(obs)
#> }
#> <environment: 0xdc9bbb0>
(two_obs_another_experiment <- sample_from(another_experiment, 2, h = 0))
#>           W A     Y
#> [1,] 0.585 1 0.507
#> [2,] 0.347 1 0.345
```

By taking an oracular look at the output of `reveal(another_experiment)`, we discover that the law $\Pi_0 \in \mathcal{M}$ encoded by default (*i.e.*, with `h=0`) in `another_experiment` differs starkly from P_0.

However, the parameter $\Psi(\Pi_0)$ is well defined. Straightforward algebra shows that $\Psi(\Pi_0) = 59/300$. The numeric computation below confirms the equality.

```r
(psi_Pi_zero <- evaluate_psi(another_experiment, h = 0))
#> [1] 0.197
```

2.4 ✪ Alternative statistical mapping

We now resume the exercize of Section 2.2. Like we did in Section 2.3, we introduce a generic version of the relevant features $q_{0,Y}$ and $Q_{0,Y}$. Specifically, we define $q_Y(y, A, W)$ to be the conditional density of Y given A and W, evaluated at y, that is implied by a generic $P \in \mathcal{M}$. Similarly, we use Q_Y to denote the corresponding cumulative distribution function.

The covariate-adjusted c-th quantile reward for action $a \in \{0, 1\}$, $\gamma_{0,a,c}$ (2.3), may be viewed as the value at P_0 of a mapping $\Gamma_{a,c}$ from $\mathcal{M}$ to $[0, 1]$ characterized by

$$\Gamma_{a,c}(P) = \inf \left\{ y \in]0, 1[: \int Q_Y(y, a, w) dQ_W(w) \geq c \right\}.$$

The difference in c-th quantile rewards, $\delta_{0,c}$ (2.5), may similarly be viewed as the value at P_0 of a mapping Δ_c from $\mathcal{M}$ to $[0, 1]$, characterized by

$$\Delta_c(P) \equiv \Gamma_{1,c}(P) - \Gamma_{0,c}(P).$$

1. Compute the numerical value of $\Gamma_{a,c}(\Pi_0)$ for $(a, c) \in \{0, 1\} \times \{1/4, 1/2, 3/4\}$ using the relevant features of `another_experiment`. Based on these results, report the numerical value of $\Delta_c(\Pi_0)$ for each $c \in \{1/4, 1/2, 3/4\}$.
2. Approximate the value of $\Gamma_{0,a,c}(\Pi_0)$ for $(a, c) \in \{0, 1\} \times \{1/4, 1/2, 3/4\}$ by drawing a large sample from the "ideal" data experiment and using empirical quantile estimates. Deduce from these results a numerical approximation to $\Delta_{0,c}(\Pi_0)$ for each $c \in \{1/4, 1/2, 3/4\}$. Confirm that your results closely match those obtained in the previous problem.
3. Building upon the code you wrote to solve the previous problem, construct a confidence interval with asymptotic level 95% for $\Delta_{0,c}(\Pi_0)$, with $c \in \{1/4, 1/2, 3/4\}$.

2.5 Representations

In Section 2.3, we reoriented our view of the target parameter to be that of a statistical functional of the law of the observed data. Specifically, we viewed the parameter as a function of specific features of the observed data law, namely Q_W and $\bar{Q}$.

2.5.1 Yet another representation

It is straightforward to show an equivalent representation of the parameter as

$$\begin{aligned}\psi_0 &= \int \frac{2a-1}{\ell\bar{G}_0(a,w)} y dP_0(w,a,y) \\ &= \mathrm{E}_{P_0}\left(\frac{2A-1}{\ell\bar{G}_0(A,W)}Y\right). \end{aligned} \tag{2.7}$$

Viewing again the parameter as a statistical mapping from $\mathcal{M}$ to $[0,1]$, it also holds that

$$\begin{aligned}\Psi(P) &= \int \frac{2a-1}{\ell\bar{G}(a,w)} y dP(w,a,y) \\ &= \mathrm{E}_{P}\left(\frac{2A-1}{\ell\bar{G}_0(A,W)}Y\right). \end{aligned} \tag{2.8}$$

2.5.2 From representations to estimation strategies

Our reason for introducing this alternative view of the target parameter will become clear when we discuss estimation of the target parameter. Specifically, the representations (2.1) and (2.7) naturally suggest different estimation strategies for ψ_0, as hinted in Section 2.3.1. The former suggests building an estimator of ψ_0 using estimators of $\bar{Q}_0$ and of $Q_{W,0}$. The latter suggests building an estimator of ψ_0 using estimators of $\ell\bar{G}_0$ and of P_0.

We return to these ideas in later sections.

2.6 ⚙ Alternative representation

1. ⊇ Show that for $a' = 0, 1$, $\gamma_{0,a',c}$ as defined in (2.3) can be equivalently expressed as

$$\inf\left\{z \in]0,1[: \int \frac{\mathbf{1}\{a=a'\}}{\ell\bar{G}(a',W)}\mathbf{1}\{y \le z\} dP_0(w,a,y) \ge c\right\}.$$

Section 3

Smoothness

3.1 Fluctuating smoothly

Within our view of the target parameter as a statistical mapping evaluated at the law of the experiment, it is natural to inquire of properties this functional enjoys. For example, we may be interested in asking how the value of $\Psi(P)$ changes as we consider laws that *get nearer to* P in $\mathcal{M}$. If small deviations from P_0 result in large changes in $\Psi(P_0)$, then we might hypothesize that it will be difficult to produce stable estimators of ψ_0. Fortunately, this turns out not to be the case for the mapping Ψ, and so we say that Ψ is a *smooth* statistical mapping.

To discuss how $\Psi(P)$ changes for distributions that *get nearer* to P in the model, we require a more concrete notion of *getting-nearness*. The notion hinges on fluctuations (or fluctuating models).

3.1.1 The `another_experiment` fluctuation

In Section 2.3.3, we discussed the nature of the object called `another_experiment` that was created when we ran `example(tlrider)`:

```
another_experiment
#> A law for (W,A,Y) in [0,1] x {0,1} x [0,1].
#>
#> If the law is fully characterized, you can use method
#> 'sample_from' to sample from it.
#>
#> If you built the law, or if you are an _oracle_, you can
#> also use methods 'reveal' to reveal its relevant features
#> (QW, Gbar, Qbar, qY -- see '?reveal'), and 'alter' to change
```

```
#> some of them.
#>
#> If all its relevant features are characterized, you can
#> use methods 'evaluate_psi' to obtain the value of 'Psi' at
#> this law (see '?evaluate_psi') and 'evaluate_eic' to obtain
#> the efficient influence curve of 'Psi' at this law (see '?
#> evaluate_eic').
```

The message is a little misleading. Indeed, `another_experiment` is not *a* law but, rather, a *collection* of laws indexed by a real-valued parameter `h`. This oracular statement (we built the object!) is evident when one looks again at the `sample_from` feature of `another_experiment`:

```
reveal(another_experiment)$sample_from
#> function(n, h) {
#>         ## preliminary
#>         n <- R.utils::Arguments$getInteger(n, c(1, Inf))
#>         h <- R.utils::Arguments$getNumeric(h)
#>         ## ## 'Gbar' and 'Qbar' factors
#>         Gbar <- another_experiment$.Gbar
#>         Qbar <- another_experiment$.Qbar
#>         ## sampling
#>         ## ## context
#>         params <- formals(another_experiment$.QW)
#>         W <- stats::runif(n, min = eval(params$min),
#>                           max = eval(params$max))
#>         ## ## action undertaken
#>         A <- stats::rbinom(n, size = 1, prob = Gbar(W))
#>         ## ## reward
#>         params <- formals(another_experiment$.qY)
#>         shape1 <- eval(params$shape1)
#>         QAW <- Qbar(cbind(A = A, W = W), h = h)
#>         Y <- stats::rbeta(n,
#>                           shape1 = shape1,
#>                           shape2 = shape1 * (1 - QAW) / QAW)
#>         ## ## observation
#>         obs <- cbind(W = W, A = A, Y = Y)
#>         return(obs)
#>       }
#> <bytecode: 0x7c7b780>
#> <environment: 0xdc9bbb0>
```

Let us call $\Pi_h \in \mathcal{M}$ the law encoded by `another_experiment` for a given `h` taken in $]-1, 1[$.

Note that

$$\mathcal{P} \equiv \{\Pi_h : h \in]-1, 1[\}$$

defines a collection of laws, *i.e.*, a statistical model.

We say that $\mathcal{P}$ is a *submodel* of $\mathcal{M}$ because $\mathcal{P} \subset \mathcal{M}$. Moreover, we say that this submodel is *through* Π_0 since $\Pi_h = \Pi_0$ when $h = 0$. We also say that $\mathcal{P}$ is a *fluctuation* of Π_0.

One could enumerate many possible submodels in $\mathcal{M}$ through Π_0. It turns out that all that matters for our purposes is the form of the submodel in a neighborhood of Π_0. We informally say that this local behavior describes the *direction* of a submodel through Π_0. We formalize this notion Section 3.3.

We now have a notion of how to move through the model space $P \in \mathcal{M}$ and can study how the value of the parameter changes as we move away from a law P. Above, we said that Ψ is a smooth parameter if it does not change "too much" as we move towards P in any particular direction. That is, we should hope that Ψ is differentiable along our submodel at P.

This idea too is formalized in Section 3.3. We now turn to illustrating this idea numerically.

3.1.2 Numerical illustration

The code below evaluates how the parameter changes for laws in $\mathcal{P}$, and approximates the derivative of the parameter along the submodel $\mathcal{P}$ at Π_0. Recall that the numerical value of $\Psi(\Pi_0)$ has already been computed and is stored in object `psi_Pi_zero`.

```
approx <- seq(-1, 1, length.out = 1e2)
psi_Pi_h <- sapply(approx, function(t) {
  evaluate_psi(another_experiment, h = t)
})
slope_approx <- (psi_Pi_h - psi_Pi_zero) / approx
slope_approx <- slope_approx[min(which(approx > 0))]
ggplot() +
  geom_point(data = data.frame(x = approx, y = psi_Pi_h), aes(x, y),
             color = "#CC6666") +
  geom_segment(aes(x = -1, y = psi_Pi_zero - slope_approx,
                   xend = 1, yend = psi_Pi_zero + slope_approx),
               arrow = arrow(length = unit(0.03, "npc")),
               color = "#9999CC") +
  geom_vline(xintercept = 0, color = "#66CC99") +
  geom_hline(yintercept = psi_Pi_zero, color = "#66CC99") +
  labs(x = "h", y = expression(Psi(Pi[h])))
```

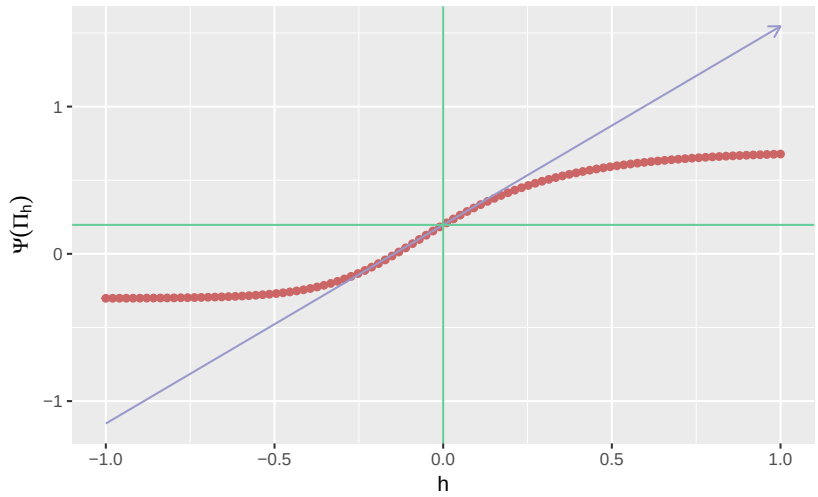


Figure 3.1: Evolution of statistical mapping Ψ along fluctuation $\{\Pi_h : h \in H\}$.

The dotted curve represents the function $h \mapsto \Psi(\Pi_h)$. The blue line represents the tangent to the previous curve at $h = 0$, which indeed appears to be differentiable around $h = 0$. In Section 3.4, we derive a closed-form expression for the slope of the blue curve.

3.2 ✪ Yet another experiment

1. Adapt the code from Problem 1 in Section 1.3 to visualize $w \mapsto \mathrm{E}_{\Pi_h}(Y|A = 1, W = w)$, $w \mapsto \mathrm{E}_{\Pi_h}(Y|A = 0, W = w)$, and $w \mapsto \mathrm{E}_{\Pi_h}(Y|A = 1, W = w) - \mathrm{E}_{\Pi_h}(Y|A = 0, W = w)$, for $h \in \{-1/2, 0, 1/2\}$.
2. Run the following chunk of code.

```
yet_another_experiment <- copy(another_experiment)
alter(yet_another_experiment,
      Qbar = function(AW, h){
        A <- AW[, "A"]
        W <- AW[, "W"]
        expit( logit( A * W + (1 - A) * W^2 ) +
               h * (2*A - 1) / ifelse(A == 1,
                                      sin((1 + W) * pi / 6),
                                      1 - sin((1 + W) * pi / 6)) *
               (Y - A * W + (1 - A) * W^2))
      })
```

3. Justify that `yet_another_fluctuation` characterizes another fluctuation of Π_0. Comment upon the similarities and differences between $\{\Pi_h : h \in]-1, 1[\}$ and $\{\Pi'_h : h \in$

$]-1,1[\}$.

4. Repeat Problem 1 above with Π'_h substituted for Π_h.
5. Re-produce Figure 3.1 for the $\{\Pi'_h : h \in]-1,1[\}$ fluctuation. Comment on the similarities and differences between the resulting figure and Figure 3.1. In particular, how does the behavior of the target parameter around $h = 0$ compare between laws Π_0 and Π'_0?

3.3 ≳ More on fluctuations and smoothness

3.3.1 Fluctuations

Let us now formally define what it means for statistical mapping Ψ to be smooth at every $P \in \mathcal{M}$. For every $h \in H \equiv]-M^{-1}, M^{-1}[$, we can define a law $P_h \in \mathcal{M}$ by setting $P_h \ll P$[1] and

$$\frac{dP_h}{dP} \equiv 1 + hs, \tag{3.1}$$

where $s : \mathcal{O} \to \mathbb{R}$ is a (measurable) function of O such that $s(O)$ is not equal to zero P-almost surely, $\mathrm{E}_P(s(O)) = 0$, and s bounded by M. We make the observation that

$$(i) \quad P_h|_{h=0} = P, \quad (ii) \quad \frac{d}{dh} \log \frac{dP_h}{dP}(O)\Big|_{h=0} = s(O). \tag{3.2}$$

Because of *(i)*, $\{P_h : h \in H\}$ is a submodel through P, also referred to as a *fluctuation* of P. The fluctuation is a one-dimensional submodel of $\mathcal{M}$ with univariate parameter $h \in H$. We note that *(ii)* indicates that the score of this submodel at $h = 0$ is s. Thus, we say that the fluctuation is *in the direction* of s.

Fluctuations of P do not necessarily take the same form as in (3.1). No matter how the fluctuation is built, for our purposes the most important feature of the fluctuation is its local shape in a neighborhood of P.

3.3.2 Smoothness and gradients

We are now prepared to provide a formal definition of smoothness of statistical mappings. We say that a statistical mapping Ψ is *smooth* at every $P \in \mathcal{M}$ if for each $P \in \mathcal{M}$, there exists a (measurable) function $D^*(P) : \mathcal{O} \to \mathbb{R}$ such that $\mathrm{E}_P(D^*(P)(O)) = 0$, $\mathrm{Var}_P(D^*(P)(O)) < \infty$, and, for every fluctuation $\{P_h : h \in H\}$ with score s at $h = 0$, the real-valued mapping $h \mapsto \Psi(P_h)$ is differentiable at $h = 0$, with a derivative equal to

[1] That is, P_h is dominated by P: if an event A satisfies $P(A) = 0$, then necessarily $P_h(A) = 0$ too.

$$\mathrm{E}_P\left(D^*(P)(O)s(O)\right). \tag{3.3}$$

The object $D^*(P)$ in (3.3) is called a gradient of Ψ at P.[2]

3.3.3 A Euclidean perspective

This terminology has a direct parallel to directional derivatives in the calculus of Euclidean geometry. Recall that if f is a differentiable mapping from $\mathbb{R}^p$ to $\mathbb{R}$, then the directional derivative of f at *a point* x (an element of $\mathbb{R}^p$) in *direction* u (a unit vector in $\mathbb{R}^p$) is the scalar product of the gradient of f and u. In words, the directional derivative of f at x can be represented as a scalar product of the direction that we approach x and the change of the function's value at x.

In the present problem, the law P is *the point* at which we evaluate the function Ψ, the score s of the fluctuation is *the direction* in which we approach the point, and the gradient describes the change in the function's value at the point.

3.3.4 The canonical gradient

In general, it is possible for many gradients to exist[3]. Yet, in the special case that the model is nonparametric, only a single gradient exists. The unique gradient is then referred to as *the canonical gradient* or *the efficient influence curve*, for reasons that will be clarified in Section 3.5. In the more general setting, the canonical gradient may be defined as the minimizer of $D \mapsto \mathrm{Var}_P(D(O))$ over the set of all gradients.

It is not difficult to check that the efficient influence curve of statistical mapping Ψ (2.6) at $P \in \mathcal{M}$ can be written as

$$\begin{aligned} D^*(P) &\equiv D_1^*(P) + D_2^*(P), \quad \text{where} \\ D_1^*(P)(O) &\equiv \bar{Q}(1,W) - \bar{Q}(0,W) - \Psi(P), \\ D_2^*(P)(O) &\equiv \frac{2A-1}{\ell\bar{G}(A,W)}(Y - \bar{Q}(A,W)). \end{aligned} \tag{3.4}$$

[2]Interestingly, if a fluctuation $\{P_h : h \in H\}$ satisfies (3.2) for a direction s such that $s \neq 0$, $\mathrm{E}_P(s(O)) = 0$ and $\mathrm{Var}_P(s(O)) < \infty$, then $h \mapsto \Psi(P_h)$ is still differentiable at $h = 0$ with a derivative equal to (3.3) beyond fluctuations of the form (3.1).

[3]This may be at first surprising given the parallel drawn in Section 3.3.3 to Euclidean geometry. However, it is important to remember that the model dictates fluctuations of P that are valid submodels with respect to the full model. In turn, this determines the possible directions from which we may approach P. Thus, depending on the direction, (3.3) may hold with different choices of D^*.

A `method` from package `tlrider` evaluates the efficient influence curve at a law described by an object of class `LAW`. It is called `evaluate_eic`. For instance, the next chunk of code evaluates the efficient influence curve $D^*(P_0)$ of Ψ (2.6) at $P_0 \in \mathcal{M}$ that is characterized by `experiment`:

```
eic_experiment <- evaluate_eic(experiment)
```

The efficient influence curve $D^*(P_0)$ is a function from $\mathbb{O}$ to $\mathbb{R}$. As such, it can be evaluated at the five independent observations drawn from P_0 in Section 1.2.2. This is what the next chunk of code does:

```
(eic_experiment(five_obs))
#> [1]  0.260  0.161 -0.387 -0.186  0.110
```

Finally, the efficient influence curve can be visualized as two images that represent $(w, y) \mapsto D^*(P_0)(w, a, y)$ for $a = 0, 1$, respectively:

```
crossing(w = seq(0, 1, length.out = 2e2),
     a = c(0, 1),
     y = seq(0, 1, length.out = 2e2)) %>%
  mutate(eic = eic_experiment(cbind(Y=y,A=a,W=w))) %>%
  ggplot(aes(x = w, y = y, fill = eic)) +
  geom_raster(interpolate = TRUE) +
  geom_contour(aes(z = eic), color = "white") +
  facet_wrap(~ a, nrow = 1,
             labeller = as_labeller(c(`0` = "a = 0", `1` = "a = 1"))) +
  labs(fill = expression(paste(D^"*", (P[0])(w,a,y))))
```

3.4 A fresh look at `another_experiment`

We can give a fresh look at Section 3.1.2 now.

3.4.1 Deriving the efficient influence curve

It is not difficult (though cumbersome) to verify that, up to a constant, $\{\Pi_h : h \in [-1, 1]\}$ is a fluctuation of Π_0 in the direction (in the sense of (3.1)) of

$$\sigma_0(O) \equiv -10\sqrt{W}A \times \beta_0(A, W) \times \left(\log(1 - Y) + \sum_{k=0}^{3} (k + \beta_0(A, W))^{-1} \right) + \text{constant}, \quad (3.5)$$

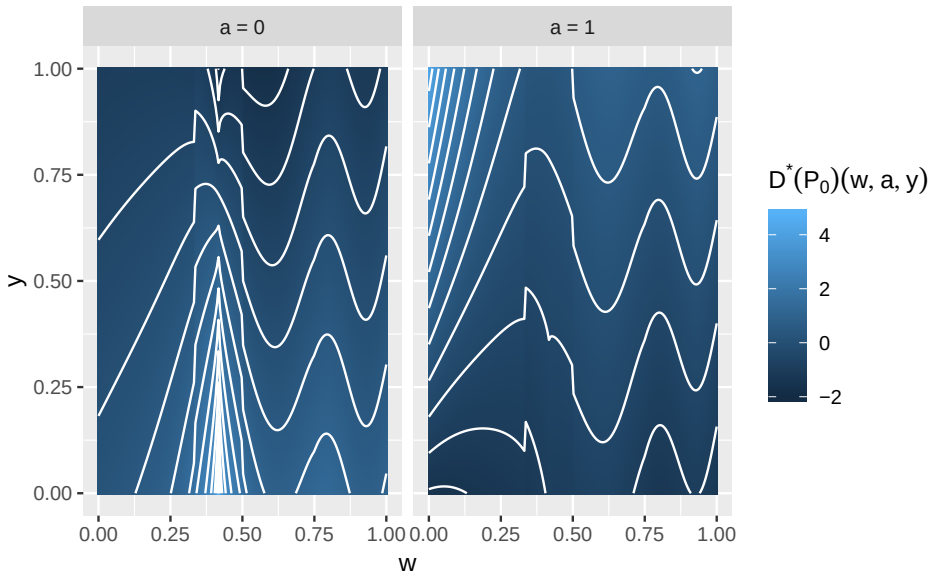


Figure 3.2: Visualizing the efficient influence curve $D^*(P_0)$ of Ψ (2.6) at P_0, the law described by `experiment`.

$$\text{where } \beta_0(A,W) \equiv \frac{1-\bar{Q}_{\Pi_0}(A,W)}{\bar{Q}_{\Pi_0}(A,W)}. \tag{3.6}$$

Consequently, the slope of the dotted curve in Figure 3.1 is equal to

$$\mathrm{E}_{\Pi_0}(D^*(\Pi_0)(O)\sigma_0(O)). \tag{3.7}$$

Since $D^*(\Pi_0)$ is centered under Π_0, knowing σ_0 up to a constant is not problematic.

3.4.2 Numerical validation

In the following code, we check the above fact numerically. When we ran `example(tlrider)`, we created a function `sigma0`. The function implements σ_0 defined in (3.6):

```
sigma0
#> function(obs, law = another_experiment) {
#>   ## preliminary
#>   Qbar <- get_feature(law, "Qbar", h = 0)
#>   QAW <- Qbar(obs[, c("A", "W")])
#>   params <- formals(get_feature(law, "qY", h = 0))
#>   shape1 <- eval(params$shape1)
#>   ## computations
```

```
#>   betaAW <- shape1 * (1 - QAW) / QAW
#>   out <- log(1 - obs[, "Y"])
#>   for (int in 1:shape1) {
#>     out <- out + 1/(int - 1 + betaAW)
#>   }
#>   out <- - out * shape1 * (1 - QAW) / QAW *
#>             10 * sqrt(obs[, "W"]) * obs[, "A"]
#>   ## no need to center given how we will use it
#>   return(out)
#> }
```

The next chunk of code approximates (3.7) pointwise and with a confidence interval of asymptotic level 95%:

```
eic_another_experiment <- evaluate_eic(another_experiment, h = 0)
obs_another_experiment <- sample_from(another_experiment, B, h = 0)
vars <- eic_another_experiment(obs_another_experiment) *
  sigma0(obs_another_experiment)

sd_hat <- sd(vars)
(slope_hat <- mean(vars))
#> [1] 1.35
(slope_CI <- slope_hat + c(-1, 1) * qnorm(1 - alpha / 2) * sd_hat / sqrt(B))
#> [1] 1.33 1.36
```

Equal to 1.349 (rounded to three decimal places — hereafter, all rounding will be to three decimal places as well), the first numerical approximation `slope_approx` is not too off!

3.5 ≳ Asymptotic linearity and statistical efficiency

3.5.1 Asymptotic linearity

Suppose that $O_1, \ldots, O_n$ are drawn independently from $P \in \mathcal{M}$. If an estimator ψ_n of $\Psi(P)$ can be written as

$$\psi_n = \Psi(P) + \frac{1}{n}\sum_{i=1}^{n} \mathrm{IC}(O_i) + o_P(1/\sqrt{n})$$

for some function $\mathrm{IC} : \mathcal{O} \to \mathbb{R}$ such that $\mathrm{E}_P(\mathrm{IC}(O)) = 0$ and $\mathrm{Var}_P(\mathrm{IC}(O)) < \infty$, then we say that ψ_n is *asymptotically linear* with *influence curve* IC. Asymptotically linear estimators are *weakly convergent*. Specifically, if ψ_n is asymptotically linear with influence curve IC, then

$$\sqrt{n}(\psi_n - \Psi(P)) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \mathrm{IC}(O_i) + o_P(1) \tag{3.8}$$

and, by the central limit theorem (recall that $O_1, \ldots, O_n$ are independent), $\sqrt{n}(\psi_n - \Psi(P))$ converges in law to a centered Gaussian distribution with variance $\mathrm{Var}_P(\mathrm{IC}(O))$.

3.5.2 Influence curves and gradients

As it happens, influence curves of regular[4] estimators are intimately related to gradients. In fact, if ψ_n is a regular, asymptotically linear estimator of $\Psi(P)$ with influence curve IC, then it must be true that Ψ is a smooth at P and that IC is a gradient of Ψ at P.

3.5.3 Asymptotic efficiency

Now recall that, in Section 3.3.4, we defined the canonical gradient as the minimizer of $D \mapsto \mathrm{Var}_P(D(O))$ over the set of all gradients. Therefore, if ψ_n is a regular, asymptotically linear estimator of $\Psi(P)$ (built from n independent observations drawn from P), then the asymptotic variance of $\sqrt{n}(\psi_n - \Psi(P))$ cannot be smaller than the variance of the canonical gradient of Ψ at P, *i.e.*,

$$\mathrm{Var}_P(D^*(P)(O)). \tag{3.9}$$

In other words, (3.9) is the lower bound on the asymptotic variance of *any* regular, asymptotically linear estimator of $\Psi(P)$. This bound is referred to as the *Cramér-Rao bound*. Any regular estimator that achieves this variance bound is said to be *asymptotically efficient* at P. Because the canonical gradient is the influence curve of an asymptotically efficient estimator, it is often referred to as the *efficient influence curve*.

3.6 ✪ Cramér-Rao bounds

1. What does the following chunk do?

```
obs <- sample_from(experiment, B)
(cramer_rao_hat <- var(eic_experiment(obs)))
#> [1] 0.287
```

[4] We can view ψ_n as the by product of an algorithm $\widehat{\Psi}$ trained on independent observations $O_1, \ldots, O_n$ drawn from P. We say that the estimator is regular at P if, for any direction $s \neq 0$ such that $\mathrm{E}_P(s(O)) = 0$ and $\mathrm{Var}_P(s(O)) < \infty$ and fluctuation $\{P_h : h \in H\}$ satisfying (3.2), the estimator $\psi_{n,1/\sqrt{n}}$ of $\Psi(P_{1/\sqrt{n}})$ obtained by training $\widehat{\Psi}$ on independent observations O_1, ..., O_n drawn from $P_{1/\sqrt{n}}$ is such that $\sqrt{n}(\psi_{n,1/\sqrt{n}} - \Psi(P_{1/\sqrt{n}}))$ converges in law to a limit that does not depend on s.

2. Same question about this one.

```
obs_another_experiment <- sample_from(another_experiment, B, h = 0)
(cramer_rao_Pi_zero_hat <-
   var(eic_another_experiment(obs_another_experiment)))
#> [1] 0.098
```

3. With a large independent sample drawn from $\Psi(P_0)$ (or $\Psi(\Pi_0)$), is it possible to construct a regular estimator ψ_n of $\Psi(P_0)$ (or $\Psi(\Pi_0)$) such that the asymptotic variance of $\sqrt{n}$ times ψ_n minus its target be smaller than the Cramér-Rao bound?
4. Is it easier to estimate $\Psi(P_0)$ or $\Psi(\Pi_0)$ (from independent observations drawn from either law)? In what sense? (Hint: you may want to compute a ratio.)

Section 4

Double-robustness

4.1 Linear approximations of parameters

4.1.1 From gradients to estimators

We learned in Section 3 that the stochastic behavior of a regular, asymptotically linear estimator of $\Psi(P)$ can be characterized by its influence curve. Moreover, we said that this influence curve must in fact be a gradient of Ψ at P.

In this section, we show that the converse is also true: given a gradient D^* of Ψ at P, under so-called *regularity conditions*, it is possible to construct an estimator with influence curve equal to $D^*(P)$. This fact will suggest concrete strategies for generating efficient estimators of smooth parameters. We take here the first step towards generating such estimators: linearizing the parameter.

4.1.2 A Euclidean perspective

As in Section 3.3.3, drawing a parallel to Euclidean geometry is helpful. We recall that if f is a differentiable mapping from $\mathbb{R}^p$ to $\mathbb{R}$, then a Taylor series approximates f at a point $x_0 \in \mathbb{R}^p$:

$$f(x_0) \approx f(x) + \langle (x_0 - x), \nabla f(x) \rangle,$$

where x is a point in $\mathbb{R}^p$, $\nabla f(x)$ is the gradient of f evaluated at x and $\langle u, v \rangle$ is the scalar product of $u, v \in \mathbb{R}^p$. As the squared distance $\|x - x_0\|^2 = \langle x - x_0, x - x_0 \rangle$ between x and x_0 decreases, the *linear approximation* to $f(x_0)$ becomes more accurate.

4.1.3 The remainder term

Returning to the present problem with this in mind, we find that indeed a similar approximation strategy may be applied.

For clarity, let us introduce a new shorthand notation. For any measurable function f of the observed data O, we may write from now on $Pf \equiv \mathrm{E}_P(f(O))$. One may argue that the notation is valuable beyong the gain of space. For instance, (3.8)

$$\sqrt{n}(\psi_n - \Psi(P)) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \mathrm{IC}(O_i) + o_P(1)$$

can be rewritten as

$$\sqrt{n}(\psi_n - \Psi(P)) = \sqrt{n}(P_n - P)\mathrm{IC} + o_P(1),$$

thus suggesting more clearly the importance of the so-called *empirical process* $\sqrt{n}(P_n - P)$.

In particular, if Ψ is smooth uniformly over directions, then for any given $P \in \mathcal{M}$, we can write

$$\Psi(P_0) = \Psi(P) + (P_0 - P)D^*(P) - \mathrm{Rem}_{P_0}(P), \tag{4.1}$$

where $\mathrm{Rem}_{P_0}(P)$ is a *remainder term* satisfying that

$$\frac{\mathrm{Rem}_{P_0}(P)}{d(P, P_0)} \to 0 \text{ as } d(P, P_0) \to 0,$$

where d is a measure of discrepancy for distributions in $\mathcal{M}$. Note that (4.1) can be equivalently written as

$$\Psi(P_0) = \Psi(P) + \mathrm{E}_{P_0}(D^*(P)(O)) - \mathrm{E}_P(D^*(P)(O)) - \mathrm{Rem}_{P_0}(P).$$

The remainder term formalizes the notion that if P is *close* to P_0 (*i.e.*, if $d(P, P_0)$ is small), then the linear approximation of $\Psi(P_0)$ is more accurate.

4.1.4 Expressing the remainder term as a function of the relevant features

The equations for the definition of the parameter (2.6), form of the canonical gradient (3.4), and linearization of parameter (4.1) combine to determine the remainder:

$$\mathrm{Rem}_{P_0}(P) \equiv \Psi(P) - \Psi(P_0) - (P_0 - P)D^*(P) \tag{4.2}$$

hence

$$\mathrm{Rem}_{P_0}(P) = \mathrm{E}_{P_0}\Bigg[\left(\bar{G}_0(W) - \bar{G}(W)\right) \times \left(\frac{\bar{Q}_0(1,W) - \bar{Q}(1,W)}{\ell\bar{G}(1,W)} + \frac{\bar{Q}_0(0,W) - \bar{Q}(0,W)}{\ell\bar{G}(0,W)} \right) \Bigg]. \quad (4.3)$$

Acting as oracles, we can compute explicitly the remainder term $\mathrm{Rem}_{P_0}(P)$. The `evaluate_remainder` method makes it very easy (simply run `?evaluate_remainder` to see the man page of the method):

```
(evaluate_remainder(experiment, experiment))
#> [1] 0
(rem <- evaluate_remainder(experiment, another_experiment,
                           list(list(), list(h = 0))))
#> [1] 0.199
```

We recover the equality $\mathrm{Rem}_{P_0}(P_0) = 0$, which is fairly obvious given (4.1). In addition, we learn that $\mathrm{Rem}_{P_0}(\Pi_0)$ equals 0.199. In the next subsection, we invite you to make better acquaintance with the remainder term by playing around with it numerically.

4.2 ✪ The remainder term

1. Compute numerically $\mathrm{Rem}_{\Pi_0}(\Pi_h)$ for $h \in [-1, 1]$ and plot your results. What do you notice?
2. ≥ Approximate $\mathrm{Rem}_{P_0}(\Pi_0)$ numerically without relying on method `evaluate_remainder` and compare the value you get with that of `rem`. (Hint: use (4.2) and a large sample of observations drawn independently from P_0.)

4.3 ≥ Double-robustness

4.3.1 The key property

Let us denote by $\|f\|_P^2$ the square of the $L^2(P)$-norm of any function f from $\mathbb{O}$ to $\mathbb{R}$ *i.e.*, using a recently introduced notation, $\|f\|_P^2 \equiv Pf^2$. For instance, $\|\bar{Q}_1 - \bar{Q}_0\|_P$ or $\|\bar{G}_1 - \bar{G}_0\|_P$ is a distance separating the features $\bar{Q}_1$ and $\bar{Q}_0$ or $\bar{G}_1$ and $\bar{G}_0$.

The efficient influence curve $D^*(P)$ at $P \in \mathcal{M}$ enjoys a rather remarkable property: it is *double-robust*. Specifically, for every $P \in \mathcal{M}$, the remainder term $\mathrm{Rem}_{P_0}(P)$ satisfies

$$\text{Rem}_{P_0}(P)^2 \leq \|\bar{Q} - \bar{Q}_0\|_P^2 \times \|(\bar{G} - \bar{G}_0)/\ell\bar{G}_0\|_P^2, \tag{4.4}$$

where $\bar{Q}$ and $\bar{G}$ are the counterparts under P to $\bar{Q}\_\{0\}\$$ and $\bar{G}_0$. The proof consists in a straightforward application of the Cauchy-Schwarz inequality to the right-hand side expression in (4.2).

4.3.2 Its direct consequence

It may not be clear yet why (4.4) is an important property, and why D^* is said *double-robust* because of it. To answer the latter question, let us consider a law $P \in \mathcal{M}$ such that *either* $\bar{Q} = \bar{Q}_0$ *or* $\bar{G} = \bar{G}_0$.

It is then the case that *either* $\|\bar{Q} - \bar{Q}_0\|_P = 0$ *or* $\|\bar{G} - \bar{G}_0\|_P = 0$. Therefore, in light of (4.4), it also holds that $\text{Rem}_{P_0}(P) = 0$.[1] It thus appears that (4.1) simplifies to

$$\begin{aligned}\Psi(P_0) &= \Psi(P) + (P_0 - P)D^*(P) \\ &= \Psi(P) + P_0 D^*(P),\end{aligned}$$

where the second equality holds because $PD^*(P) = 0$ for all $P \in \mathcal{M}$ by definition of $D^*(P)$.

It is now clear that for such a law $P \in \mathcal{M}$, $\Psi(P) = \Psi(P_0)$ is equivalent to

$$P_0 D^*(P) = 0. \tag{4.5}$$

Most importantly, in words, if P solves the so-called P_0-specific efficient influence curve equation (4.5) and if, in addition, P has the same $\bar{Q}$-feature *or* $\bar{G}$-feature as P_0, then $\Psi(P) = \Psi(P_0)$.

The conclusion is valid no matter how P may differ from P_0 otherwise, hence the notion of being *double-robust*. This property is useful to build consistent estimators of $\Psi(P)$, as we shall see in Section 5.

4.4 ✪ Double-robustness

1. Go back to Problem 1 in 4.2. In light of Section 4.3, what is happening?
2. Create a copy of `experiment` and replace its `Gbar` feature with some other function of W (see `?copy`, `?alter` and Problem 2 in Section 3.2). Call P' the element of model $\mathcal{M}$ thus characterized. Can you guess the values of $\text{Rem}_{P_0}(P')$, $\Psi(P')$ and $P_0 D^*(P')$? Support your argument.

[1] This also trivially follows from (4.3).

3. Add more exercizes?

Section 5

Inference

5.1 Where we stand

In the previous sections, we analyzed our target parameter and presented relevant theory for understanding the statistical properties of certain types of estimators of the parameter. The theory is also relevant for building and comparing a variety of estimators.

We assume from now on that we have available a sample $O_1, \ldots, O_B$ of independent observations drawn from P_0. This is literally the case!, and the observations are stored in `obs` that we created in Section 3.6.

```
iter <- 1e2
```

Equal to 0.1 million, the sample size `B` is very large. We will in fact use 100 disjoint subsamples composed of n independent observations among $O_1, \ldots, O_B$, where n equals `B/iter`, *i.e.*, 1000. We will thus be in a position to investigate the statistical properties of every estimation procedure by replicating it independently 100 times.

5.2 Where we go

The following sections explore different statistical paths to inferring ψ_0 or, rather (though equivalently), $\Psi(P_0)$.

- Section 6 present a simple inference strategy. It can be carried out in situations where $\bar{G}_0$ is already known to the statistician.
- Section 7 discusses the estimation of some infinite-dimensional features of P_0. The resulting estimators are later used to infer ψ_0.
- Section...

Section 6

A simple inference strategy

6.1 A cautionary detour

Let us introduce first the following estimator:

$$\begin{aligned}\psi_n^a &\equiv \frac{\mathrm{E}_{P_n}(AY)}{\mathrm{E}_{P_n}(A)} - \frac{\mathrm{E}_{P_n}((1-A)Y)}{\mathrm{E}_{P_n}(1-A)} \\ &= \frac{\sum_{i=1}^n \mathbf{1}\{A_i = Y_i = 1\}}{\sum_{i=1}^n \mathbf{1}\{A_i = 1\}} - \frac{\sum_{i=1}^n \mathbf{1}\{A_i = 0, Y_i = 1\}}{\sum_{i=1}^n \mathbf{1}\{A_i = 0\}}.\end{aligned} \tag{6.1}$$

It estimates

$$\begin{aligned}\Phi(P_0) &\equiv \frac{\mathrm{E}_{P_0}(AY)}{\mathrm{E}_{P_0}(A)} - \frac{\mathrm{E}_{P_0}((1-A)Y)}{\mathrm{E}_{P_0}(1-A)} \\ &= \mathrm{E}_{P_0}(Y|A=1) - \mathrm{E}_{P_0}(Y|A=0).\end{aligned}$$

We seize this opportunity to demonstrate numerically the obvious fact that ψ_n^a *does not* estimate $\Psi(P_0)$ because, in general, $\Psi(P_0)$ and $\Phi(P_0)$ differ. This is apparent in the following alternative expression of $\Phi(P_0)$:

$$\begin{aligned}\Phi(P_0) &= \mathrm{E}_{P_0}\left(\mathrm{E}_{P_0}(Y \mid A, W)|A=1)\right) - \mathrm{E}_{P_0}\left(\mathrm{E}_{P_0}(Y \mid A, W)|A=0\right) \\ &= \int \bar{Q}_0(1,w) dP_{0,W|A=1}(w) - \int \bar{Q}_0(0,w) dP_{0,W|A=0}(w).\end{aligned}$$

Contrast the above equalities and (2.1). In the latter, the outer integral is against the marginal law of W under P_0. In the former, the outer integrals are respectively against the conditional laws of W given $A = 1$ and $A = 0$ under P_0.

6.2 ✪ Delta-method

Consider the next chunk of code:

```
compute_irrelevant_estimator <- function(obs) {
  Y <- pull(obs, Y)
  A <- pull(obs, A)
  psi_n <- mean(A * Y) / mean(A) - mean((1 - A) * Y) / mean(1 - A)
  Var_n <- cov(cbind(A * Y, A, (1 - A) * Y, (1 - A)))
  phi_n <- c(1 / mean(A), -mean(A * Y) / mean(A)^2,
             -1 / mean(1 - A),
             mean((1 - A) * Y) / mean(1 - A)^2)
  var_n <- as.numeric(t(phi_n) %*% Var_n %*% phi_n)
  sig_n <- sqrt(var_n / nrow(obs))
  tibble(psi_n = psi_n, sig_n = sig_n)
}
```

Function `compute_irrelevant_estimator` computes the estimator ψ_n^a (6.1) based on the data set in `obs`.

Introduce $X_n \equiv n^{-1} \sum_{i=1}^n (A_i Y_i, A_i, (1 - A_i) Y_i, 1 - A_i)^\top$ and $X \equiv (AY, A, (1 - A)Y, 1 - A)^\top$. It happens that X_n is asymptotically Gaussian: as n goes to infinity,

$$\sqrt{n} \left(X_n - \mathrm{E}_{P_0}(X) \right)$$

converges in law to the centered Gaussian law with covariance matrix

$$V_0 \equiv \mathrm{E}_{P_0} \left((X - \mathrm{E}_{P_0}(X)) \times (X - \mathrm{E}_{P_0}(X))^\top \right).$$

Let $f : \mathbb{R} \times \mathbb{R}^* \times \mathbb{R} \times \mathbb{R}^* \to$ be given by $f(r, s, t, u) = r/s - t/u$. The function is differentiable.

1. Check that $\psi_n^a = f(X_n)$. Point out to the line where ψ_n^a is computed in the body of `compute_irrelevant_estimator`. Also point out to the line where the above asymptotic variance of X_n is estimated with its empirical counterpart, say V_n.
2. ⊇ Argue how the delta-method yields that $\sqrt{n}(\psi_n^a - \Phi(P_0))$ converges in law to the centered Gaussian law with a variance that can be estimated with

$$v_n^a \equiv \nabla f(X_n) \times V_n \times \nabla f(X_n)^\top. \tag{6.2}$$

3. Check that the gradient ∇f of f is given by $\nabla f(r,s,t,u) \equiv (1/s, -r/s^2, -1/u, t/u^2)$. Point out to the line where the asymptotic variance of ψ_n^a is estimated.

6.3 IPTW estimator assuming the mechanism of action known

6.3.1 A simple substitution estimator

Let us assume for a moment that we know $\bar{G}_0$. This would have been the case indeed if P_0 were a controlled experiment. Note that, on the contrary, assuming $\bar{Q}_0$ known would be difficult to justify.

```
Gbar <- get_feature(experiment, "Gbar")
```

The alternative expression (2.7) suggests to estimate $\Psi(P_0)$ with

$$\psi_n^b \equiv \mathrm{E}_{P_n}\left(\frac{2A-1}{\ell\bar{G}_0(A,W)}Y\right) \tag{6.3}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{2A_i-1}{\ell\bar{G}_0(A_i,W_i)}Y_i\right). \tag{6.4}$$

Note how P_n is substituted for P_0 in (6.4) relative to (2.7). This justifies that we call ψ_n^b a *substitution estimator* (for the same reason, ψ_n^a is a substitution estimator of $\Phi(P_0)$). It is also dubbed an IPTW (inverse probability of treatment weighted) estimator because of the denominators $\ell\bar{G}_0(A_i,W_i)$ in its definition.[1]

In Section 8.2, we develop another IPTW estimator that does not assume that $\bar{G}_0$ is known beforehand.

6.3.2 Elementary statistical properties

It is easy to check that ψ_n^b estimates $\Psi(P_0)$ consistently, but this is too little to request from an estimator of ψ_0. Better, ψ_n^b also satisfies a central limit theorem: $\sqrt{n}(\psi_n^b - \psi_0)$ converges in law to a centered Gaussian law with asymptotic variance

$$v^b \equiv \mathrm{Var}_{P_0}\left(\frac{2A-1}{\ell\bar{G}_0(A,W)}Y\right),$$

[1] We could have used the alternative expression IPAW, where A (like action) is substituted for T (like treatment).

where v^b can be consistently estimated by its empirical counterpart

$$v_n^b \equiv \text{Var}_{P_n}\left(\frac{2A-1}{\ell\bar{G}_0(A,W)}Y\right) \tag{6.5}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{2A_i-1}{\ell\bar{G}_0(A_i,W_i)}Y_i - \psi_n^b\right)^2. \tag{6.6}$$

We investigate *empirically* the statistical behavior of ψ_n^b in Section 6.3.3.

6.3.3 Empirical investigation

The next chunk of code investigates the empirical behaviors of estimators ψ_n^a and ψ_n^b. As explained in Section 5, we first make `iter` data sets out of the `obs` data set (second line), then build the estimators on each of them (fourth and fifth lines). After the first series of commands the object `psi_hat_ab`, a `tibble`, contains 200 rows and four columns. For each smaller data set (identified by its `id`), two rows contain the values of either ψ_n^a and $\sqrt{v_n^a}/\sqrt{n}$ (if `type` equals `a`) or ψ_n^b and $\sqrt{v_n^b}/\sqrt{n}$ (if `type` equals `b`).

After the second series of commands, the object `psi_hat_ab` contains, in addition, the values of the recentered (with respect to ψ_0) and renormalized $\sqrt{n}/\sqrt{v_n^a}(\psi_n^a - \psi_0)$ and $\sqrt{n}/\sqrt{v_n^b}(\psi_n^b - \psi_0)$, where v_n^a (6.2) and v_n^b (6.5) estimate the asymptotic variances of ψ_n^a and ψ_n^b, respectively. Finally, `bias_ab` reports amounts of bias (at the renormalized scale).

```
psi_hat_ab <- obs %>% as_tibble() %>%
  mutate(id = (seq_len(n()) - 1) %% iter) %>%
  nest(-id, .key = "obs") %>%
  mutate(est_a = map(obs, ~ compute_irrelevant_estimator(.)),
         est_b = map(obs, ~ compute_iptw(as.matrix(.), Gbar))) %>%
  gather(`est_a`, `est_b`, key = "type", value = "estimates") %>%
  extract(type, "type", "_([ab])$") %>%
  unnest(estimates) %>% select(-obs)

(psi_hat_ab)
#> # A tibble: 200 x 4
#>      id type  psi_n  sig_n
#>   <dbl> <chr> <dbl>  <dbl>
#> 1     0 a     0.130 0.0174
#> 2     1 a     0.126 0.0180
#> 3     2 a     0.112 0.0161
#> 4     3 a     0.116 0.0164
#> 5     4 a     0.110 0.0187
```

```
#> 6     5 a     0.140 0.0178
#> # ... with 194 more rows

psi_hat_ab <- psi_hat_ab %>%
  group_by(id) %>%
  mutate(clt = (psi_n - psi_zero) / sig_n)

(psi_hat_ab)
#> # A tibble: 200 x 5
#> # Groups:   id [100]
#>      id type  psi_n  sig_n   clt
#>   <dbl> <chr> <dbl>  <dbl> <dbl>
#> 1     0 a     0.130 0.0174  2.71
#> 2     1 a     0.126 0.0180  2.40
#> 3     2 a     0.112 0.0161  1.78
#> 4     3 a     0.116 0.0164  2.01
#> 5     4 a     0.110 0.0187  1.42
#> 6     5 a     0.140 0.0178  3.19
#> # ... with 194 more rows

(bias_ab <- psi_hat_ab %>%
   group_by(type) %>% summarise(bias = mean(clt)))
#> # A tibble: 2 x 2
#>   type    bias
#>   <chr>  <dbl>
#> 1 a     1.53
#> 2 b     0.0922

fig_bias_ab <- ggplot() +
  geom_line(aes(x = x, y = y),
            data = tibble(x = seq(-3, 3, length.out = 1e3),
                          y = dnorm(x)),
            linetype = 1, alpha = 0.5) +
  geom_density(aes(clt, fill = type, colour = type),
               psi_hat_ab, alpha = 0.1) +
  geom_vline(aes(xintercept = bias, colour = type),
             bias_ab, size = 1.5, alpha = 0.5)

fig_bias_ab +
  labs(y = "",
       x = expression(paste(sqrt(n/v[n])^{list(a, b)})*
                           (psi[n]^{list(a, b)} - psi[0])))
```

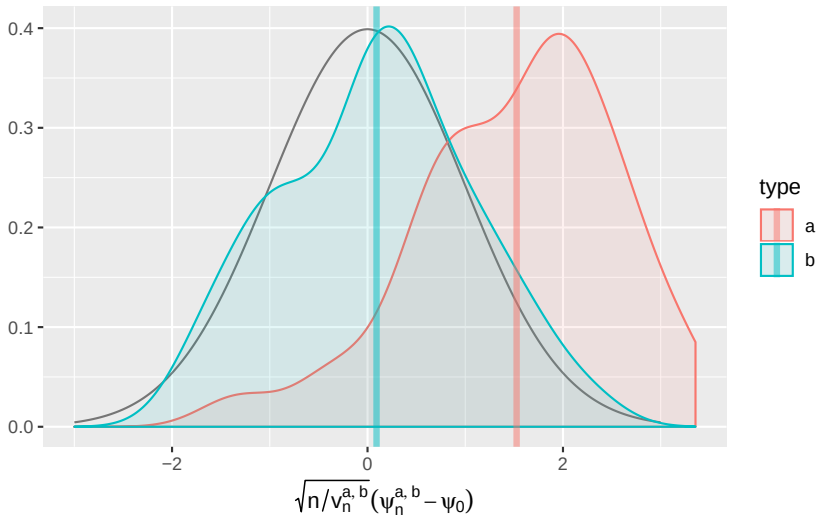


Figure 6.1: Kernel density estimators of the law of two estimators of ψ_0 (recentered with respect to ψ_0, and renormalized), one of them misconceived (a), the other assuming that $\bar{G}_0$ is known (b). Built based on `iter` independent realizations of each estimator.

By the above chunk of code, the averages of $\sqrt{n/v_n^a}(\psi_n^a - \psi_0)$ and $\sqrt{n/v_n^b}(\psi_n^b - \psi_0)$ computed across the realizations of the two estimators are respectively equal to 1.526 and 0.092 (see `bias_ab`). Interpreted as amounts of bias, those two quantities are represented by vertical lines in Figure 6.1. The red and blue bell-shaped curves represent the empirical laws of ψ_n^a and ψ_n^b (recentered with respect to ψ_0, and renormalized) as estimated by kernel density estimation. The latter is close to the black curve, which represents the standard normal density.

Section 7

Nuisance parameters

7.1 Anatomy of an expression

From now, all the inference strategies that we will present unfold in two or three stages. For all of them, the first stage consists in estimating a selection of features of the law P_0 of the experiment. Specifically, the features are chosen among $Q_{0,W}$ (the marginal law of W under P_0), $\bar{G}_0$ (the conditional probability that $A = 1$ given W under P_0) and $\bar{Q}_0$ (the conditional mean of Y given A and W under P_0).

In this context, because they are not the parameter of primary interest (*i.e.*, they are not the real-values feature $\Psi(P_0)$), they are often referred to as *nuisance parameters* of P_0. The unflaterring expression conveys the notion that their estimation is merely an intermediate step along our path towards an inference of the target parameter.

As for the reason why $Q_{0,W}$, $\bar{G}_0$ and $\bar{Q}_0$ are singled out, it is because of their role in the definition of Ψ and the efficient influence curve $D^*(P_0)$.

7.2 An algorithmic stance

In general, we can view an estimator of any feature f_0 of P_0 as the output of an algorithm $\widehat{\mathcal{A}}$ that maps any element of

$$\mathcal{M}^{\text{empirical}} \equiv \left\{ \frac{1}{m} \sum_{i=1}^{m} \text{Dirac}(o_i) : m \geq 1, o_1, \ldots, o_m \in [0,1] \times \{0,1\} \times [0,1] \right\}$$

to the set $\mathcal{F}$ where f_0 is know to live. Here, $\mathcal{M}^{\text{empirical}}$ can be interpreted as the set of all possible empirical measures summarizing the outcomes of any number of replications of the experiment P_0. In particular, P_n belongs to this set.

The `tlrider` package includes such template algorithms for the estimation of $Q_{0,W}$, $\bar{G}_0$ and $\bar{Q}_0$. We illustrate how they work and their use in the next sections.

7.3 QW

For instance, `estimate_QW` is an algorithm $\widehat{\mathcal{A}}_{Q_W}$ for the estimation of the marginal law of W under P_0 (to see its man page, simply run `?estimate_QW`). It is a map from $\mathcal{M}^{\text{empirical}}$ to the set of laws on $[0,1]$. The following chunk of code estimates $Q_{0,W}$ based on the $n = 1000$ first observations in `obs`:

```
QW_hat <- estimate_QW(head(obs, 1e3))
```

It is easy to sample independent observations from `QW_hat`. To do so, we create an object of class `LAW` then set its marginal law of W to that described by `QW_hat` and specify its `sample_from` feature:

```
empirical_experiment <- LAW()
alter(empirical_experiment, QW = QW_hat)
alter(empirical_experiment, sample_from = function(n) {
  QW <- get_feature(empirical_experiment, "QW")
  W <- sample(pull(QW, "value"), n, prob = pull(QW, "weight"))
  cbind(W = W, A = NA, Y = NA)
})
W <- sample_from(empirical_experiment, 1e3) %>% as.tibble
W %>% ggplot() +
  geom_histogram(aes(x = W, y = stat(density)), bins = 40) +
  stat_function(fun = get_feature(experiment, "QW"), col = "red")
```

Note that all the Ws sampled from `QW_hat` fall in the set $\{W_1, \ldots, W_n\}$ of observed Ws in `obs` (an obvious fact given the definition of the `sample_from` feature of `empirical_experiment`:

```
(length(intersect(pull(W, W), head(obs[, "W"], 1e3))))
#> [1] 1000
```

This is because `estimate_QW` estimates $Q_{0,W}$ with its empirical counterpart, *i.e.*,

$$\frac{1}{n}\sum_{i=1}^{n} \text{Dirac}(W_i).$$

7.4 Gbar

Another template algorithm is built-in into `tlrider`: `estimate_Gbar` (to see its man page, simply run `?estimate_Gbar`). Unlike `estimate_QW`, `estimate_Gbar` needs further specifi-

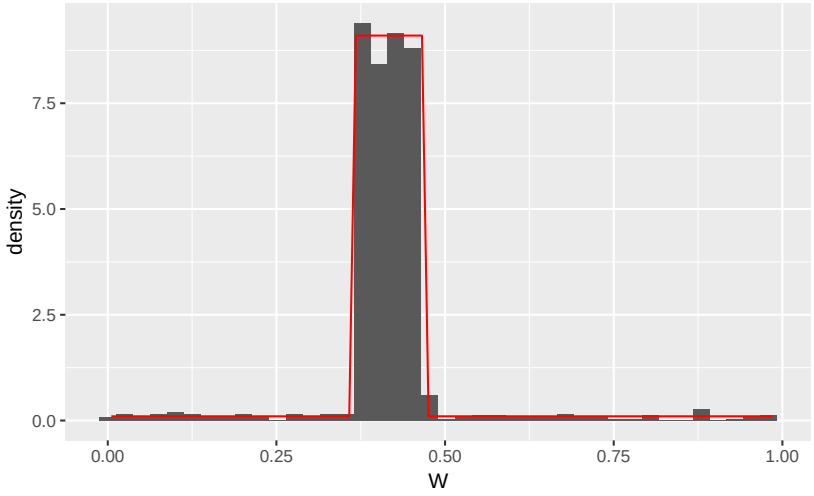


Figure 7.1: Histogram representing 1000 observations drawn independently from `QW_hat`. The superimposed red curve is the true density of $Q_{0,W}$.

cation of the algorithm. The package also includes examples of such specifications.

There are two sorts of specifications, of which we say that they are either *working model-based* or *machine learning-based*. We discuss the former sort in the next subsection. The latter sort is discussed in Section 7.6.

7.4.1 Working model-based algorithms

Let us take a look at `working_model_G_one` for instance:

```
working_model_G_one
#> $model
#> function (...)
#> {
#>     trim_glm_fit(glm(family = binomial(), ...))
#> }
#> <environment: 0x83535e0>
#>
#> $formula
#> A ~ I(W^0.5) + I(abs(W - 5/12)^0.5) + I(W^1) + I(abs(W - 5/12)^1) +
#>     I(W^1.5) + I(abs(W - 5/12)^1.5)
#> <environment: 0x83535e0>
#>
#> $type_of_preds
#> [1] "response"
```

```
#>
#> attr(,"ML")
#> [1] FALSE
```

and focus on its `model` and `formula` attributes. The former relies on the `glm` and `binomial` functions from `base R`, and on `trim_glm_fit` (which removes information that we do not need from the standard output of `glm`, simply run `?trim_glm_fit` to see the function's man page). The latter is a `formula` that characterizes what we call a *working model* for $\bar{G}_0$.

In words, by using `working_model_G_one` we implicitly choose the so-called logistic (or negative binomial) loss function L_a given by

$$-L_a(f)(A,W) \equiv A \log f(W) + (1-A)\log(1-f(W)) \tag{7.1}$$

for any function $f : [0,1] \to [0,1]$ paired with the working model

$$\mathcal{F}_1 \equiv \{f_\theta : \theta \in \mathbb{R}^5\}$$

where, for any $\theta \in \mathbb{R}^5$,

$$\text{logit}\, f_\theta(W) \equiv \theta_0 + \sum_{j=1}^{4} \theta_j W^{j/2}.$$

We acted as oracles when we specified the working model: it is *well-specified*, *i.e.*, it happens that $\bar{G}_0$ is the unique minimizer of the risk entailed by L_a over $\mathcal{F}_1$:

$$\bar{G}_0 = \underset{f_\theta \in \mathcal{F}_1}{\arg\min}\, \mathrm{E}_{P_0}\left(L_a(f_\theta)(A,W)\right).$$

Therefore, the estimator $\bar{G}_n$ obtained by minimizing the empirical risk

$$\mathrm{E}_{P_n}\left(L_a(f_\theta)(A,W)\right) = \frac{1}{n}\sum_{i=1}^{n} L_a(f_\theta)(A_i,W_i)$$

over $\mathcal{F}_1$ estimates $\bar{G}_0$ consistently.

Of course, it is seldom certain in real life that the target feature, here $\bar{G}_0$, belongs to the working model.[1] Suppose for instance that we choose a small finite-dimensional working model $\mathcal{F}_2$ without acting as an oracle. Then consistency certainly fails to hold. However, if $\bar{G}_0$ can nevertheless be *projected* unambiguously onto $\mathcal{F}_2$ (an assumption that cannot be checked), then the estimator might converge to the projection.

[1] In fact, if one knows nothing about the feature, then it is *certain* that it does not belong to whichever small finite-dimensional working model we may come up with.

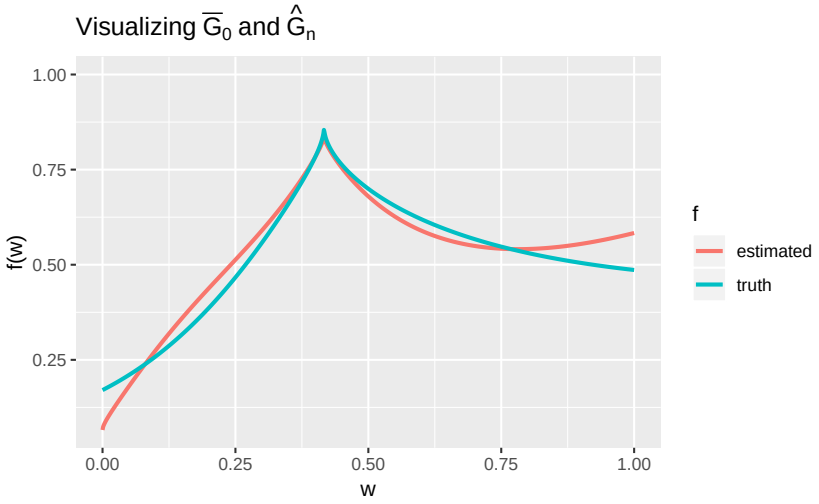


Figure 7.2: Comparing $\bar{G}_n \equiv \widehat{\mathcal{A}}_{\bar{G},1}(P_n)$ and $\bar{G}_0$. The estimator is consistent because the algorithm relies on a working model that is correctly specified.

7.4.2 Visualization

To illustrate the use of the algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$ obtained by combining `estimate_Gbar` and `working_model_G_one`, let us estimate $\bar{G}_0$ based on the first $n = 1000$ observations in `obs`:

```
Gbar_hat <- estimate_Gbar(head(obs, 1e3), algorithm = working_model_G_one)
```

Using `compute_Gbar_hat_W`[2] (simply run `?compute_Gbar_hat_W` to see its man page) makes it is easy to compare visually the estimator $\bar{G}_n \equiv \widehat{\mathcal{A}}_{\bar{G},1}(P_n)$ with its target $\bar{G}0$:

```
tibble(w = seq(0, 1, length.out = 1e3)) %>%
  mutate("truth" = Gbar(w),
         "estimated" = compute_Gbar_hatW(w, Gbar_hat)) %>%
  gather("f", "value", -w) %>%
  ggplot() + geom_line(aes(x = w, y = value, color = f), size = 1) +
  labs(y = "f(w)",
       title = bquote("Visualizing" ~ bar(G)[0] ~ "and" ~ hat(G)[n])) +
  ylim(NA, 1)
```

[2]See also the companion function `compute_lGbar_hat_AW` (run `?compute_lGbar_hat_AW` to see its man page.

7.5 ⚙ `Qbar`, working model-based algorithms

A third template algorithm is built-in into `tlrider`: `estimate_Qbar` (to see its man page, simply run `?estimate_Qbar`). Like `estimate_Gbar`, `estimate_Qbar` needs further specification of the algorithm. The package also includes examples of such specifications, which can also be either working model-based (see Section 7.4) or machine learning-based (see Sections 7.6 and 7.7).

There are built-in specifications similar to `working_model_G_one`, *e.g.*,

```
working_model_Q_one
#> $model
#> function (...)
#> {
#>     trim_glm_fit(glm(family = binomial(), ...))
#> }
#> <environment: 0x83535e0>
#>
#> $formula
#> Y ~ A * (I(W^0.5) + I(W^1) + I(W^1.5))
#> <environment: 0x83535e0>
#>
#> $type_of_preds
#> [1] "response"
#>
#> attr(,"ML")
#> [1] FALSE
#> attr(,"stratify")
#> [1] FALSE
```

1. Drawing inspiration from Section 7.4, comment upon and use the algorithm $\widehat{\mathcal{A}}_{\bar{Q},1}$ obtained by combining `estimate_Gbar` and `working_model_Q_one`.

7.6 `Qbar`

7.6.1 `Qbar`, machine learning-based algorithms

We explained how algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$ is based on a working model (and *you* did for $\widehat{\mathcal{A}}_{\bar{Q},1}$). It is not the case that all algorithms are based on working models in the same (admittedly rather narrow) sense. We propose to say that those algorithms that are not based on working models like $\widehat{\mathcal{A}}_{\bar{G},1}$, for instance, are instead *machine learning-based*.

Typically, machine learning-based algorithms are more data-adaptive; they rely on larger

working models, and/or fine-tune parameters that must be calibrated, *e.g.* by cross-validation. Furthermore, they call for being stacked, *i.e.*, combined by means of another outer algorithm (involving cross-validation) into a more powerful machine learning-based *meta-algorithm*. The super learning methodology is a popular stacking algorithm.

We will elaborate further on this important topic in another forthcoming part. Here, we merely illustrate the concept with two specifications built-in into `tlrider`. Based on the *k-nearest neighbors* non-parametric estimating methodology, the first one is discussed in the next subsection. Based on *boosted trees*, another non-parametric estimating methodology, the second one is used in the exercize that follows the next subsection.

7.6.2 `Qbar`, kNN algorithm

Algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ is obtained by combining `estimate_Qbar` and `kknn_algo`. The training of $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ (*i.e.*, the making of the output $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}(P_n)$ is implemented based on function `caret::train` of the `caret` (classification and regression training) package (to see its man page, simply run `?caret::train`). Some additional specifications are provided in `kknn_grid` and `kknn_control`.

In a nutshell, $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ estimates $\bar{Q}_0(1,\cdot)$ and $\bar{Q}_0(0,\cdot)$ separately. Each of them is estimated by applying the k-nearest neighbors methodology as it is implemented in function `kknn::train.kknn` from the `kknn` package (to see its man page, simply run `?kknn::train.kknn`).[3] The following chunk of code trains algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ on P_n:

```
Qbar_hat_kknn <- estimate_Qbar(head(obs, 1e3),
                               algorithm = kknn_algo,
                               trControl = kknn_control,
                               tuneGrid = kknn_grid)
```

Using `compute_Qbar_hat_AW` (simply run `?compute_Qbar_hat_AW` to see its man page) makes it is easy to compare visually the estimator $\bar{Q}_{n,\text{kNN}} \equiv \widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}(P_n)$ with its target $\bar{Q}0$, see Figure 7.3.

```
fig <- tibble(w = seq(0, 1, length.out = 1e3),
              truth_1 = Qbar(cbind(A = 1, W = w)),
              truth_0 = Qbar(cbind(A = 0, W = w)),
              kNN_1 = compute_Qbar_hatAW(1, w, Qbar_hat_kknn),
              kNN_0 = compute_Qbar_hatAW(0, w, Qbar_hat_kknn))
```

[3]Specifically, argument `kmax` (maximum number of neighbors considered) is set to 5, argument `distance` (parameter of the Minkowski distance) is set to 2, and argument `kernel` is set to `gaussian`. The best value of k is chosen between 1 and `kmax` by leave-one-out. No outer cross-validation is needed.

7.6.3 `Qbar`, boosted trees algorithm

Algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$ is obtained by combining `estimate_Qbar` and `bstTree_algo`. The training of $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$ (*i.e.*, the making of the output $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}(P_n)$ is implemented based on function `caret::train` of the `caret` package. Some additional specifications are provided in `bstTree_grid` and `bstTree_control`.

In a nutshell, $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$ estimates $\bar{Q}_0(1,\cdot)$ and $\bar{Q}_0(0,\cdot)$ separately. Each of them is estimated by boosted trees as implemented in function `bst::bst` from the `bst` (gradient boosting) package (to see its man page, simply run `?bst::bst`).[4] The following chunk of code trains algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$ on P_n, and reveals what are the optimal fine-tune parameters for the estimation of $\bar{Q}_0(1,\cdot)$ and $\bar{Q}_0(0,\cdot)$:

```
Qbar_hat_trees <- estimate_Qbar(head(obs, 1e3),
                                algorithm = bstTree_algo,
                                trControl = bstTree_control,
                                tuneGrid = bstTree_grid)

Qbar_hat_trees %>% dplyr::filter(a == "one") %>% pull(fit) %>%
  capture.output %>% tail(3) %>% str_wrap(width = 60) %>% cat
#> The final values used for the model were mstop = 20,
#> maxdepth = 1 and nu = 0.2.

Qbar_hat_trees %>% dplyr::filter(a == "zero") %>% pull(fit) %>%
  capture.output %>% tail(3) %>% str_wrap(width = 60) %>% cat
#> The final values used for the model were mstop = 30,
#> maxdepth = 1 and nu = 0.1.
```

We can compare visually the estimators $\bar{Q}_{n,\text{kNN}}$, $\bar{Q}_{n,\text{trees}} \equiv \widehat{\mathcal{A}}_{\bar{Q},\text{trees}}(P_n)$ with its target $\bar{Q}0$, see Figure 7.3. In summary, $\bar{Q}_{n,\text{kNN}}$ is rather good, though very versatile at the vincinity of the break points. As for $\bar{Q}_{n,\text{trees}}$, it does not seem to capture the shape of its target.

```
fig %>%
  mutate(trees_1 = compute_Qbar_hatAW(1, w, Qbar_hat_trees),
         trees_0 = compute_Qbar_hatAW(0, w, Qbar_hat_trees)) %>%
  gather("f", "value", -w) %>%
  extract(f, c("f", "a"), "([^_]+)_([01]+)") %>%
  mutate(a = paste0("a=", a)) %>%
  ggplot +
  geom_line(aes(x = w, y = value, color = f), size = 1) +
```

[4] Specifically, argument `mstop` (number of boosting iterations for prediction) is one among 10, 20 and 30; argument `nu` (stepsize of the shrinkage parameter) is one among 0.1 and 0.2; argument `maxdepth` (maximum depth of the base learner, a tree) is one among 1, 2 and 5. An outer 10-fold cross-validation is carried out to select the best combination of fine-tune parameters.

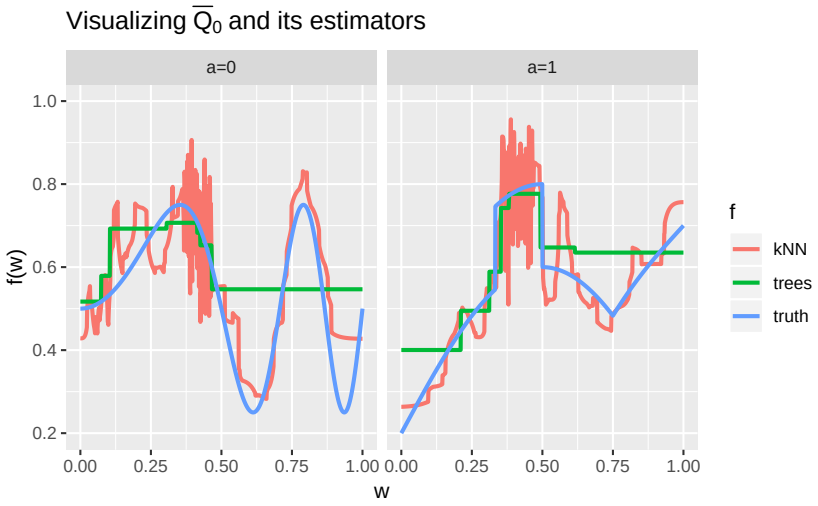


Figure 7.3: Comparing to their target two (machine learning-based) estimators of $\bar{Q}_0$, one based on the k-nearest neighbors and the other on boosted trees.

```
  labs(y = "f(w)",
       title = bquote("Visualizing" ~ bar(Q)[0] ~ "and its estimators")) +
  ylim(NA, 1) +
  facet_wrap(~ a)
```

7.7 ⚙ ≥ `Qbar`, machine learning-based algorithms

1. Using `estimate_Q`, make your own machine learning-based algorithm for the estimation of $\bar{Q}_0$.
2. Train your algorithm on the same data set as $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ and $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$. If, like $\widehat{\mathcal{A}}_{\bar{Q},\text{trees}}$, your algorithm includes a fine-tuning procedure, comment upon the optimal, data-driven specification.
3. Plot your estimators of $\bar{Q}_0(1,\cdot)$ and $\bar{Q}_0(0,\cdot)$ on Figure 7.3.

Section 8

Two "naive" plug-in inference strategies

8.1 Why "naive"?

In this section, we present and discuss two plug-in strategies for the inference of $\Psi(P_0)$. In light of Section 7.1, both unfold in *two* stages. During the first stage, some features among $Q_{0,W}$, $\bar{G}_0$ and $\bar{Q}_0$ (the Ψ-specific nuisance parameters, see Section 7) are estimated. During the second stage, the estimators are substituted for their theoretical counterparts in the definition of $\Psi(P_0)$, thus yielding estimators of $\Psi(P_0)$.

Although the strategies sound well conceived, a theoretical analysis reveals that they lack a third stage trying to correct an inherent flaw. They are thus said *naive*. The analysis and a first *modus operandi* are presented in Section **??**.

8.2 IPTW estimator

8.2.1 Construction and computation

In Section 6.3, we developed an IPTW substitution estimator, ψ_n^b, *assuming that* we knew $\bar{G}_0$ beforehand. What if we did not? Obviously, we could estimate it and substitute the estimator of $\ell\bar{G}_0$ for $\ell\bar{G}_0$ in (6.4).

Let $\widehat{\mathcal{A}}_{\bar{G}}$ be an algorithm designed for the estimation of $\bar{G}_0$ (see Section 7.4). We denote by $\bar{G}_n \equiv \widehat{\mathcal{A}}_{\bar{G}}(P_n)$ the output of the algorithm trained on P_n, and by $\ell\bar{G}_n$ the resulting (empirical) function given by

$$\ell\bar{G}_n(A,W) \equiv A\bar{G}_n(W) + (1-A)(1-\bar{G}_n(W)).$$

In light of (6.4), introduce

$$\psi_n^c \equiv \frac{1}{n}\sum_{i=1}^{n}\left(\frac{2A_i-1}{\ell\bar{G}_n(A_i,W_i)}Y_i\right).$$

From a computational point of view, the `tlrider` package makes it easy to build ψ_n^c. Recall that

```
compute_iptw(head(obs, 1e3), Gbar)
```

implements the computation of ψ_n^b based on the $n = 1000$ first observations stored in `obs`, using the true feature $\bar{G}_0$ stored in `Gbar`, see Section 6.3.3 and the construction of `psi_hat_ab`. Similarly,

```
Ghat <- estimate_Gbar(head(obs, 1e3), working_model_G_one)
compute_iptw(head(obs, 1e3), wrapper(Ghat)) %>% pull(psi_n)
#> [1] 0.0707
```

implements *(i)* the estimation of $\bar{G}_0$ with $\bar{G}_n$/`Ghat` using algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$ (first line) then *(ii)* the computation of ψ_n^c (second line), both based on the same observations as above.

Note how we use function `wrapper` (simply run `?wrapper` to see its man page).

8.2.2 Elementary statistical properties

Because $\bar{G}_n$ minimizes the empirical risk over a finite-dimensional, identifiable, and **well-specified** working model, $\sqrt{n}(\psi_n^c-\psi_0)$ converges in law to a centered Gaussian law. Moreover, the asymptotic variance of $\sqrt{n}(\psi_n^c-\psi_0)$ is **conservatively**[1] estimated with

$$\begin{aligned} v_n^c &\equiv \mathrm{Var}_{P_n}\left(\frac{2A-1}{\ell\bar{G}_n(A,W)}Y\right) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{2A_i-1}{\ell\bar{G}_n(A_i,W_i)}Y_i-\psi_n^c\right)^2. \end{aligned}$$

We investigate *empirically* the statistical behavior of ψ_n^c in Section 8.2.3. For an analysis of the reason why v_n^c is a conservative estimator of the asymptotic variance of $\sqrt{n}(\psi_n^c-\psi_0)$, see here.

Before proceeding, let us touch upon what would have happened if we had used a less amenable algorithm $\widehat{\mathcal{A}}_{\bar{G}}$. For instance, $\widehat{\mathcal{A}}_{\bar{G}}$ could still be well-specified[2] but so *versatile/complex* (as opposed to being based on well-behaved, finite-dimensional parametric

[1] In words, v_n^c converges to an upper-bound of the true asymptotic variance.

[2] Well-specified *e.g.* in the sense that the target $\bar{G}_0$ of $\widehat{\mathcal{A}}_{\bar{G}}$ belongs to the closure of the algorithm's image $\widehat{\mathcal{A}}_{\bar{G}}(\mathcal{M}^{\text{empirical}})$ or, in other words, can be approximated arbitrarily well by an output of the algorithm.

model) that the estimator $\bar{G}_n$, though still consistent, would converge slowly to its target. Then, root-n consistency would fail to hold. Or $\widehat{\mathcal{A}}_{\bar{G}}$ could be mis-specified and there would be no guarantee at all that the resulting estimator ψ_n^c be even consistent.

8.2.3 Empirical investigation

Let us compute ψ_n^c on the same `iter = 100` independent samples of independent observations drawn from P_0 as in Section 6.3. As explained in Sections 5 and 6.3.3, we first make `iter` data sets out of the `obs` data set (third line), then train algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$ on each of them (fifth to seventh lines). After the first series of commands the object `learned_features_fixed_sample_size`, a `tibble`, contains 100 rows and three columns.

We created `learned_features_fixed_sample_size` to store the estimators of $\bar{G}_0$ for future use. We will at a later stage enrich the object, for instance by adding to it estimators of $\bar{Q}_0$ obtained by training different algorithms on each smaller data set.

In the second series of commands, the object `psi_hat_abc` is obtained by adding to `psi_hat_ab` (see Section 6.3.3) an 100 by four `tibble` containing notably the values of ψ_n^c and $\sqrt{v_n^c}/\sqrt{n}$ computed by calling `compute_iptw`. The object also contains the values of the recentered (with respect to ψ_0) and renormalized $\sqrt{n}/\sqrt{v_n^c}(\psi_n^c - \psi_0)$. Finally, `bias_abc` reports amounts of bias (at the renormalized scale).

```
learned_features_fixed_sample_size <-
  obs %>% as_tibble() %>%
  mutate(id = (seq_len(n()) - 1) %% iter) %>%
  nest(-id, .key = "obs") %>%
  mutate(Gbar_hat =
           map(obs,
               ~ estimate_Gbar(., algorithm = working_model_G_one)))

psi_hat_abc <-
  learned_features_fixed_sample_size %>%
  mutate(est_c =
           map2(obs, Gbar_hat,
                ~ compute_iptw(as.matrix(.x), wrapper(.y, FALSE)))) %>%
  unnest(est_c) %>% select(-Gbar_hat, -obs) %>%
  mutate(clt = (psi_n - psi_zero) / sig_n,
         type = "c") %>%
  full_join(psi_hat_ab)

(bias_abc <- psi_hat_abc %>%
   group_by(type) %>% summarise(bias = mean(clt)))
#> # A tibble: 3 x 2
```

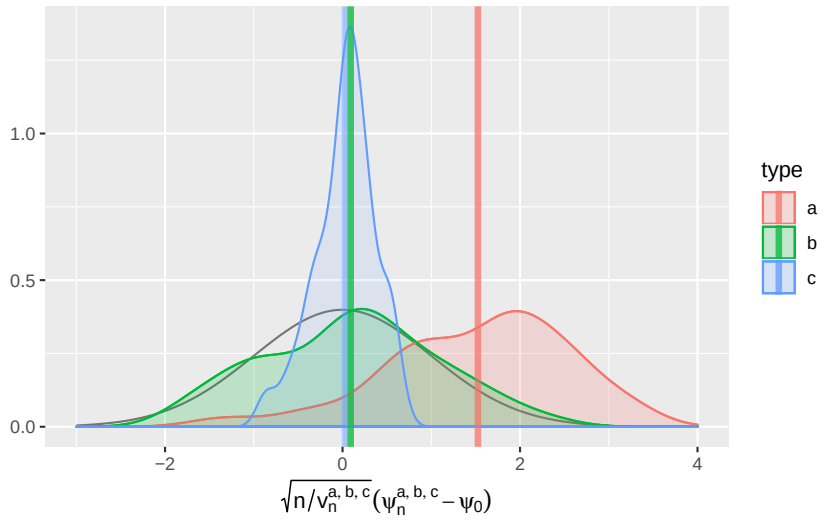


Figure 8.1: Kernel density estimators of the law of three estimators of ψ_0 (recentered with respect to ψ_0, and renormalized), one of them misconceived (a), one assuming that $\bar{G}_0$ is known (b) and one that hinges on the estimation of $\bar{G}_0$ (c). The present figure includes Figure 6.1 (but the colors differ). Built based on `iter` independent realizations of each estimator.

```
#>   type    bias
#>   <chr>  <dbl>
#> 1 a      1.53
#> 2 b      0.0922
#> 3 c      0.0254
```

By the above chunk of code, the average of $\sqrt{n/v_n^c}(\psi_n^c - \psi_0)$ computed across the realizations is equal to 0.025 (see `bias_abc`). In words, the average bias of ψ_n^c is of the same magnitude as that of ψ_n^b despite the fact that the construction of ψ_n^c hinges on the estimation of $\bar{G}_0$ (based on the well-specified algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$).

We represent the empirical laws of the recentered (with respect to ψ_0) and renormalized ψ_n^a, ψ_n^b and ψ_n^c in Figures 8.1 (kernel density estimators) and 8.2 (quantile-quantile plots).

```
fig_bias_ab +
  geom_density(aes(clt, fill = type, colour = type), psi_hat_abc, alpha = 0.1) +
  geom_vline(aes(xintercept = bias, colour = type),
             bias_abc, size = 1.5, alpha = 0.5) +
  xlim(-3, 4) +
  labs(y = "",
       x = expression(paste(sqrt(n/v[n])^{list(a, b, c)})*
                            (psi[n]^{list(a, b, c)} - psi[0])))
```

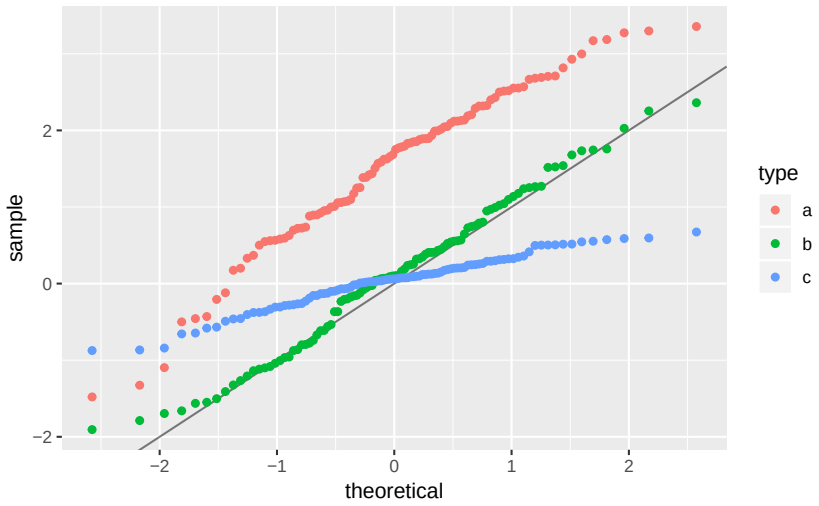


Figure 8.2: Quantile-quantile plot of the standard normal law against the empirical laws of three estimators of ψ_0, one of them misconceived (a), one assuming that $\bar{G}_0$ is known (b) and one that hinges on the estimation of $\bar{G}_0$ (c). Built based on `iter` independent realizations of each estimator.

```
ggplot(psi_hat_abc, aes(sample = clt, fill = type, colour = type)) +
  geom_abline(intercept = 0, slope = 1, alpha = 0.5) +
  geom_qq(alpha = 1)
```

Figures 8.1 and 8.2 confirm that ψ_n^c behaves as well as ψ_n^b in terms of bias — but remember that we acted as oracles when we chose the well-specified algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$. They also corroborate that v_n^c, the estimator of the asymptotic variance of $\sqrt{n}(\psi_n^c - \psi_0)$, is conservative: for instance, the corresponding bell-shaped blue curve is too much concentrated around its axis of symmetry.

The actual asymptotic variance of $\sqrt{n}(\psi_n^c - \psi_0)$ can be estimated with the empirical variance of the `iter` replications of the construction of ψ_n^c.

```
(emp_sig_n <- psi_hat_abc %>% filter(type == "c") %>%
   summarize(sd(psi_n)) %>% pull)
#> [1] 0.0191
(summ_sig_n <- psi_hat_abc %>% filter(type == "c") %>% select(sig_n) %>%
   summary)
#>      sig_n
#>  Min.   :0.0531
#>  1st Qu.:0.0550
#>  Median :0.0557
#>  Mean   :0.0561
```

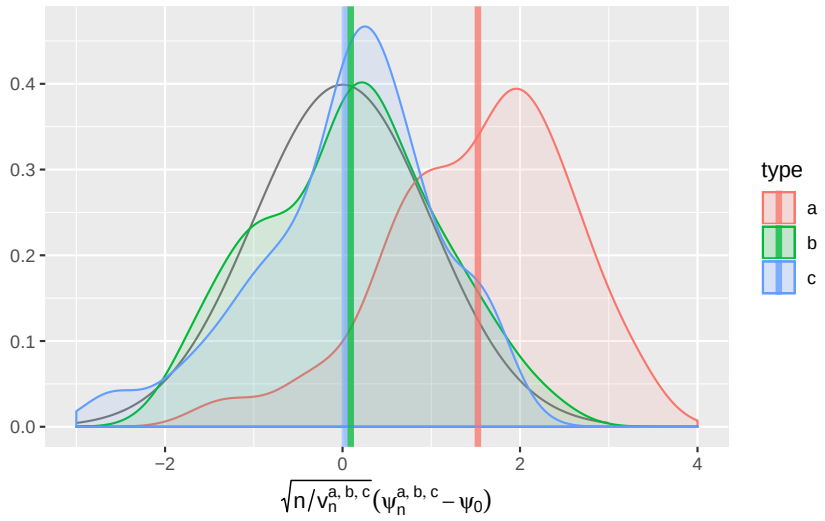


Figure 8.3: Kernel density estimators of the law of three estimators of ψ_0 (recentered with respect to ψ_0, and renormalized), one of them misconceived (a), one assuming that $\bar{G}_0$ is known (b) and one that hinges on the estimation of $\bar{G}_0$ **and an estimator of the asymptotic variance computed across the replications** (c). The present figure includes Figure 6.1 (but the colors differ) and it should be compared to Figure 8.2. Built based on `iter` independent realizations of each estimator.

```
#>  3rd Qu.:0.0570
#>  Max.   :0.0616
```

The empirical standard deviation is approximately 2.939 times smaller than the average *estimated* standard deviation. The estimator is conservative indeed! Furthermore, note the better fit with the density of the standard normal density of the kernel density estimator of the law of $\sqrt{n}(\psi_n^c - \psi_0)$ **renormalized with** `emp_sig_n`.

```
clt_c <- psi_hat_abc %>% filter(type == "c") %>%
  mutate(clt = clt * sig_n /  emp_sig_n)

fig_bias_ab +
  geom_density(aes(clt, fill = type, colour = type), clt_c, alpha = 0.1) +
  geom_vline(aes(xintercept = bias, colour = type),
             bias_abc, size = 1.5, alpha = 0.5) +
  xlim(-3, 4) +
  labs(y = "",
       x = expression(paste(sqrt(n/v[n]^{list(a, b, c)})*
                              (psi[n]^{list(a, b, c)} - psi[0]))))
```

Workaround. In a real world data-analysis, one could correct the estimation of the asymp-

totic variance of $\sqrt{n}(\psi_n^c - \psi_0)$. We could for instance derive the influence function as it is described here and use the corresponding influence function-based estimator of the variance. Or one could rely on the bootstrap.[3] This, however, would only make sense if one knew for sure that the algorithm for the estimation of $\bar{G}_0$ is well-specified.

8.3 ✪ Investigating further the IPTW inference strategy

1. Building upon the chunks of code devoted to the repeated computation of ψ_n^b and its companion quantities, construct confidence intervals for ψ_0 of (asymptotic) level 95%, and check if the empirical coverage is satisfactory. Note that if the coverage was exactly 95%, then the number of confidence intervals that would contain ψ_0 would follow a binomial law with parameters `iter` and `0.95`, and recall that function `binom.test` performs an exact test of a simple null hypothesis about the probability of success in a Bernoulli experiment against its three one-sided and two-sided alternatives.

2. Discuss what happens when the dimension of the (still well-specified) working model grows. Start with the built-in working model `working_model_G_two`. The following chunk of code re-defines `working_model_G_two`

```
## make sure '1/2' and '1' belong to 'powers'
powers <- rep(seq(1/4, 3, by = 1/4), each = 2)
working_model_G_two <- list(
  model = function(...) {trim_glm_fit(glm(family = binomial(), ...))},
  formula = stats::as.formula(
    paste("A ~",
          paste(c("I(W^", "I(abs(W - 5/12)^"),
                powers,
                sep = "", collapse = ") + "),
          ")")
  ),
  type_of_preds = "response"
)
attr(working_model_G_two, "ML") <- FALSE
```

3. Play around with argument `powers` (making sure that `1/2` and `1` belong to it), and plot graphics similar to those presented in Figures 8.1 and 8.2.

4. Discuss what happens when the working model is mis-specified. You could use the built-in working model `working_model_G_three`.

[3] That is, replicate the construction of ψ_n^c many times based on data sets obtained by resampling from the original data set, then estimate the asymptotic variance with the empirical variance of ψ_n^c computed across the replications.

5. Repeat the analysis developed in response to problem 1 above but for ψ_n^c. What can you say about the coverage of the confidence intervals?

6. ⊇ (Follow-up to problem 5). Implement the bootstrap procedure evoked at the end of Section 8.2.3. Repeat the analysis developed in response to problem 1. Compare your results with those to problem 5.

7. ⊇ Is it legitimate to infer the asymptotic variance of ψ_n^c with v_n^c when one relies on a very data-adaptive/versatile algorithm to estimate $\bar{G}_0$?

8.4 G-computation estimator

8.4.1 Construction and computation

Let $\widehat{\mathcal{A}}_{Q_W}$ be an algorithm designed for the estimation of $Q_{0,W}$ (see Section 7.3). We denote by $Q_{n,W} \equiv \widehat{\mathcal{A}}_{Q_W}(P_n)$ the output of the algorithm trained on P_n.

Let $\widehat{\mathcal{A}}_{\bar{Q}}$ be an algorithm designed for the estimation of $\bar{Q}_0$ (see Section 7.6). We denote by $\bar{Q}_n \equiv \widehat{\mathcal{A}}_{\bar{Q}}(P_n)$ the output of the algorithm trained on P_n.

Equation (2.1) suggests the following, simple estimator of $\Psi(P_0)$:

$$\psi_n \equiv \int \left(\bar{Q}_n(1,w) - \bar{Q}_n(0,w)\right) dQ_{n,W}(w). \tag{8.1}$$

In words, this estimator is implemented by first regressing Y on (A,W), then by marginalizing with respect to the estimated law of W. The resulting estimator is referred to as a *G-computation* (or *standardization*) estimator.

From a computational point of view, the `tlrider` package makes it easy to build the G-computation estimator. Recall that we have already estimated the marginal law $Q_{0,W}$ of W under P_0 by training the algorithm $\widehat{\mathcal{A}}_{Q_W}$ as it is implemented in `estimate_QW` on the $n = 1000$ first observations in `obs` (see Section 7.3):

```
QW_hat <- estimate_QW(head(obs, 1e3))
```

Recall that $\widehat{\mathcal{A}}_{\bar{Q},1}$ is the algorithm for the estimation of $\bar{Q}_0$ as it is implemented in `estimate_Qbar` with its argument `algorithm` set to the built-in `working_model_Q_one` (see Section 7.5). Recall also that $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ is the algorithm for the estimation of $\bar{Q}_0$ as it is implemented in `estimate_Qbar` with its argument `algorithm` set to the built-in `kknn_algo` (see Section 7.6.2). We have already trained the latter on the $n = 1000$ first observations in `obs`. Let us train the former on the same data set:

```
Qbar_hat_kknn <- estimate_Qbar(head(obs, 1e3),
                               algorithm = kknn_algo,
```

```
                                  trControl = knn_control,
                                  tuneGrid = knn_grid)
```

```
Qbar_hat_d <- estimate_Qbar(head(obs, 1e3), working_model_Q_one)
```

With these estimators handy, computing the G-computation estimator is as simple as running the following chunk of code:

```
(compute_gcomp(QW_hat, wrapper(Qbar_hat_kknn, FALSE), 1e3))
#> # A tibble: 1 x 2
#>    psi_n   sig_n
#>    <dbl>   <dbl>
#> 1 0.0722 0.00487
(compute_gcomp(QW_hat, wrapper(Qbar_hat_d, FALSE), 1e3))
#> # A tibble: 1 x 2
#>    psi_n   sig_n
#>    <dbl>   <dbl>
#> 1 0.0742 0.00215
```

Note how we use function `wrapper` again, and that it is necessary to provide the number of observations upon which the estimation of the Q_W and $\bar{Q}$ features of P_0.

8.4.2 Elementary statistical properties

This subsection is very similar to its counterpart for the IPTW estimator, see Section 8.2.2.

Let us denote by $\bar{Q}_{n,1}$ the output of algorithm $\widehat{\mathcal{A}}_{\bar{Q},1}$ trained on P_n, and recall that $\bar{Q}_{n,\text{kNN}}$ is the output of algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ trained on P_n. Let ψ_n^d and ψ_n^e be the G-computation estimators obtained by substituting $\bar{Q}_{n,1}$ and $\bar{Q}_{n,\text{kNN}}$ for $\bar{Q}_n$ in (8.1), respectively.

If $\bar{Q}_{n,\bullet}$ minimized the empirical risk over a finite-dimensional, identifiable, and **well-specified** working model, then $\sqrt{n}(\psi_n^\bullet - \psi_0)$ would converge in law to a centered Gaussian law (here $\psi_n^\bullet$ represents the G-computation estimator obtained by substituting $\bar{Q}_{n,\bullet}$ for $\bar{Q}_n$ in (8.1)). Moreover, the asymptotic variance of $\sqrt{n}(\psi_n^\bullet - \psi_0)$ would be estimated **anti-conservatively**[4] with

$$v_n^d \equiv \text{Var}_{P_n}\left(\bar{Q}_{n,1}(1,\cdot) - \bar{Q}_{n,1}(0,\cdot)\right) \tag{8.2}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\bar{Q}_{n,1}(1,W_i) - \bar{Q}_{n,1}(0,W_i) - \psi_n^d\right)^2. \tag{8.3}$$

[4] In words, v_n^d converges to a lower-bound of the true asymptotic variance.

Unfortunately, algorithm $\widehat{\mathcal{A}}_{\bar{Q},1}$ is mis-specified and $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ is not based on a finite-dimensional working model. Nevertheless, function `compute_gcomp` estimates (in general, very poorly) the asymptotic variance with (8.3).

We investigate *empirically* the statistical behavior of ψ_n^d in Section 8.4.3. For an analysis of the reason why v_n^d is an anti-conservative estimator of the asymptotic variance of $\sqrt{n}(\psi_n^d - \psi_0)$, see here. We wish to emphasize that anti-conservativeness is even more embarrassing that conservativeness (both being contingent on the fact that the algorithms are well-specified, fact that cannot be true in the present case in real world situations).

What would happen if we used a less amenable algorithm $\widehat{\mathcal{A}}_{\bar{Q}}$. For instance, $\widehat{\mathcal{A}}_{\bar{Q}}$ could still be well-specified but so *versatile/complex* (as opposed to being based on well-behaved, finite-dimensional parametric model) that the estimator $\bar{Q}_n$, though still consistent, would converge slowly to its target. Then, root-n consistency would fail to hold. We can explore empirically this situation with estimator ψ_n^e that hinges on algorithm $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$. Or $\widehat{\mathcal{A}}_{\bar{Q}}$ could be mis-specified and there would be no guarantee at all that the resulting estimator ψ_n be even consistent.

8.4.3 Empirical investigation

Fixed sample size

Let us compute ψ_n^d and ψ_n^e on the same `iter = 100` independent samples of independent observations drawn from P_0 as in Sections 6.3 and 8.2.3. We first enrich object `learned_features_fixed_sample_size` that was created in Section 8.2.3, adding to it estimators of $\bar{Q}_0$ obtained by training algorithms $\widehat{\mathcal{A}}_{\bar{Q},1}$ and $\widehat{\mathcal{A}}_{\bar{Q},\text{kNN}}$ on each smaller data set.

The second series of commands creates object `psi_hat_de`, an 100 by six `tibble` containing notably the values of ψ_n^d and $\sqrt{v_n^d}/\sqrt{n}$ computed by calling `compute_gcomp`, and those of the recentered (with respect to ψ_0) and renormalized $\sqrt{n}/\sqrt{v_n^d}(\psi_n^d - \psi_0)$. Because we know beforehand that v_n^d under-estimates the actual asymptotic variance of $\sqrt{n}(\psi_n^d - \psi_0)$, the `tibble` also includes the values of $\sqrt{n}/\sqrt{v^{d*}}(\psi_n^d - \psi_0)$ where the estimator v^{d*} of the asymptotic variance is computed *across the replications of* ψ_n^d. The tibble includes the same quantities pertaining to ψ_n^e, although there is no theoretical guarantee that the central limit theorem does hold and, even if it did, that the counterpart v_n^e to v_n^d estimates in any way the asymptotic variance of $\sqrt{n}(\psi_n^e - \psi_0)$.

Finally, `bias_de` reports amounts of bias (at the renormalized scales — plural). There is one value of bias for each combination of *(i)* type of the estimator (`d` or `e`) and *(ii)* how the renormalization is carried out, either based on v_n^d and v_n^e (`auto_renormalization` is `TRUE`) *or* on the estimator of the asymptotic variance computed across the replications of ψ_n^d and ψ_n^e (`auto_renormalization` is `FALSE`).

```
learned_features_fixed_sample_size <-
  learned_features_fixed_sample_size %>%
  mutate(Qbar_hat_d =
           map(obs,
               ~ estimate_Qbar(., algorithm = working_model_Q_one)),
         Qbar_hat_e =
           map(obs,
               ~ estimate_Qbar(., algorithm = kknn_algo,
                               trControl = kknn_control,
                               tuneGrid = kknn_grid))) %>%
  mutate(QW = map(obs, estimate_QW),
         est_d =
           pmap(list(QW, Qbar_hat_d, n()),
                ~ compute_gcomp(..1, wrapper(..2, FALSE), ..3)),
         est_e =
           pmap(list(QW, Qbar_hat_e, n()),
                ~ compute_gcomp(..1, wrapper(..2, FALSE), ..3)))

psi_hat_de <- learned_features_fixed_sample_size %>%
  select(est_d, est_e) %>%
  gather(`est_d`, `est_e`, key = "type", value = "estimates") %>%
  extract(type, "type", "_([de])$") %>%
  unnest(estimates) %>%
  group_by(type) %>%
  mutate(sig_alt = sd(psi_n)) %>%
  mutate(clt_ = (psi_n - psi_zero) / sig_n,
         clt_alt = (psi_n - psi_zero) / sig_alt) %>%
  gather(`clt_`, `clt_alt`, key = "key", value = "clt") %>%
  extract(key, "key", "_(.*)$") %>%
  mutate(key = ifelse(key == "", TRUE, FALSE)) %>%
  rename("auto_renormalization" = key)

(bias_de <- psi_hat_de %>%
   group_by(type, auto_renormalization) %>%
   summarize(bias = mean(clt)))
#> # A tibble: 4 x 3
#> # Groups:   type [?]
#>   type  auto_renormalization  bias
#>   <chr> <lgl>                <dbl>
#> 1 d     FALSE                0.356
#> 2 d     TRUE                 1.08
#> 3 e     FALSE                0.130
```

```
#> 4 e      TRUE                  0.150
```

```
fig <- ggplot() +
  geom_line(aes(x = x, y = y),
            data = tibble(x = seq(-4, 4, length.out = 1e3),
                          y = dnorm(x)),
            linetype = 1, alpha = 0.5) +
  geom_density(aes(clt, fill = type, colour = type),
               psi_hat_de, alpha = 0.1) +
  geom_vline(aes(xintercept = bias, colour = type),
             bias_de, size = 1.5, alpha = 0.5) +
  facet_wrap(~ auto_renormalization,
             labeller =
               as_labeller(c(`TRUE` = "auto-renormalization: TRUE",
                             `FALSE` = "auto-renormalization: FALSE")),
             scales = "free")

fig +
  labs(y = "",
       x = expression(paste(sqrt(n/v[n]^{list(d, e)})*
                            (psi[n]^{list(d, e)} - psi[0]))))
```

We represent the empirical laws of the recentered (with respect to ψ_0) and renormalized ψ_n^d and ψ_n^e in Figure 8.4 (kernel density estimators). Two renormalization schemes are considered, either based on an estimator of the asymptotic variance (left) or on the empirical variance computed across the `iter` independent replications of the estimators (right). We emphasize that the x-axis ranges differ starkly between the left and right plots.

Two important comments are in order. First, on the one hand, the G-computation estimator ψ_n^d is biased. Specifically, by the above chunk of code, the averages of $\sqrt{n/v_n^d}(\psi_n^d - \psi_0)$ and $\sqrt{n/v_n^{d*}}(\psi_n^d - \psi_0)$ computed across the realizations are equal to 1.076 and 0.356 (see `bias_de`). On the other hand, the G-computation estimator ψ_n^e is biased too, though slightly less than ψ_n^d. Specifically, by the above chunk of code, the averages of $\sqrt{n/v_n^e}(\psi_n^e - \psi_0)$ and $\sqrt{n/v^{e*}}(\psi_n^e - \psi_0)$ computed across the realizations are equal to 0.15 and 0.13 (see `bias_de`). We can provide an oracular explanation. Estimator ψ_n^d suffers from the poor approximation of $\bar{Q}_0$ by $\widehat{\mathcal{A}}_{\bar{Q},1}(P_n)$, a result of the algorithm's mis-specification. As for ψ_n^e, it behaves better because $\widehat{\mathcal{A}}_{\bar{Q},\mathrm{kNN}}(P_n)$ approximates $\bar{Q}_0$ better than $\widehat{\mathcal{A}}_{\bar{Q},1}(P_n)$, an apparent consequence of the greater versatility of the algorithm.

Second, we get a visual confirmation that v_n^d under-estimates the actual asymptotic variance of $\sqrt{n}(\psi_n^d - \psi_0)$: the right-hand side red bell-shaped curve is too dispersed. In contrast, the right-hand side blue bell-shaped curve is much closer to the black curve that represents the density of the standard normal law. Looking at the left-hand side plot reveals that the

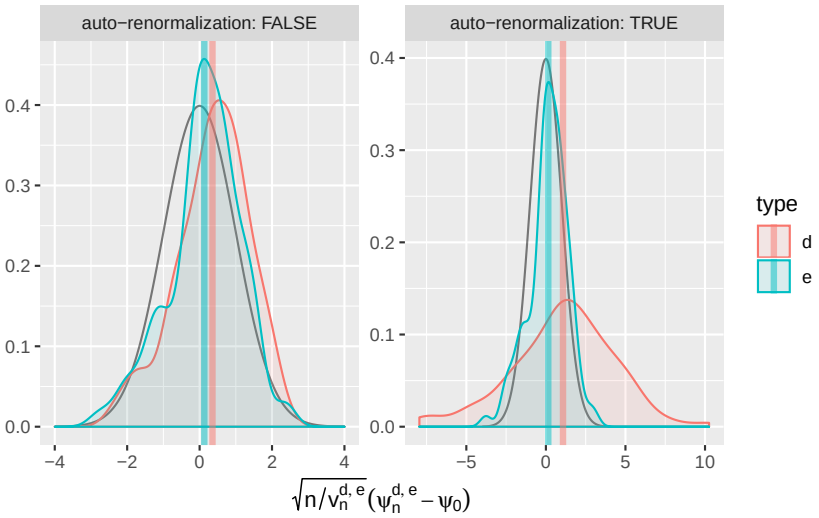


Figure 8.4: Kernel density estimators of the law of two estimators of ψ_0 (recentered with respect to ψ_0, and renormalized). The estimators respectively hinge on algorithms $\widehat{\mathcal{A}}_{\bar{Q},1}$ (d) and $\widehat{\mathcal{A}}_{\bar{Q},\mathrm{kNN}}$ (e) to estimate $\bar{Q}_0$. Two renormalization schemes are considered, either based on an estimator of the asymptotic variance (left) or on the empirical variance computed across the `iter` independent replications of the estimators (right). We emphasize that the x-axis ranges differ starkly between the left and right plots.

empirical law of $\sqrt{n/v^{d*}}(\psi_n^d - \psi_0)$, once translated to compensate for the bias, is rather close to the black curve. This means that the random variable is approximately distributed like a Gaussian random variable. On the contrary, the empirical law of $\sqrt{n/v^{e*}}(\psi_n^e - \psi_0)$ does not strike us as being as closely Gaussian-like as that of $\sqrt{n/v^{d*}}(\psi_n^d - \psi_0)$. By being more data-adaptive than $\widehat{\mathcal{A}}_{\bar{Q},1}$, algorithm $\widehat{\mathcal{A}}_{\bar{Q},\mathrm{kNN}}$ yields a better estimator of $\bar{Q}_0$. However, the rate of convergence of $\widehat{\mathcal{A}}_{\bar{Q},\mathrm{kNN}}(P_n)$ to its limit may be slower than root-n, invalidating a central limit theorem.

How do the estimated variances of ψ_n^d and ψ_n^e compare with their empirical counterparts (computed across the `iter` replications of the construction of the two estimators)?

```
## psi_n^d
(psi_hat_de %>% ungroup %>%
   filter(type == "d" & auto_renormalization) %>% pull(sig_n) %>% summary)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.00269 0.00539 0.00641 0.00648 0.00753 0.01012
## psi_n^e
(psi_hat_de %>% ungroup %>%
   filter(type == "e" & auto_renormalization) %>% pull(sig_n) %>% summary)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.0133  0.0149  0.0156  0.0157  0.0163  0.0188
```

The empirical standard deviation of ψ_n^d is approximately 2.933 times larger than the average *estimated* standard deviation. The estimator is anti-conservative indeed!

As for the empirical standard deviation of ψ_n^e, it is approximately 1.214 times larger than the average *estimated* standard deviation.

Varying sample size

```
##
## not updated yet
##

sample_size <- c(2e3, 3e3)
block_size <- sum(sample_size)


learned_features_varying_sample_size <- obs %>% as.tibble %>%
  head(n = (nrow(.) %/% block_size) * block_size) %>%
  mutate(block = label(1:nrow(.), sample_size)) %>%
  nest(-block, .key = "obs")
```

First, we cut the data set into independent sub-data sets of sample size n in { 2000, 3000 }. Second, we infer ψ_0 as shown two chunks earlier. We thus obtain 20 independent realizations of each estimator derived on data sets of 2, increasing sample sizes.

```
learned_features_varying_sample_size <-
  learned_features_varying_sample_size %>%
  mutate(Qbar_hat_d =
           map(obs,
               ~ estimate_Qbar(., algorithm = working_model_Q_one)),
         Qbar_hat_e =
           map(obs,
               ~ estimate_Qbar(., algorithm = kknn_algo,
                               trControl = kknn_control,
                               tuneGrid = kknn_grid))) %>%
  mutate(QW = map(obs, estimate_QW),
         est_d =
           pmap(list(QW, Qbar_hat_d, n()),
                ~ compute_gcomp(..1, wrapper(..2, FALSE), ..3)),
         est_e =
           pmap(list(QW, Qbar_hat_e, n()),
                ~ compute_gcomp(..1, wrapper(..2, FALSE), ..3)))

root_n_bias <- learned_features_varying_sample_size %>%
  mutate(block = unlist(map(strsplit(block, "_"), ~.x[2])),
         sample_size = sample_size[as.integer(block)]) %>%
  select(block, sample_size, est_d, est_e) %>%
  gather(`est_d`, `est_e`, key = "type", value = "estimates") %>%
  extract(type, "type", "_([de])$") %>%
  unnest(estimates) %>%
  group_by(block, type) %>%
  mutate(sig_alt = sd(psi_n)) %>%
  mutate(clt_ = (psi_n - psi_zero) / sig_n,
         clt_alt = (psi_n - psi_zero) / sig_alt) %>%
  gather(`clt_`, `clt_alt`, key = "key", value = "clt") %>%
  extract(key, "key", "_(.*)$") %>%
  mutate(key = ifelse(key == "", TRUE, FALSE)) %>%
  rename("auto_renormalization" = key)
```

The `tibble` called `root_n_bias` reports root-n times bias for all combinations of estimator and sample size. The next chunk of code presents visually our findings, see Figure 8.5. Note how we include the realizations of the estimators derived earlier and contained in `psi_hat_de` (thus breaking the independence between components of `root_n_bias`, a small price to pay in this context).

```
root_n_bias <- learned_features_fixed_sample_size %>%
   mutate(block = "0",
          sample_size = B/iter) %>%  # because *fixed* sample size
```

```
    select(block, sample_size, est_d, est_e) %>%
    gather(`est_d`, `est_e`, key = "type", value = "estimates") %>%
    extract(type, "type", "_([de])$") %>%
    unnest(estimates) %>%
    group_by(block, type) %>%
    mutate(sig_alt = sd(psi_n)) %>%
    mutate(clt_ = (psi_n - psi_zero) / sig_n,
           clt_alt = (psi_n - psi_zero) / sig_alt) %>%
    gather(`clt_`, `clt_alt`, key = "key", value = "clt") %>%
    extract(key, "key", "_(.*)$") %>%
    mutate(key = ifelse(key == "", TRUE, FALSE)) %>%
    rename("auto_renormalization" = key) %>%
    full_join(root_n_bias)

root_n_bias %>%
  ggplot() +
  stat_summary(aes(x = sample_size, y = clt,
                   group = interaction(sample_size, type),
                   color = type),
               fun.data = mean_se, fun.args = list(mult = 2),
               position = position_dodge(width = 250), cex = 1) +
  stat_summary(aes(x = sample_size, y = clt,
                   group = interaction(sample_size, type),
                   color = type),
               fun.data = mean_se, fun.args = list(mult = 2),
               position = position_dodge(width = 250), cex = 1,
               geom = "errorbar", width = 750) +
  stat_summary(aes(x = sample_size, y = clt,
                   color = type),
               fun.y = mean,
               position = position_dodge(width = 250),
               geom = "polygon", fill = NA) +
  geom_point(aes(x = sample_size, y = clt,
                 group = interaction(sample_size, type),
                 color = type),
             position = position_dodge(width = 250),
             alpha = 0.1) +
  scale_x_continuous(breaks = unique(c(B / iter, sample_size))) +
  labs(x = "sample size n",
       y = expression(paste(sqrt(n) * (psi[n]^{list(d, e)} - psi[0])))) +
  facet_wrap(~ auto_renormalization,
             labeller =
```

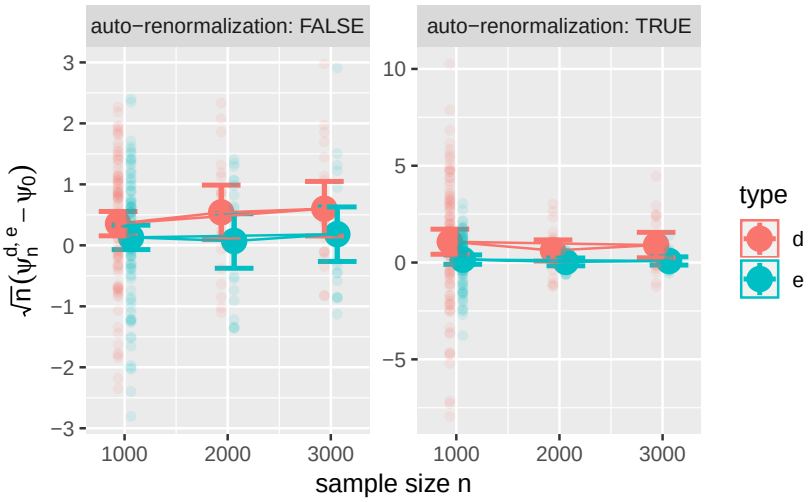


Figure 8.5: Evolution of root-n times bias versus sample size for two inference methodology of ψ_0 based on the estimation of $\bar{Q}_0$. Big dots represent the average biases and vertical lines represent twice the standard error.

```
              as_labeller(c(`TRUE` = "auto-renormalization: TRUE",
                            `FALSE` = "auto-renormalization: FALSE")),
            scales = "free")
```

Section 9

Work in progress

(...)

Section 10

Notation

- $O = (W, A, Y)$, the generic summary of how one realization of the experiments of interest unfold, our generic observation; $W \in [0,1]$ is the context of action, $A \in \{0,1\}$ is the action undertaken, and $Y \in [0,1]$ is the reward of action A in context W. We denote by $\mathbb{O} \equiv [0,1] \times \{0,1\} \times [0,1]$ the set where a generic O takes its values.
- P, P_0, Π_0, Π_h, Π'_0, Π'_h, laws (on $\mathbb{O}$) for O.
- $Pf \equiv \mathrm{E}_P(f(O))$ for any law P for O and function f from $\mathbb{O}$ to $\mathbb{R}^p$.
- $\|f\|_P^2 \equiv Pf^2 = \mathrm{E}_P(f(O)^2) = \int f(o)^2 dP(o)$, the square of the $L^2(P)$-norm of f, a function from $\mathbb{O}$ to $\mathbb{R}$.
- P_n, the empirical measure. If the observations are O_1, ..., O_n, then P_n is a law such that the generic random variable O drawn from P_n takes its values in $\{O_1, \ldots, O_n\}$ in such a way that $O = O_i$ with probability n^{-1} for each $1 \leq i \leq n$.
- $\sqrt{n}(P_n - P)$, where P_n is the empirical measure associated to $O_1, \ldots, O_n$ drawn independently from P, the empirical process.
- $\mathcal{M}$, the model, that is, the collection of *all* laws from which O can be drawn and that meet some constraints.
- $\mathcal{M}^{\text{empirical}}$, the collection of all discrete laws on $[0,1] \times \{0,1\} \times [0,1]$, of which P_n is a distinguished element.
- Q_W, $Q_{0,W}$, marginal laws for W (under P and P_0, respectively).
- $\bar{G}(W) \equiv \Pr_P(A = 1|W)$, $\bar{G}_0(W) \equiv \Pr_{P_0}(A = 1|W)$, conditional probabilities of action $A = 1$ given W (under P and P_0, respectively). For each $a \in \{0,1\}$, $\ell\bar{G}(a, W) \equiv \Pr_P(A = a|W)$ and $\ell\bar{G}_0(a, W) \equiv \Pr_{P_0}(A = a|W)$.
- $\bar{Q}(A, W) = \mathrm{E}_P(Y|A, W)$, $\bar{Q}_0(A, W) = \mathrm{E}_{P_0}(Y|A, W)$, the conditional means of Y given A and W (under P and P_0, respectively).
- q_Y, $q_{0,Y}$, conditional densities of Y given A and W (under P and P_0, respectively).

- $\Psi : \mathcal{M} \to [0,1]$, given by $\Psi(P) \equiv \int \left(\bar{Q}(1,w) - \bar{Q}(0,w)\right) dQ_W(w)$, the statistical mapping of interest.
- $\psi \equiv \Psi(P)$, $\psi_0 \equiv \Psi(P_0)$.
- $\widehat{\mathcal{A}}$, $\widehat{\mathcal{A}}_{\bar{G},1}$, $\widehat{\mathcal{A}}_{\bar{Q},1}$, algorithms to be trained on P_n, *i.e.*, mappings from $\mathcal{M}^{\text{empirical}}$ to the set where lives the feature targeted by the algorithm.

Section 11

Basic results and their proofs

11.1 NPSEM

The experiment can also be summarized by a *nonparametric system of structural equations*: for some deterministic functions f_w, f_a, f_y and independent sources of randomness U_w, U_a, U_y,

1. sample the context where the counterfactual rewards will be generated, the action will be undertaken and the actual reward will be obtained, $W = f_w(U_w)$;
2. sample the two counterfactual rewards of the two actions that can be undertaken, $Y_0 = f_y(0, W, U_y)$ and $Y_1 = f_y(1, W, U_y)$;
3. sample which action is carried out in the given context, $A = f_a(W, U_a)$;
4. define the corresponding reward, $Y = AY_1 + (1 - A)Y_0$;
5. summarize the course of the experiment with the observation $O = (W, A, Y)$, thus concealing Y_0 and Y_1.

11.2 Identification

Let $\mathbb{P}_0$ be an experiment that generates $\mathbb{O} \equiv (W, Y_0, Y_1, A, Y)$. We think of W as the context where an action is undertaken, of Y_0 and Y_1 as the counterfactual (potential) rewards that actions $a = 0$ and $a = 1$ would entail, of A as the action carried out, and of Y as the reward received in response to action A. Consider the following assumptions:

1. **Randomization**: under $\mathbb{P}_0$, the counterfactual rewards (Y_0, Y_1) and action A are conditionally independent given W, *i.e.*, $(Y_0, Y_1) \perp A \mid W$.
2. **Consistency**: under $\mathbb{P}_0$, if action A is undertaken then reward Y_A is received, *i.e.*, $Y = Y_A$ (or $Y = Y_a$ given that $A = a$).

3. **Positivity**: under $\mathbb{P}_0$, both actions $a = 0$ and $a = 1$ have ($\mathbb{P}_0$-almost surely) a positive probability to be undertaken given W, *i.e.*, $\Pr_{\mathbb{P}_0}(\ell\bar{G}_0(a, W) > 0) = 1$ for $a = 0, 1$.

Proposition 11.1 (Identification)**.** Under the above assumptions, it holds that

$$\psi_0 = \mathrm{E}_{\mathbb{P}_0}(Y_1 - Y_0) = \mathrm{E}_{\mathbb{P}_0}(Y_1) - \mathrm{E}_{\mathbb{P}_0}(Y_0).$$

Proof. Set arbitrarily $a \in \{0, 1\}$. By the randomization assumption on the one hand (second equality) and by the consistency and positivity assumptions on the other hand (third equality), it holds that

$$\begin{aligned}\mathrm{E}_{\mathbb{P}_0}(Y_a) &= \int \mathrm{E}_{\mathbb{P}_0}(Y_a \mid W = w) dQ_{0,W}(w) = \int \mathrm{E}_{\mathbb{P}_0}(Y_a \mid A = a, W = w) dQ_{0,W}(w) \\ &= \int \mathrm{E}_{P_0}(Y \mid A = a, W = w) dQ_{0,W}(w) = \int \bar{Q}_0(a, W) dQ_{0,W}(w).\end{aligned}$$

The stated result easily follows. □

Remark. The positivity assumption is needed for $\mathrm{E}_{P_0}(Y \mid A = a, W) \equiv \bar{Q}_0(a, W)$ to be well-defined.

11.3 Building a confidence interval

Let Φ be the standard normal distribution function. Let X_1, ..., X_n be independently drawn from a given law.

11.3.1 CLT & Slutsky's lemma

Assume that $\sigma^2 \equiv \mathrm{Var}(X_1)$ is finite. Let $m \equiv \mathrm{E}(X_1)$ be the mean of X_1 and $\bar{X}_n \equiv n^{-1}\sum_{i=1}^n X_i$ be the empirical mean. By the central limit theorem (CLT), it holds that $\sqrt{n}(\bar{X}_n - m)$ converges in law as n grows to the centered Gaussian law with variance σ^2.

Moreover, if σ_n^2 is a (positive) consistent estimator of σ^2 then, by Slutsky's lemma, $\sqrt{n}/\sigma_n(\bar{X}_n - m)$ converges in law to the standard normal law. The empirical variance $n^{-1}\sum_{i=1}^n (X_i - \bar{X}_n)^2$ is such an estimator.

Proposition 11.2. Under the above assumptions,

$$\left[\bar{X}_n \pm \Phi^{-1}(1-\alpha)\frac{\sigma_n}{\sqrt{n}}\right]$$

is a confidence interval for m with asymptotic level $(1 - 2\alpha)$.

11.3.2 CLT and order statistics

Suppose that the law of X_1 admits a continuous distribution function F. Set $p \in]0,1[$ and, assuming that n is large, find $k \geq 1$ and $l \geq 1$ such that

$$\frac{k}{n} \approx p - \Phi^{-1}(1-\alpha)\sqrt{\frac{p(1-p)}{n}}$$

and

$$\frac{l}{n} \approx p + \Phi^{-1}(1-\alpha)\sqrt{\frac{p(1-p)}{n}}.$$

Proposition 11.3. Under the above assumptions, $[X_{(k)}, X_{(l)}]$ is a confidence interval for $F^{-1}(p)$ with asymptotic level $1-2\alpha$.

11.4 Another representation of the parameter of interest

For notational simplicitiy, note that $(2a-1)$ equals 1 if $a=1$ and -1 if $a=0$. Now, for each $a = 0, 1$,

$$\begin{aligned}
\mathrm{E}_{P_0}\left(\frac{\mathbf{1}\{A=a\}Y}{\ell\bar{G}_0(a,W)}\right) &= \mathrm{E}_{P_0}\left(\mathrm{E}_{P_0}\left(\frac{\mathbf{1}\{A=a\}Y}{\ell\bar{G}_0(a,W)}\middle| A, W\right)\right) \\
&= \mathrm{E}_{P_0}\left(\frac{\mathbf{1}\{A=a\}}{\ell\bar{G}_0(a,W)}\bar{Q}_0(A,W)\right) \\
&= \mathrm{E}_{P_0}\left(\frac{\mathbf{1}\{A=a\}}{\ell\bar{G}_0(a,W)}\bar{Q}_0(a,W)\right) \\
&= \mathrm{E}_{P_0}\left(\mathrm{E}_{P_0}\left(\frac{\mathbf{1}\{A=a\}}{\ell\bar{G}_0(a,W)}\bar{Q}_0(a,W)\middle| W\right)\right) \\
&= \mathrm{E}_{P_0}\left(\frac{\ell\bar{G}_0(a,W)}{\ell\bar{G}_0(a,W)}\bar{Q}_0(a,W)\middle| W\right) \\
&= \mathrm{E}_{P_0}\left(\bar{Q}_0(a,W)\right),
\end{aligned}$$

where the first, fourth and sixth equalities follow from the tower rule[1], and the second and fifth hold by definition of the conditional expectation. This completes the proof.

[1] For any random variable (U, V) such that $\mathrm{E}(U|V)$ and $\mathrm{E}(U)$ are well defined, it holds that $\mathrm{E}(\mathrm{E}(U|V)) = \mathrm{E}(U)$.

11.5 The delta-method

Let f be a map from $\Theta \subset \mathbb{R}^p$ to $\mathbb{R}^q$ that is differentiable at $\theta \in \Theta$. Let X_n be a random vector taking its values in Θ.

Proposition 11.4. If $\sqrt{n}(X_n - \theta)$ converges in law to the Gaussian law with mean μ and covariance matrix Σ, then $\sqrt{n}(f(X_n) - f(\theta))$ converge in law to the Gaussian law with mean $\nabla f(\theta) \times \mu$ and covariance matrix $\nabla f(\theta) \times \Sigma \times \nabla f(\theta)^\top$. In addition, if Σ_n estimates Σ consistently then, by Slutsky's lemma, the asymptotic variance of $\sqrt{n}(f(X_n) - f(\theta))$ is consistently estimated with $\nabla f(X_n) \times \Sigma_n \times \nabla f(X_n)^\top$.

11.6 Asymptotic negligibility of the remainder term

Recall that $\|f\|_P^2 \equiv \mathrm{E}_P\left(f(O)^2\right)$ is the $L_2(P)$-norm of f, a measurable function from $\mathcal{O}$ to $\mathbb{R}$. Assume that for $a = 0, 1$, $\ell \bar{G}_n(a, W) \geq \delta > 0$ $Q_{0,W}$-almost everywhere.

The Cauchy-Schwarz inequality then implies that, for $a = 0, 1$,

$$\mathrm{Rem}_{P_0}(\hat{P}_n) \leq \frac{2}{\delta} \max_{a=0,1} \left(\|\bar{Q}_n(a, \cdot) - \bar{Q}_0(a, \cdot)\|_{P_0} \right) \times \|\bar{G}_n - \bar{G}_0\|_{P_0}.$$

Therefore, if for $a = 0, 1$,

$$\|\bar{Q}_n(a, \cdot) - \bar{Q}_0(a, \cdot)\|_{P_0} = o_{P_0}(n^{-1/4})$$

and

$$\|\bar{G}_n - \bar{G}_0\|_{P_0} = o_{P_0}(n^{-1/4}),$$

then

$$\mathrm{Rem}_{P_0}(\hat{P}_n) = o_{P_0}(n^{-1/2}).$$

Section 12

More results and their proofs

Written too quickly. Checks needed!

12.1 Estimation of the asymptotic variance of an estimator

12.1.1 IPTW estimator based on a well-specified model

Sketch (to extend later on)

The IPTW estimator ψ_n^b relies on algorithm $\widehat{\mathcal{A}}_{\bar{G},1}$, which is "well-specified" in the sense that its output $\bar{G}_n \equiv \widehat{\mathcal{A}}_{\bar{G},1}(P_n)$ minimizes the empirical risk over a finite-dimensional, identifiable, well-specified working model for $\bar{G}_0$. If one introduces D given by

$$D(O) \equiv \frac{(2A-1)}{\ell\bar{G}_0(A,W)}Y,$$

then the influence curve of ψ_n^b equals $D-\Psi(P_0)$ minus the projection of D onto the tangent space of the above parametric model for $\bar{G}_0$. The variance of the influence curve is thus smaller than that of D, hence the conservativeness.

12.1.2 G-computation estimator based on a well-specified model

Sketch (to extend later on, see [van der Laan and Rose, 2011] page 527)

Consider a G-computation estimator ψ_n that relies on an algorithm $\widehat{\mathcal{A}}_{\bar{Q}}$ that is "well-specified" in the sense that its output $\bar{Q}_n \equiv \widehat{\mathcal{A}}_{\bar{Q}}(P_n)$ minimizes the empirical risk over a finite-dimensional, identifiable, well-specified working model for $\bar{Q}_0$. If one introduces D given by

$$D(O) \equiv \bar{Q}_0(1, W) - \bar{Q}_0(0, W)$$

then the influence curve of ψ_n equals $D - \Psi(P_0)$ plus a function of O that is orthogonal to $D - \Psi(P_0)$. Thus the variance of the influence curve is larger than that of D, hence the anti-conservativeness.

Bibliography

Mark J. van der Laan and Sherri Rose. *Targeted learning*. Springer Series in Statistics. Springer, New York, 2011. ISBN 978-1-4419-9781-4. URL https://doi.org/10.1007/978-1-4419-9782-1. Causal inference for observational and experimental data.

Mark J. van der Laan and Sherri Rose. *Targeted learning in data science*. Springer Series in Statistics. Springer, Cham, 2018. ISBN 978-3-319-65303-7; 978-3-319-65304-4. URL https://doi.org/10.1007/978-3-319-65304-4. Causal inference for complex longitudinal studies.

Hadley Wickham and Garrett Grolemund. *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc., 2016.

Index