Math Camp Notes

Walter W. Zhang*

Chicago Booth PhD Math Camp

July 19, 2023

Objective

This set of notes covers the topics of Statistical Inference I, Statistical Inference III, Optimization Theory, and Dynamic Programming from the Booth PhD Math Camp taught during Summer 2020, 2021, and 2022. The provided material aims to assist students in preparing for their first-year courses and are accompanied by computational examples.

*Questions or comments: walterwzhang@chicagobooth.edu

Contents

How to read these notes

This set of notes covers half of the Chicago Booth PhD Math Camp given during Summer 2020, 2021, and 2022. Karthik's notes cover the other half of the course, which includes Linear Algebra, Real Analysis, Probability, and Statistical Inference II.[1]

Combined, the notes intend to provide a comprehensive introduction to the material that PhD students may see over their first year courses. Students are *not* expected to know all the material before starting their classes, but they should hopefully be able to grasp concepts and trace themes out in the material that will be repeated throughout their first-year courses.

The notes are structured as follows. Statistical Inference I provides an introduction to the fundamentals of statistical inference. Statistical Inference III provides an overview of panel data methods and introduces causal inference. Optimization Theory introduces constrained, unconstrained, and convex optimization. Dynamic Programming first covers fixed point theorems, then introduces dynamic programming, and finally provides the general setup.

The fundamentals sections under each part provide the essential materials and key themes that students should see in their courses. At a minimum, these should be reviewed in preparation for the first-year courses. The topics sections provide additional content for student interested in exploring more. The computational examples give a sense of how the tools and machinery taught can be implemented in practice.[2]

The split for fundamentals vs. topics is not made explicitly for Statistical Inference III and Dynamic Programming. For Statistical Inference III, Section 5.2 should be treated as the topics section, and for Dynamic Programming, Section 11.4 should be treated as the topics section. The machine learning introduction for Statistical Inference III has been incorporated into the its computational example.

Various sections of these notes are based off material from Jianfei Cao's Math Camp notes, Chris Hansen's applied econometrics lecture notes, John Cochrane's time series analysis notes, Günter J. Hitsch's notes on causality, Galichon (2016) for overviews of the optimal transport and fixed point theorems, and Adda and Cooper (2003) for the introduction to dynamic programming. The comic strips are from PhD Comics.

I thank Malaina Brown, Jefferey R. Russell, Kim Mayer, Jianfei Cao, Ali Goli, James W. Kiselik, my PhD cohort, and my co-instructor Karthik Srinivasan for all their help in designing and running the course. I thank all the students who took my course over the years for refining the course materials.

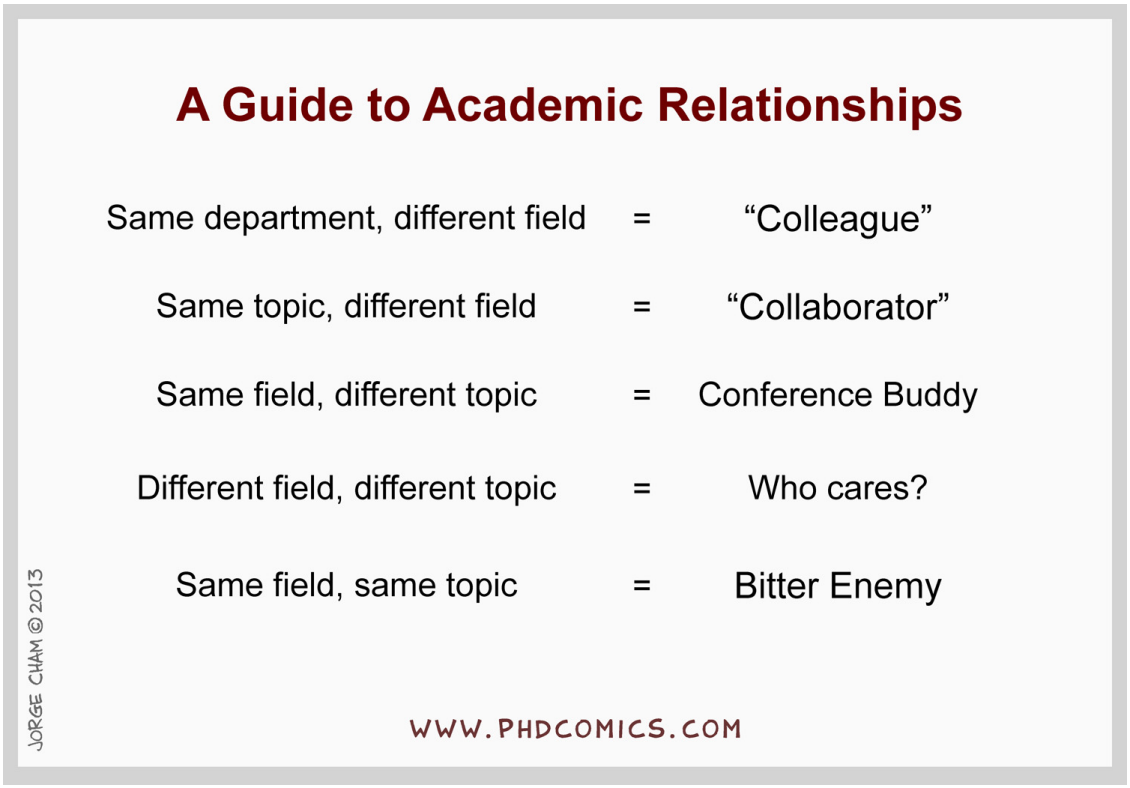
[1] Karthik's notes can be found on his website.

[2] The coding examples are available on RStudio Cloud: `https://posit.cloud/content/2778671`.

Part I

Statistical Inference I - Fundamentals

This section provides an overview of the fundamental concepts in statistical inference used over the first year courses. These concepts should provide background for the first quarter econometrics courses at Booth or the Economics department. The companion RMarkdown notebook walks through a derivation of the properties of an estimator and provides a simulation study comparing biased and unbiased estimators.



1 Fundamentals

1.1 Building Blocks

Definition 1. Let $\{X_n\}_{n=1}^{\infty}$ and X be random vectors on $\mathbb{R}^k$.

(i) (Convergence in distribution) $X_n \xrightarrow{d} X$, if $\Pr(X_n \leq x) \to \Pr(X \leq x)$ for all continuous points of $x \mapsto \Pr(X \leq x)$

(ii) (Convergence in probability) $X_n \xrightarrow{p} X$, if $\Pr(|X_n - X| \geq \epsilon) \to 0$ for all $\epsilon > 0$

(iii) (Almost sure convergence) $X_n \xrightarrow{as} X$, if $\Pr(\lim_{n\to\infty} X_n = X) = 1$

Remark.

1. If $X_n \xrightarrow{as} X$, then $X_n \xrightarrow{p} X$; If $X_n \xrightarrow{p} X$, then $X_n \xrightarrow{d} X$.
2. If $X_n \xrightarrow{d} X$ and $Y_n \xrightarrow{d} Y$, it is not necessary that $(X_n, Y_n)' \xrightarrow{d} (X, Y)'$.
   Counterexample: Let $X \sim N(0,1)$, $X_n = X$ and $Y_n = -X$ for each n. Then, $X_n \xrightarrow{d} X$ and $Y_n \xrightarrow{d} X$, but $(X_n, Y_n)' \xrightarrow{d} (X, X)'$ does not hold. This is because

$$\begin{bmatrix} X_n \\ Y_n \end{bmatrix} = \begin{bmatrix} X \\ -X \end{bmatrix} \xrightarrow{d} N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\right),$$

   but

$$\begin{bmatrix} X \\ X \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right).$$

Lemma 1. *(Markov's Inequality) Let X be a random variable. Then,*

$$P(|X| \geq a) \leq \frac{E[|X|]}{a}$$

for $a > 0$. More generally,

$$P(|X| \geq a) \leq \frac{E[f(|X|)]}{f(a)}$$

for $a > 0$ and a non-negative increasing function f.

Proof. $P(|X| \geq a) = E[\mathbb{1}\{|X| \geq a\}] \leq E[\frac{|X|}{a}\mathbb{1}\{|X| \geq a\}] = E[|X|/a] = E[|X|]/a.$ □

Corollary 1. *(Chebyshev's Inequality)*

$$P(|X - \mu| \geq k\sigma) \leq 1/k^2,$$

where $\mu = E[X]$ and $\sigma^2 = Var[X]$.

Note that this implies for any random variable, the probability of being 2 standard deviations away is less than $1/4$.

Lemma 2. *(Continuity of probability measure) Let $P(A) = \Pr(X \in A)$ be the induced probability measure. Then,*

$$A_1 \subset A_2 \subset \cdots \Longrightarrow P(\cup_{n\geq 1} A_n) = \lim_{n\to\infty} P(A_n)$$

and

$$B_1 \supset B_2 \supset \cdots \Longrightarrow P(\cap_{n\geq 1} B_n) = \lim_{n\to\infty} P(B_n).$$

Example 1. $X \sim N(0, I_2)$, $A_n = \{(X_1, X_2) : |X_1| < n, |X_2| < n\}$.

Remark.

1. Even if both X and Y are normally distributed, it does not follow that (X, Y) is jointly normal. Counterexample: $X_1 \sim N(0, 1)$ and

$$X_2 = \begin{cases} X_1 & \text{if } U \leq 1/2 \\ -X_1 & \text{if } U > 1/2 \end{cases}$$

   where U is uniformly distributed on $[0, 1]$ and independent of X_1. Then $X_2 \sim N(0, 1)$. But $X_2|X_1$ is not normally distributed, violating requirements of joint normality.
2. If (X, Y) is jointly normal and $Cov[X, Y] = 0$, then X is independent of Y.

1.1.1 Useful Theorems

Theorem 1. *(Continuous Mapping Theorem/CMT) Let* $\{X_n\}_{n=1}^{\infty}$ *and* X *be random vectors on* $\mathbb{R}^k$*, and* $g : \mathbb{R}^k \to \mathbb{R}^d$ *is continuous function on a set* $C \subset \mathbb{R}^k$ *where* $\Pr\{X \in C\} = 1$.

(i) If $X_n \xrightarrow{d} X$*, then* $g(X_n) \xrightarrow{d} g(X)$
(ii) If $X_n \xrightarrow{p} X$*, then* $g(X_n) \xrightarrow{p} g(X)$
(iii) If $X_n \xrightarrow{as} X$*, then* $g(X_n) \xrightarrow{as} g(X)$

CMT is useful when the consistency or asymptotic distribution of $g(X_n)$ are hard to obtain but those of X_n are easier to acquire.

Proof. (ii) Fix $\epsilon > 0$. For each $\delta > 0$, let $B_\delta = \{x \in \mathbb{R}^k : \exists y \in \mathbb{R}^k,\ s.t.\ d(x, y) < \delta \text{ and } d(g(x), g(y)) > \epsilon\}$. Then, for some $x \in \mathbb{R}^k$, if $d(g(x), g(y)) > \epsilon$ and $x \notin B_\delta$, then $d(x, y) \geq \delta$. Thus,

$$\Pr(d(g(X_n), g(X)) > \epsilon) \leq \Pr(X \in B_\delta) + \Pr(d(X_n, X) \geq \delta).$$

Let $A_n = B_{1/n} \cap C$. Pick any $x \in C$. By continuity of g, there exits $\eta > 0$, such that $d(x, y) < \eta$ implies $d(g(x), g(y)) < \epsilon$. That is, for each $n > 1/\eta$, $x \notin B_{1/n}$. So $(\cap_{n\geq 1} A_n) \cap C = \emptyset$. By continuity of probability measure,

$$\lim_{n\to\infty} \Pr(X \in B_{1/n}) = \lim_{n\to\infty} \Pr(X \in A_n) = \Pr(X \in \cap_{n\geq 1} A_n) \leq 1 - \Pr(X \in C) = 0.$$

This shows $\Pr(X \in B_\delta) \to 0$ as $\delta \to 0$. Then,

$$\limsup_{n\to\infty} \Pr(d(g(X_n), g(X)) > \epsilon) \leq \Pr(X \in B_\delta) + 0.$$

Letting δ go to zero on both sides shows the theorem.

(iii) Let Ω be the sample space. Assume for each $\omega \in \Omega_1 \subset \Omega$, we have $X_n(\omega) \to X(\omega)$. Let $\Omega_2 = \{\omega \in \Omega : X(\omega) \in C\}$. Then, for each $\omega \in \Omega_0 = \Omega_1 \cap \Omega_2$, $X_n(\omega) \to X(\omega)$ implies $g(X_n(\omega)) \to g(X(\omega))$ by CMT of non-random sequence. Also, $\Pr(\Omega_0) = \Pr((\Omega_1^c \cup \Omega_2^c)^c) = 1 - \Pr(\Omega_1^c \cup \Omega_2^c) \geq 1 - \Pr(\Omega_1^c) - \Pr(\Omega_2^c) = 1$. □

Theorem 2. *(Weak Law of Large Number/WLLN) If* $\{X_i\}_{i=1}^{\infty}$ *is an i.i.d. sequence of random vectors such that* $E[|X_i|] < \infty$*, then* $\bar{X}_n \xrightarrow{p} E[X_i]$ *as* $n \to \infty$*, where* $\bar{X}_n = n^{-1}\sum_{i=1}^{n} X_i$*.*

WLLN is often used to show consistency.

Proof. If $Var[X_i] < \infty$, WLLN holds by Chebyshev's Inequality. □

Theorem 3. *(Strong Law of Large Number/SLLN) If* $\{X_i\}_{i=1}^{\infty}$ *is an i.i.d. sequence of random vectors such that* $E[|X_i|] < \infty$*, then* $\bar{X}_n \xrightarrow{as} E[X_i]$ *as* $n \to \infty$*, where* $\bar{X}_n = n^{-1}\sum_{i=1}^{n} X_i$*.*

Theorem 4. *(Central Limit Theorem/CLT) If* $\{X_i\}_{i=1}^{\infty}$ *is an i.i.d. sequence of random vectors such that* $Var[X_i] < \infty$*, then* $\sqrt{n}(\bar{X}_n - E[X_i]) \xrightarrow{d} N(0, Var[X_i])$*, as* $n \to \infty$*.*

CLT is often used to show asymptotic normality (often combined with CMT part (i)).

Proposition 1. *(Law of Iterated Expectation/LIE) Suppose* X *and* Y *are random variables and* $E[X]$ *exists, then* $E[X] = E[E[X|Y]]$*. More generally,* $E[X|Z] = E[E[X|Y,Z]|Z]$*.*

LIE is useful when dealing with mean independence (e.g. $E[u|x] = 0$).

Exercise 1. Let $f(y) = E[X|Y = y]$. Verify $E[X] = E[f(Y)]$.

	y_1	y_2
x_1	1/9	2/9
x_2	1/3	1/3

1.2 Statistical Inference

The three goals of statistical inference are *estimation*, *hypothesis testing*, and constructing a *confidence region.*

1.2.1 Estimation

Assume the data $\{(W_i, Z_i)\}_{i=1}^{n}$ are generated by some probability measure $P_{\theta,\eta}$. Suppose the parameter of interest is θ and we observe W_i but not Z_i. Then an **estimator** of θ is a function $\hat{\theta}_n = \hat{\theta}_n(\{W_i\}_{i=1}^{n})$.

Example 2. $Y_i = X_i + \epsilon_i$, where $X_i \sim N(\mu, \sigma^2)$ and $\epsilon_i \sim N(0, v^2)$. We observe Y_i and only care about acquiring an estimate of μ.

Definition 2. $\hat{\theta}_n$ is an **unbiased** estimator of θ if $E[\hat{\theta}_n] = \theta$. It is **consistent** or **asymptotically unbiased** if $\hat{\theta}_n \xrightarrow{p} \theta$. It is **asymptotically normal** if $g(n)(\hat{\theta}_n - \theta) \xrightarrow{d} N(0, \Sigma)$, where $g(n) \to \infty$ as $n \to \infty$. We call $g(n)$ the **rate of convergence**.

Example 3. Suppose we have an i.i.d. sample of observations $\{X_i\}_{i=1}^{n}$ and we know $X_i \sim N(\mu, \sigma^2)$. A natural estimator for μ is the sample analog $\bar{X}_n = n^{-1}\sum_{i=1}^{n} X_i$. Note that $\bar{X}_n \sim N(\mu, \frac{\sigma^2}{n})$ and is unbiased, consistent, and asymptotically normal. The rate of convergence is $\sqrt{n}$. Note that the rate of convergence is unique.

1.2.2 Hypothesis Testing

Suppose we want to perform hypothesis testing where the null hypothesis is H_0, and the significance level is α. A **test** is a function $\phi_n = \phi_n(\{W_i\}_{i=1}^n)$ that takes values between 0 and 1. Usually, $\phi_n = 1$ means rejecting the null hypothesis and $\phi_n = 0$ means failing to reject the null hypothesis.

Definition 3. The **size** of a test is $E_P[\phi_n]$ for some probability measure P that satisfies H_0. The test is **consistent in level** if $\limsup_{n\to\infty} E_P[\phi_n] \le \alpha$. The **power** of a test is $E_{P'}[\phi_n]$ for some P' that does not satisfies H_0.

Example 4. $\phi_n = \mathbb{1}\{|t| > T\}$ for some critical value T.

Type-I error refers to rejecting when the null hypothesis is true. The probability of Type-I error is the size. **Type-II error** refers to not rejecting when the null hypothesis is false. The probability of Type-II error is $(1 - power)$, which depends on the true probability measure.

Example 5. (A trivial test that is consistent in level) Let $\phi_n = \alpha$. Then it is consistent in level. In fact, it has correct size for each n.

Example 6. In the previous example, consider the null hypothesis $H_0 : \mu = \mu_0$. Let q_1 and q_2 be the $\alpha/2$-quantile and $(1-\alpha/2)$-quantile of $N(\mu_0, \frac{\sigma^2}{n})$, respectively. Then,

$$\Pr(q_1 \le \bar{X}_n \le q_2) = 1 - \alpha$$

or under the null hypothesis, there is large probability that $\bar{X}_n \in [q_1, q_2]$ when α is small. We can then construct a test $\phi_n = \phi_n(X_1, X_2, \ldots, X_n) \in [0, 1]$ such that $\phi_n = \mathbb{1}\{\bar{X}_n \notin [q_1, q_2]\}$. In this example, the size is just α. Under some alternative $\mu = \mu_1$ where $\mu_1 \neq \mu_0$, the Type-II error is $F_{\mu_1}(q_2) - F_{\mu_1}(q_1)$, where F_{μ_1} is the c.d.f. of $\bar{X}_n$ under $\mu = \mu_1$.

1.2.3 Confidence Regions

A **confidence region** is a random set $C_n = C_n(\{W_i\}_{i=1}^n)$ such that $\liminf_{n\to\infty} \Pr(\theta \in C_n) \ge 1 - \alpha$, where θ is the parameter of interest and α is significant level. Note that the probability is taken over C_n.

Example 7. In our previous example, let $C_n = [p_1, p_2]$, where p_1 and p_2 are the $\alpha/2$-quantile and $(1-\alpha/2)$-quantile of $N(\bar{X}_n, \frac{\sigma^2}{n})$, respectively. Notice that $\Pr(p_1 \le \mu \le p_2) = \Pr(p_1 - \bar{X}_n \le \mu - \bar{X}_n \le p_2 - \bar{X}_n)$, implying $\Pr(\mu \in C_n) = 1 - \alpha$.

Remark.

1. In the above example, the exact distribution of the estimator can be calculated. In more general cases where distribution of the estimator cannot be obtained, we usually use CLT to derive the asymptotic distribution and perform hypothesis testing (or constructing confidence region). That is why we need $\liminf_{n\to\infty} \Pr(\theta \in C_n) \ge 1 - \alpha$ instead of just $\Pr(\theta \in C_n) \ge 1 - \alpha$.
2. Hypothesis testing and constructing the confidence region are equivalent in the sense that the confidence region can be obtained by *inverting the test*. Let $\phi_n(t)$ denote the test of $H_0 : \theta = t$ and can only take on 1 or 0 (either rejecting or not rejecting with probability one), then we have

$\limsup_{n\to\infty} E_P[\phi_n(\theta)] \leq \alpha$. Construct the confidence region by inverting the test such that $C_n = \{t \in \Theta : \phi_n(t) = 0\}$. Then

$$\liminf_{n\to\infty} \Pr(\theta \in C_n) = \liminf_{n\to\infty} \Pr(\phi_n(\theta) = 0) = 1 - \liminf_{n\to\infty} E[\phi_n(\theta)] \geq 1 - \alpha.$$

1.3 Linear Models

1.3.1 Interpretations

Suppose we have a data sample of $\{(X_i, Y_i)\}_{i=1}^n$ and assume a linear model

$$Y = X'\beta + U,$$

where Y and U are scalars, and X and β are k-dimensional vectors. There are three interpretations of this linear regression equation.

Interpretation 1. (Linear Conditional Expectation) We assume the conditional expectation of Y is linear in X, i.e. $E[Y|X] = X'\beta$. Then, we must have mean independence or $E[U|X] = 0$.

Interpretation 2. (Best Linear Predictor) Here $E[Y|X]$ is not necessarily linear in X. Let

$$\beta = \arg\min_{b\in\mathbb{R}^k} E[(Y - X'b)^2],$$

then $X'\beta$ is the best predictor of Y among all functions that is linear in X. Note that we only have $E[XU] = 0$, which is *weaker* than mean independence.

Interpretation 3. (Linear Causal Model) X is the observed determinant of Y and U is the unobserved determinant. The relationship between X and U is not determined by the model.

1.3.2 OLS

The standard procedure of estimating a statistical model is

(i) propose an estimator
(ii) show consistency
(iii) derive its asymptotic distribution.

Suppose we have i.i.d. data $\{(X_i, Y_i)\}_{i=1}^n$, where for each i, Y_i is a scalar and X_i is a k-dimensional vector. Consider the linear model

$$Y = X'\beta + U,$$

where Y and U are scalars, and X and β are k-dimensional column vectors. It can have any of the above interpretations, which depends on your research question. We consider the set of assumptions:

Assumption 1. ***[A1]***: $E[XU] = 0$.
[A1']: $E[U|X] = 0$.
[A1"]: $E[U] = 0$ *and* $X \perp U$.

Assumption 2. ***[A2]***: $E[XX'] < \infty$ *and is non-singular.*

Assumption 3. *[A3]*: $E[XX'U^2] < \infty$.
[A3']: $E[U^2|X] = \sigma^2 < \infty$.

Remark. Note that **[A1"]**⊂**[A1']**⊂**[A1]**. Under **[A2]**, **[A3']**⊂**[A3]**. The error is said to be **homoskedastic** if $Var[U|X]$ does not vary with X. Alternatively, if it does, then it is **heteroskedastic**. In terms of our assumptions, we need **[A1']** and **[A3']** for U to be homoskedastic.

Under **[A1]** (or stronger assumptions), the linear model can be rewritten as

$$E[XY] = E[XX']\beta + E[XU] = E[XX']\beta.$$

Under **[A2]**,

$$\beta = E[XX']^{-1}E[XY].$$

Therefore, we propose the OLS estimator using the sample analog

$$\hat{\beta}_{OLS} = \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i'\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} X_i Y_i\right).$$

Note that

$$\hat{\beta}_{OLS} = \beta + \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i'\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} X_i U_i\right).$$

Under **[A1']** or **[A1"]**, $\hat{\beta}$ is unbiased.

By WLLN and CMT, $\hat{\beta}_{OLS} \xrightarrow{p} \beta$. Note that to show consistency, we only need **[A1]** and **[A2]**.

Under **[A1]**, **[A2]**, and **[A3]**,

$$\sqrt{n}(\hat{\beta}_{OLS} - \beta) = \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i'\right)^{-1}\left(\frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_i U_i\right) \xrightarrow{d} N(0, E[XX']^{-1}E[XX'U^2]E[XX']^{-1}),$$

by CMT, WLLN, and CLT. Define the asymptotic covariance matrix as

$$V = E[XX']^{-1}E[XX'U^2]E[XX']^{-1}.$$

Case 1: **[A3']** (homoskedasticity)

Then $V = \sigma^2 E[XX']$. A consistent estimator of V is $\hat{V}_{hom} = \hat{\sigma}^2(n^{-1}\sum_i X_i X_i')$, where $\hat{\sigma}^2 = n^{-1}\sum_i \hat{U}_i^2$ and $\hat{U}_i = Y_i - X_i'\hat{\beta}_{OLS}$. Consistency: By WLLN and CMT

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{1}{n}\sum_i (Y_i - X_i'\hat{\beta}_{OLS})^2 \\
&= \frac{1}{n}\sum_i (X_i'(\beta - \hat{\beta}_{OLS}) + U_i)^2 \\
&= (\beta - \hat{\beta}_{OLS})'\left(\frac{1}{n}\sum_i X_i X_i'\right)(\beta - \hat{\beta}_{OLS}) + 2\left(\frac{1}{n}\sum_i U_i X_i'\right)(\beta - \hat{\beta}_{OLS}) + \frac{1}{n}\sum_i U_i^2
\end{aligned}$$

$\hat{\sigma}^2 \xrightarrow{p} \sigma^2$,

so by WLLN and CMT

$$\hat{V}_{hom} \xrightarrow{p} \sigma^2 E[X_i X_i'].$$

Case 2: **[A3]** (heteroskedasticity)

A consistent estimator of V is the sample analog:

$$\hat{V}_{het} = \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i'\right)^{-1} \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i' \hat{U}_i^2\right) \left(\frac{1}{n}\sum_{i=1}^{n} X_i X_i'\right)^{-1},$$

where the residual $\hat{U}_i = Y_i - X_i'\hat{\beta}_{OLS}$.

1.3.3 Efficiency

An estimator is **efficient** when it achieves the lowest variance possible. Or we can say an estimator is *more efficient* when it has a lower variance than another estimator.

Example 8. For i.i.d. sample $\{X_i\}_{i=1}^n$ where $E[X_i] = \mu < \infty$ and $Var[X_i] = \sigma^2 < \infty$, we propose two estimators for μ, $\hat{\mu}_1 = (n/2)^{-1}\sum_{i\ odd} X_i$ and $\hat{\mu}_2 = \bar{X}_n$. Both estimators are consistent, but $Var[\hat{\mu}_1] = 2\sigma^2/n > \sigma^2/n = Var[\hat{\mu}_2]$. We say $\hat{\mu}_2$ is more efficient than $\hat{\mu}_1$.

Example 9. We say a square matrix $M \in \mathbb{R}^{k\times k}$ is **positive semi-definite** if $\forall x \in \mathbb{R}^k$, $x'Mx \geq 0$. If estimator $\hat{\beta}_1$ and $\hat{\beta}_2$ are k-dimensional vectors, we say $\hat{\beta}_1$ is more efficient than $\hat{\beta}_2$ if $Var[\hat{\beta}_2] - Var[\hat{\beta}_1]$ is positive semi-definite. The intuition is that a linear combination of $\hat{\beta}_1$ will always have a lower variance of the same linear combination of $\hat{\beta}_2$, i.e. $Var[c'\hat{\beta}_2] - Var[c'\hat{\beta}_1] = c'(Var[\hat{\beta}_2] - Var[\hat{\beta}_1])c \geq 0$.

Theorem 5. *(Gauss-Markov Theorem) Assume i.i.d. sampling, $E[U|X] = 0$ and $E[U^2|X] = \sigma^2$. Then the OLS estimator $\hat{\beta}_{OLS}$ is the **best linear unbiased estimator (BLUE)**, i.e. among all estimators $\tilde{\beta}$ of the form $\tilde{\beta} = \sum_{i=1}^n a_i Y_i$ with $a_i = a_i(\{X_i\}_{i=1}^n)$ being a k-dimensional function of the regressors, such that $E[\tilde{\beta}|X_1,\ldots,X_n] = \beta$, we must have $\hat{\beta}_{OLS} = \arg\min_{\tilde{\beta}} Var[\tilde{\beta}|X_1,\ldots,X_n]$.*

Proof. Let $\mathbb{Y} = (Y_1,\ldots,Y_n)'$ and $\mathbb{X} = (X_1,\ldots,X_n)'$. Let $\tilde{\beta} = A\mathbb{Y}$ where A is a function of $\mathbb{X}$. Write

$$\tilde{\beta} = \hat{\beta}_{OLS} + D\mathbb{Y}$$

for $D = A - (\mathbb{X}'\mathbb{X})^{-1}\mathbb{X}$. Then $\tilde{\beta}$ is unbiased only when $E[D\mathbb{Y}|\mathbb{X}] = D\mathbb{X}\beta = 0$ for each β, implying $D\mathbb{X} = 0$.

Thus,

$$\begin{aligned} Var[\tilde{\beta}|\mathbb{X}] &= AVar[\mathbb{Y}|\mathbb{X}]A' \\ &= \sigma^2(D + (\mathbb{X}'\mathbb{X})^{-1}\mathbb{X}')(D + (\mathbb{X}'\mathbb{X})^{-1}\mathbb{X}')' \\ &= \sigma^2 DD' + \sigma^2(\mathbb{X}'\mathbb{X})^{-1} \geq Var[\hat{\beta}_{OLS}|\mathbb{X}]. \end{aligned}$$

□

In more general cases, U is heteroskedastic ($E[U^2|X]$ is a function of X). Suppose $E[U|X] = 0$ and

$E[U^2|X = x] = \sigma^2(x)$, then the linear model can be written as

$$\frac{1}{\sigma(X)}Y = \frac{1}{\sigma(X)}X'\beta + \frac{1}{\sigma(X)}U,$$

or equivalently a transformed linear model such that

$$Y^* = (X^*)'\beta + U^*,$$

where $Y^* = Y/\sigma(X)$, and etc. Note that this is a linear model with homoskedasticity since

$$\textbf{[A1']}: \quad E[\sigma(X)^{-1}U|\sigma(X)^{-1}X] = E[E[\sigma(X)^{-1}U|X, \sigma(X)^{-1}X]|\sigma(X)^{-1}X] = 0.$$
$$\textbf{[A3']}: \quad E[(\sigma(X)^{-1}U)^2|\sigma(X)^{-1}X] = E[E[\sigma(X)^{-2}U^2|X, \sigma(X)^{-1}X]|\sigma(X)^{-1}X] = 1.$$

By Gauss-Markov Theorem, OLS regression of $\sigma(X)^{-1}Y$ on $\sigma(X)^{-1}X$ yields an efficient estimator. This is called the **generalized least square estimator (GLS)**, which is

$$\hat{\beta}_{GLS} = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sigma^2(X_i)}X_iX_i'\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sigma^2(X_i)}X_iY_i\right).$$

Exercise 2. Derive consistency and the limiting distribution for $\hat{\beta}_{GLS}$.

We can show $\hat{\beta}_{GLS}$ is unbiased, consistent, and asymptotically normal. Also, by the Gauss-Markov theorem, for each $\tilde{\beta} = A\mathbb{Y}^*$ such that $E[\tilde{\beta}|\mathbb{X}^*] = \beta$, we must have

$$Var[\tilde{\beta}|X_1^*, \ldots, X_n^*] - Var[\hat{\beta}_{GLS}|X_1^*, \ldots, X_n^*] \geq 0,$$

i.e. the difference between the two matrices is positive semi-definite.

Note that in practice, GLS is *infeasible*, since we don't observe $\sigma^2(X_i)$ from the sample. We now need extra assumptions from our economic model to proceed. For example, if your intuition says the conditional variance is quadratic in X, then you may assume

$$E[U^2|X] = aX^2 + bX + c,$$

for some unknown parameters a, b, c. Note that this is again a conditional expectation relationship, and an OLS estimator of U^2 on X is consistent. Since we do not observe U, we use the OLS regression residuals $\hat{U}_i = Y_i - X_i'\hat{\beta}_{OLS}$ to estimate (a, b, c) and obtain $(\hat{a}, \hat{b}, \hat{c})$. Finally, we replace $\sigma^2(X_i)$ in the formula of GLS by $\hat{\sigma}^2(X_i) = \hat{a}X_i^2 + \hat{b}X_i + \hat{c}$. This is called a **feasible generalized least square estimator (FGLS)**. The general procedure is outlined in the following Algorithm.

Note that in Step 2, there is no guarantee that your estimates will yield a meaningful variance in such that $\hat{\sigma}(X_i) \geq 0$ for each i. Generally, this will not cause problems in the limit as long as your conditional variance is correctly specified.

Algorithm 1 Feasible Generalized Least Squares (FGLS)

1. Do OLS and form the residuals $\hat{U}_i = Y_i - X_i'\hat{\beta}_{OLS}$.
2. Propose a model for conditional variance of the disturbance:

$$E[U^2|X = x] = \sigma^2(x),$$

where $\sigma^2 : \mathbb{R}^k \to \mathbb{R}$. Exploit the relationship between $\hat{U}_i^2$ and X_i to obtain a function estimator $\hat{\sigma}^2(x)$.

3. The FGLS estimator is

$$\hat{\beta}_{FGLS} = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\hat{\sigma}^2(X_i)}X_iX_i'\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\hat{\sigma}^2(X_i)}X_iY_i\right).$$

1.4 MLE, GMM, and M-Estimators

1.4.1 Maximum Likelihood Estimation

Unconditional Likelihood

Suppose we observed a data sample $\{X_i\}_{i=1}^n$, where the joint density is $p_X(x_1, x_2, \ldots, x_n; \theta_0)$ and $\theta \in \Theta$. Assume for each $\theta \in \Theta$, the density function $p_X(x_1, x_2, \ldots, x_n; \theta)$ exists. The **likelihood function** of this sample is defined by $\ell_n(\theta) \equiv \ell_n(\theta|X_1, \ldots, X_n) = p_X(X_1, \ldots, X_n; \theta)$. The **log-likelihood function** is defined by $L_n(\theta) = n^{-1}\log \ell_n(\theta)$. The **Maximum Likelihood (ML) estimator** is defined by

$$\hat{\theta}_{ML} = \arg\max_{\theta\in\Theta} \ell_n(\theta) = \arg\max_{\theta\in\Theta} L_n(\theta).$$

Example 10. Suppose we have i.i.d. sampling and for each i, X_i has density $p(x; \theta_0)$. Then the likelihood is $\ell_n(\theta) = \prod_{i=1}^n p(X_i; \theta)$. The log-likelihood is $L_n(\theta) = n^{-1}\sum_{i=1}^n \log p(X_i; \theta)$.

Example 11. Suppose the data sample is not i.i.d. and follows AR(1) such that $X_i = \rho X_{i-1} + U_i$, $U_i \overset{i.i.d.}{\sim} N(0, \sigma^2)$ and $X_0 = 0$ is non-random. The parameter of interest is $\theta = (\rho, \sigma^2)$. Then, the density can be written as

$$p_X(x_1, \ldots, x_n; \theta) = p_{X_n|X_1,\ldots,X_{n-1}}(x_n|x_1, \ldots, x_{n-1}; \theta) \times \cdots \times p_{X_1}(x_1|\theta) = \prod_{i=1}^{n} p_{X_i|X_{i-1}}(x_i|x_{i-1}; \theta),$$

where $p_{X_i|X_{i-1}}$ is the density of $N(\rho x_{i-1}, \sigma^2)$.

Example 12. Let X_i follow i.i.d. uniform distribution on $[0, \theta_0]$ with $0 < \theta_0 < \infty$. Then, $p(x; \theta) = \theta^{-1}\mathbb{1}\{0 \le x \le \theta\}$ and $\ell_n(\theta) = \theta^{-n}\mathbb{1}\{0 \le X_1, \ldots, X_n \le \theta\}$. The ML estimator for θ_0 is $\hat{\theta}_{ML} = \max_i X_i$.

Conditional Likelihood

Suppose now we have data on both X and Y with the sample $\{(Y_i, X_i)\}_{i=1}^n$. The condition density of all Y_i's on all X_i's is $p_{Y|X}(y_1, \ldots, y_n|x_1, \ldots, x_n; \theta_0)$. The conditional likelihood is defined by

$$\ell_n(\theta) = p_{Y|X}(Y_1, \ldots, Y_n|X_1, \ldots, X_n; \theta).$$

The ML estimator is defined accordingly. Similarly, if the sample is i.i.d. with conditional density $p_{Y|X}(y|x; \theta)$, the likelihood function is just $\ell_n(\theta) = \prod_{i=1}^n p(Y_i|X_i; \theta)$ and the log-likelihood is $L_n(\theta) = n^{-1} \sum_{i=1}^n \log p(Y_i|X_i; \theta)$.

Example 13. We revisit the linear model $Y = X'\beta + U$ and assume $X \perp U$ and $U \overset{i.i.d.}{\sim} N(0, \sigma^2)$. Let the parameter of interest be $\theta = (\beta, \sigma^2)$. The conditional density is

$$p(y|x; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - x'\beta)^2}{2\sigma^2}\right)$$

and the log-likelihood is

$$L_n(\theta) = \frac{1}{n} \sum_{i=1}^n -\log(\sqrt{2\pi}\sigma) - \frac{(y - x'\beta)^2}{2\sigma^2}.$$

We take the first order condition w.r.t. β of the log-likelihood and that gives us that $\sum_{i=1}^n X_i(Y_i - X_i'\beta) = 0$. Solving the first order condition to β yields that

$$\hat{\beta}_{ML} = \left(\frac{1}{n} \sum_{i=1}^n X_i X_i'\right)^{-1} \frac{1}{n} \sum_{i=1}^n X_i Y_i = \hat{\beta}_{OLS}.$$

This equivalence of the maximum likelihood estimator and the OLS estimator is generally not true but holds in the this specific setup. Because ML estimation requires extra distributional assumptions, the ML estimator is expected to outperform estimators that do not require these types of assumptions.

Exercise 3. Derive the formula of $\hat{\sigma}^2_{ML}$.

Example 14. Suppose

$$Y_i = \mathbb{1}\{X'\theta_0 + U \geq 0\}$$

where U follows some distribution F. Then the conditional density is

$$p(y|x; \theta) = (1 - F(-x'\theta))^y F(-x'\theta)^{1-y}$$

and the log-likelihood is

$$L_n(\theta) = \frac{1}{n} \sum_{i=1}^n Y_i \log(1 - F(-X_i'\theta)) + (1 - Y_i) \log F(-X_i'\theta).$$

In general, there is no analytical solution to the maximization problem. If F is standard normal, this model is called a Probit model. It is called a Logit model when

$$F(u) = \frac{\exp(u)}{(1 + \exp(u))}.$$

The ML estimator can be solved efficiently in these two models.

1.4.2 Generalized Method of Moments

Method of moments are for a class of estimators that are solved for by *equating the sample analog of the moments to the populations ones.* Common estimators can be written under a method of moments framework. Examples include

[OLS]: $E[X(Y - X_i'\beta)] = 0$

[IV]: $E[Z(Y - X_i'\beta)] = 0$

[2SLS]: $\pi' E[Z(Y - X_i'\beta)] = 0$, where $\pi = E[ZZ']^{-1}E[ZX]$.

[ML]: $E[\frac{\partial}{\partial\theta}\log p(Y|X;\theta)] = 0$

In general cases, the solution to the sample analog equations does not necessarily exist. Suppose we have J moments where $E[m_j(X, Y; \theta_0)] = 0$ for $j = 1, \ldots, J$. Let $m = (m_1, \ldots, m_J)'$. The **GMM estimator** is defined by

$$\hat{\theta}_{GMM} = \arg\min_{\theta\in\Theta} \left(\frac{1}{n}\sum_{i=1}^{n} m(X_i, Y_i; \theta)\right)' W \left(\frac{1}{n}\sum_{i=1}^{n} m(X_i, Y_i; \theta)\right),$$

for some weighting matrix W.

Exercise 4. Write out OLS, IV, 2SLS, and ML as a GMM estimator.

Consistency and asymptotic normality can be established for each W, but the question becomes how can we choose W? It can be shown that efficiency bound is achieved when $W = E[m(X, Y; \theta_0)m(X, Y; \theta_0)']^{-1}$, where θ_0 is the true underlying parameter. Notice the similarity to GLS (generalized least square) with this choice, and the optimal weighting matrix is also infeasible. We can use a **feasible 2-step GMM** estimator outlines in the following algorithm.

Algorithm 2 Feasible 2-Step GMM

Step 1: Perform GMM estimation using identity weighting matrix and obtain $\hat{\theta}_1$.
Step 2: Use the sample analog of the efficient weighting matrix

$$\hat{W} = \left(n^{-1}\sum_{i=1}^{n} m(X_i, Y_i; \hat{\theta}_1)m(X_i, Y_i; \hat{\theta}_1)'\right)^{-1}$$

and perform the GMM estimation again to attain $\hat{\theta}_2$

The resulting estimator, $\hat{\theta}_2$, is the 2-step GMM estimator.

These two steps can be further iterated for the **iterative GMM estimator**.

1.4.3 M-estimator

An M-**estimator** is defined by minimizing a sum over a function of the sample, i.e.

$$\hat{\theta}_M = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} f(X_i, Y_i; \theta).$$

Examples of M-estimators are OLS and ML estimators.

A much more general version of this estimator is the **Extremum estimator**, which includes many popular estimators, including OLS, ML, GMM, and other estimators. It is defined by minimizing some criterion function, which needs not always be a sum.

$$\hat{\theta}_{EE} = \arg\min_{\theta \in \Theta} \hat{Q}(\theta),$$

where $\hat{Q}$ is a function of the data. With some regularity conditions, consistency and asymptotic normality can be established.

2 Topics

2.1 Time Series

A time series is a set of repeated observations of the same variable, such as a country's GDP. We can denote it as $\{x_1, x_2, \ldots, x_T\}$or $\{x_t\}$ for $t = 1, \ldots T$. Here we let x_t to be a random variable. As such, we can apply the standard econometric tools from the first section to analyze x_t. However, the twist that makes it interesting is that we no longer assume that x_t is i.i.d. and allow for dependence across time. Intuitively, this means that GDP at time t may depend on GDP at time $t-1$.

We focus on using parametric models for the joint distribution $\{x_t\}$, which allows us to use a few parameters to characterize the model. Standard time series models include ARMA and ARCH type of models and useful tools in time series econometrics include vector auto-regression (VAR), stationarity, unit roots, impulse response functions, error-correction models, martingales, and co-integration. Time series econometrics have a have emphasis on the VAR framework of representing time series models. Thus, it would be useful to brush up on linear algebra (i.e. eigenvalues/eigenvectors, SVD, Cholesky Decomposition) for those interested in further exploring this topic. Regardless, we will focus on just linear ARMA models for this introductory section.

We first define a **white noise process** ϵ_t. A standard assumption for white noise is $\epsilon_t \overset{i.i.d.}{\sim} N(0, \sigma_\epsilon^2)$. This assumption nests three main implications.

1. $E[\epsilon_t] = E[\epsilon_t|\epsilon_{t-1}, \epsilon_{t-2}, \ldots] = E[\epsilon_t|\mathcal{I}_{t-1}] = 0$ where $\mathcal{I}_{t-1}$ represents the information set at period $t-1$
2. $E[\epsilon_t \epsilon_{t-j}] = Cov(\epsilon_t \epsilon_{t-j}) = 0, \ \forall j \neq t$
3. $Var(\epsilon_t) = Var(\epsilon_t|\epsilon_{t-1}, \epsilon_{t-2}, \ldots) = Var(\epsilon_t|\mathcal{I}_{t-1}) = \sigma_\epsilon^2$

Implications (1) and (2) imply that there is no **serial correlation** or predictability of ϵ_t from past values. Implication (3) implies **conditional homoskedasticity**, which means conditional on the past information, there is a constant conditional variance. Note that Implication (1) itself implies ϵ_t is a martingale difference sequence.

We will focus on linear combinations of the white noise process. These models include the auto-regression, AR(p), moving average, MA(q), and auto-regression moving average, ARMA(p,q), models.

$$
\begin{aligned}
AR(1):\ & x_t = \phi x_{t-1} + \epsilon_t \\
MA(1):\ & x_t = \epsilon_t + \theta \epsilon_{t-1} \\
AR(p):\ & x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + \epsilon_t \\
MA(q):\ & x_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \\
ARMA(p,q):\ & x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + \epsilon_t + \theta \epsilon_{t-1} + \cdots \theta_q \epsilon_{t-q}
\end{aligned}
$$

These three types of models all are additive combinations of x_t, ϵ_t and their past values.

Remark.

1. These models are assumed to describe a time series with mean zero. Any means will be absorbed by a constant in the time series.
2. An AR(1) model with $|\phi| < 1$ is stationary, $|\phi| = 1$ has a unit root, and $|\phi| > 1$ is non-stationary/explosive. To see what these imply let $x_0 = 2$ plot out the subsequent time series.

Example 15. An stationary AR(1) model can be written as MA(∞) model by recursively substituting.

$$
\begin{aligned}
x_t &= \phi x_{t-1} + \epsilon_t \\
&= \phi(\phi x_{t-2} + \epsilon_{t-1}) + \epsilon_t = \phi^2 x_{t-2} + \phi \epsilon_{t-1} + \epsilon_t \\
&= \phi^k x_{t-k} + \phi^{k-1} \epsilon_{t-k+1} + \cdots + \phi^2 \epsilon_{t-2} + \phi \epsilon_{t-1} + \epsilon_t
\end{aligned}
$$

So far we have written the AR(1) as a ARMA(k,$k-1$). If we let $|\phi| < 1$, then $\lim_{k\to\infty} \phi^k x_{t-k} = 0$ so

$$x_t = \sum_{j=0}^{\infty} \phi^j \epsilon_{t-j}$$

which is a MA(∞).

We can also re-write the ARMA class of models with the **lag operator** L. Effectively, applying L yields

$$Lx_t = x_{t-1}$$

and moves the time index back. L is an operator that takes in a entire time series ($\{x_t\}_{t=1}^T$) and yields another ($\{x_{t-1}\}_{t=1}^T$). The lag operator can be further generated for some integer j

$$
\begin{aligned}
L^j x_t &= x_{t-j} \\
L^{-j} x_t &= x_{t+j}
\end{aligned}
$$

and we can construct **lag polynomials**,

$$a(L)x_t = (a_0 L^0 + a_1 L^3)x_t = a_0 x_t + a_1 x_{t-3}.$$

In turn, we can collapse the ARMA models to

$$\begin{aligned} AR:&\ a(L)x_t = \epsilon_t \\ MA:&\ x_t = b(L)\epsilon_t \\ ARMA:&\ a(L)x_t = b(L)\epsilon_t \end{aligned}$$

These lag polynomials have the following general properties:

1. Multiplicativity: $a(L)b(L) = (a_0 + aL)(b_0 + bL) = a_0b_0 + (a_0b + ab_0)L + abL^2$
2. Commutation: $a(L)b(L) = b(L)a(L)$
3. Integer powers: $a(L)^2 = a(L)a(L)$
4. Inversion: $a(L) = (1-\lambda_1 L)(1-\lambda_2 L) \iff a(L)^{-1} = (1-\lambda_1 L)^{-1}(1-\lambda_2 L)^{-1} = \sum_{j=0}^{\infty} \lambda_1^j L^j \sum_{j=0}^{\infty} \lambda_2^j L^j = c_1(1-\lambda_1)^{-1} + c_2(1-\lambda_2)^{-1}$ for some constants $c_1, c_2 \in \mathbb{R}$.

Exercise 5. Show that a MA(1) models can written as an AR(∞) with Lag operators. For example, note that the AR(1) can be inverted as

$$(1-\phi L)x_t = \epsilon_t \iff x_t = (1-\phi L)^{-1}\epsilon_t.$$

Did you need to assume that $|\theta| < 1$ in the MA(1) model for the inversion to work?

In the multivariate case, various ARMA models can be stacked into a VAR(1) model (or written as a VAR(1)). Let x_t be a multivariate time series,

$$x_t = \begin{bmatrix} y_t \\ z_t \end{bmatrix}$$

with $\epsilon_t \overset{iid}{\sim} N(0, \Sigma)$ where

$$\epsilon_t = \begin{bmatrix} \delta_t \\ \nu_t \end{bmatrix},\ E[\epsilon_t] = 0,\ E[\epsilon_t \epsilon_t'] = \Sigma = \begin{bmatrix} \sigma_\delta^2 & \sigma_{\delta\nu} \\ \sigma_{\delta\nu} & \sigma_\nu^2 \end{bmatrix},\ E[\epsilon_t \epsilon_{t-j}'] = 0$$

for some integer $j \neq t$. The VAR(1) is then $x_t = \phi x_{t-1} + \epsilon_t$ or equivalently

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} \phi_{yy} & \phi_{yz} \\ \phi_{zy} & \phi_{zz} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} \delta_t \\ \nu_t \end{bmatrix}$$

or

$$\begin{aligned} y_t =& \phi_{yy} y_{t-1} + \phi_{yz} z_{t-1} + \delta_t \\ z_t =& \phi_{zy} y_{t-1} + \phi_{zz} z_{t-1} + \nu_t. \end{aligned}$$

Note that the lags for y and z appear in the top and bottom equations and the VAR(1) captures the cross-variable time series dynamics. Multivariate ARMA models can be manipulated and inverted in a similar

manner to the univariate case. The VAR approach implies that time series analysis heavily uses linear algebra and matrix operations.

Exercise 6. Stack three time series together in a VAR(1)

2.2 Bayesian Approach

The econometrics treatment in Section 1 is mainly frequentist. The recipe that we followed when proposing a new estimator is appeal to asymptotics ($n \to \infty$) with the WLLN and CLT to respectively derive the consistency and limiting distribution of the proposed estimator. This approach notably (1) does not use our prior knowledge about what the estimator (2) requires n to tend to infinity for our confidence intervals to hold.

If we have domain knowledge about the problem or have a sense of the what the estimate will be like from prior studies or data, we may want to put a **prior** distribution on the estimator. The prior should bake in the our a priori beliefs on the estimate and should be set ideally before looking at the data. The **likelihood** is then estimated from the data as from Section 1.4. The **posterior** distribution is the probability distribution of the estimate after considering your prior and your data/likelihood estimate.

More formally, we let θ be the parameter of interest, $\theta \sim \pi(\theta)$ be the prior, and $f(x|\theta)$ be the likelihood. Then the posterior density $\pi(\theta|x)$ arises from **Bayes Theorem**

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int f(x|\theta)\pi(\theta)d\theta} = \frac{1}{Z(x)} f(x|\theta)\pi(\theta) \propto f(x|\theta)\pi(\theta)$$

where $Z(x) = \int f(x|\theta)\pi(\theta)d\theta$ is the marginal density of x and is also called the partition function.

The Bayesian Approach is often termed "*finite sample precise*". This means the posterior distribution will hold for a finite n. Further, as $n \to \infty$, we should recover the frequentist estimate by *Bernstein-von Mises Theorem*. Intuitively, this means that as n tends to infinity, the likelihood should dominate the posterior and the prior should have little to no influence on the posterior. In applied work, $n \to \infty$ is unattainable so a frequentist approach implicitly assumes that the asymptotic case is a good assumption to the problem for quantifying uncertainty. If it is not, then *finite-sample bias* may occur and the subsequent estimate can biased.

Since the approach requires a stance (or choice) on the distribution of the prior and likelihood, the full posterior can be informative and can be used for *Bayesian Decision Theory*. Often the posterior mode or mean is used as a point estimate and the 95% high probability posterior interval (or credible region or interval) is used for uncertainty quantification. More generally, knowledge of the posterior distribution can also let us estimate **posterior expectations** for some function (or decision) of the parameters $h(\theta)$,

$$E_{\pi(\theta|x)}[h(\theta)] = \int h(\theta)\pi(\theta|x)d\theta$$

and to get the posterior mean we would choose $h(\theta) = \theta$.

Example 16. (Bayesian Linear Regression) We consider the linear regression of response variable y on a vector of p covariates X, and have

$$y = X'\beta + \epsilon$$

where $\epsilon \overset{i.i.d.}{\sim} N(0, \sigma^2 I)$ where $\sigma^2 > 0$ is a constant and known a priori. I is the identity matrix that has dimensions $p \times p$. β is the coefficient of interest.

Then from our modeling assumptions, the response variable is generated from the distribution

$$y \sim N(X'\beta, \sigma^2 I).$$

We impose prior $P(\beta|X)$ on β and have likelihood $P(y|\beta, X)$. Then, Bayes Rule yields the posterior

$$P(\beta|y, X) = \frac{P(y|\beta, X)P(\beta|X)}{\int P(y|\beta, X)P(\beta|X)d\beta} = \frac{P(y|\beta, X)P(\beta|X)}{P(y|X)} \propto P(y|\beta, X)P(\beta|X).$$

As the number of data points tend to infinity, we should expect the likelihood to wash out the prior and thus recover the standard OLS/ML estimate in the linear regression setup.

Example 17. (Bernstein-von Mises Theorem Intuition) To provide intuition for the "likelihood washes out the prior as the number of data points go to infinity" we will work out the following linear regression example where the Bayesian MAP (maximum a posteriori) estimate is the OLS estimate as $n \to \infty$. The Bayesian MAP estimate is the mode of the posterior distribution.

We consider linear model $Y = X\beta + U$ where we observe data $\{x_i, y_i\}_{i=1}^n$. Our data set is of size n and for simplicity we have only one variable in X so β is a one dimensional parameter. We assume $U \overset{i.i.d.}{\sim} N(0, 1)$ and impose a prior $\pi(\beta|X) = \pi(\beta) = N(0, 1)$ on β. We will show the MAP estimate will be same as the OLS estimate of β, or $\hat{\beta} = (E[X'X])^{-1}E[XY]$ as $n \to \infty$.

Since $U \overset{i.i.d.}{\sim} N(0, 1)$ so $Y \overset{i.i.d.}{\sim} N(X\beta, 1)$. Then, the conditional density is

$$p(y_i|x_i; \beta) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y_i - x_i\beta)^2}{2})$$

and our likelihood is

$$f(y|\beta, X) = \frac{1}{\sqrt{2\pi}} \prod_{i=1}^{n} \exp(-\frac{(y_i - x_i\beta)^2}{2}).$$

Our prior on β is $N(0, 1)$, which has density

$$P(\beta|X) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\beta^2}{2}).$$

Then from Bayes' Rule, we have the posterior,

$$\begin{aligned} P(\beta|y, X) = \frac{f(y|\beta, X)P(\beta|X)}{\int f(y|\beta, X)P(\beta|X)d\beta} &\propto f(y|\beta, X)P(\beta|X) \\ &\propto \frac{1}{2\pi} \exp(-\frac{\beta^2}{2}) \prod_{i=1}^{n} \exp(-\frac{(y_i - x_i\beta)^2}{2}) \\ &\propto \frac{1}{2\pi} \exp\big(-\frac{\beta^2}{2} - \sum_{i=1}^{n} \frac{(y_i - x_i\beta)^2}{2}\big) \end{aligned}$$

The MAP estimate is the β value that maximizes the posterior density (or the posterior mode), then

$$\begin{aligned}
\hat{\beta}_{MAP} &= \arg\max_{\beta} P(\beta|y,X) \\
&= \arg\max_{\beta} \frac{1}{n} \log\big(P(\beta|y,X)\big) \\
&= \arg\max_{\beta} \frac{1}{n} \log\left(\frac{1}{2\pi} \exp\big(-\frac{\beta^2}{2} - \sum_{i=1}^{n} \frac{(y_i - x_i\beta)^2}{2}\big)\right) \\
&= \arg\max_{\beta} \frac{1}{2n}\big(-\beta^2 - \sum_{i=1}^{n}(y_i - x_i\beta)^2\big).
\end{aligned}$$

Taking the FOC to β,

$$\begin{aligned}
\frac{1}{2n}(-2\beta + 2\sum_{i=1}^{n} x_i(y_i - x_i\beta)) &= 0 \\
-\frac{1}{n}\beta + \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \beta\frac{1}{n}\sum_{i=1}^{n} x_i x_i &= 0 \\
\frac{1}{n}\beta + \beta\frac{1}{n}\sum_{i=1}^{n} x_i x_i &= \frac{1}{n}\sum_{i=1}^{n} x_i y_i \\
\beta(\frac{1}{n} + \frac{1}{n}\sum_{i=1}^{n} x_i x_i) &= \frac{1}{n}\sum_{i=1}^{n} x_i y_i \\
\hat{\beta}_{MAP} &= (\frac{1}{n} + \frac{1}{n}\sum_{i=1}^{n} x_i x_i)^{-1}\frac{1}{n}\sum_{i=1}^{n} x_i y_i.
\end{aligned}$$

Now consider the asymptotic case ($n \to \infty$) and using the WLLN,

$$\begin{aligned}
\frac{1}{n} &\xrightarrow{p} 0 \\
\frac{1}{n}\sum_{i=1}^{n} x_i x_i &\xrightarrow{p} E[XX] \\
\frac{1}{n}\sum_{i=1}^{n} x_i y_i &\xrightarrow{p} E[XY]
\end{aligned}$$

Assuming that $(0 + E[XX]) = E[XX] < \infty$ or is non-singular and using the CMT, then

$$\operatorname*{plim}_{n\to\infty} \hat{\beta}_{MAP} = (E[XX])^{-1}E[XY] = \beta = \operatorname*{plim}_{n\to\infty} \hat{\beta}_{OLS}$$

which the same as our OLS estimate for β in the probability limit. We see that the effect of prior is "washed out by the likelihood" because the term in the FOC from the prior, $\frac{1}{n}$, goes to 0 as $n \to \infty$.

Notice that in our setup that $\hat{\beta}_{OLS} = \hat{\beta}_{ML}$. We had also showed this more generally in our example with the linear model in the conditional likelihood subsection of the maximum likelihood estimation section. Generally, the Bernstein-von Mises Theorem implies the *MAP estimate converges to the MLE estimate* as

the number of data points go to infinity.

Modern methods in Bayesian Estimation often implement Markov Chain Monte Carlo (MCMC). Since different combinations of prior distributions and likelihood functions may not be analytically tractable different MCMC samplers are used to sample from the posterior of interest.

Lastly, different models can have computational advantages from using a Frequentist or Bayesian Approach. For example, in discrete choice, the Logit model is computationally easy to estimate with a ML Estimator, but has a posterior distribution that is difficult to sample from. In comparison, the Probit model is easy to sample from but is computationally difficult to estimate with an ML Estimator. As such, applied researcher may choose the computationally easier approach and then appeal to asymptotics and Bernstein-von Mises Theorem for an approximate equivalence of the Frequentist and Bayesian Approaches.

2.3 Bootstrap Methods

Bootstrap methods, especially the non-parametric bootstrap, are often used by applied researchers to compute the standard error of a complicated model. These methods fall under a class of resampling techniques, which also include the jackknife and sub-sampling. Resampling techniques uses the sampling information from the empirical distribution of the data. The drawbacks of these techniques are they require much more computational power and often have more difficult theory. Within the bootstrap, many different types of methods exist and we will focus on the non-parametric bootstrap.

The **bootstrap** is the distribution acquired by estimation on samples from i.i.d. sampling with replacement from the original data set. Suppose that I have a sample of 10 PhD students in the math camp and wanted to determine the average height of the students. If we index each student from 1 to 10, then each bootstrap sample will be a drawn with replacement of the original 10 PhD students. The students in the original data set are indexed $\{1, 2, \ldots, 10\}$, and the first bootstrap sample could be students $\{1, 1, 1, 2, 6, 7, 8, 8, 9, 10\}$. Note that in the first bootstrap sample, student 1 appears three times and student 8 appears twice. Students 3, 4, 5 do not appear at all.

Remark. We can compute the probability that an individual observation appears in the bootstrap sample.

$$P(\text{Observation in Bootstrap Sample}) = 1 - (1 - \frac{1}{n})^n \to 1 - \exp(-1) \approx 0.632$$

where the limit is taken as $n \to \infty$. This means that around 37% of unique observations in the original data do not appear in the bootstrap sample.

Continuing with our height example, we denote student i's height as h_i. Then, the average height in the original data is $\bar{h} = \frac{1}{10}\sum_{i=1}^{10} h_i$ and for each bootstrap iteration b, the average height is $\bar{h}^b = \frac{1}{10}\sum_{i=1}^{10} h_i^b$ where the b superscript denotes the bootstrap sample. $\bar{h}^b$ is just the average height for the students in bootstrap sample b. The total number of bootstrap replications is B for $b \in \{1, \ldots, B\}$. Often, applied researchers set B to be a large number, say $B = 100$, 1000, or even 10000, and this choice often depends on the computational resources available to the researcher.

The bootstrap estimator is

$$\hat{h}_B = \frac{1}{B}\sum_{b=1}^{B} \bar{h}^b$$

or the average of the bootstrap estimators across the bootstrap iterations. The variance of the estimator is then

$$\hat{Var}(\hat{h}_B) = \frac{1}{B-1}\sum_{b=1}^{B}(\bar{h}^b - \hat{h}_B)^2$$

which is the sample variance across bootstrap iterations. The standard error of the estimator is

$$se(\hat{h}_B) = \sqrt{\hat{Var}(\hat{h}_B)}.$$

Lastly, the normal-approximation bootstrap confidence intervals are

$$C^{boot} = [\hat{h}_B - z_{1-\alpha/2}se(\hat{h}_B), \hat{h}_B + z_{1-\alpha/2}se(\hat{h}_B)]$$

where $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard normal distribution.

Remark.

1. The steps for computing the variance and the standard error are similar to the standard computation procedures for the variance and standard errors but are over bootstrap iterations instead of observations.
2. The bootstrap confidence interval is also similar to the asymptotic confidence interval but instead uses the bootstrap standard error instead of the asymptotic standard error. The normal-approximation here relies on the normal approximation to t-ratio and can be inaccurate in finite samples. Other methods, such as the bias-corrected percentile method, can have better estimates.
3. Bootstrap standard errors should be considered as tools to analyze precision than to construct robust confidence intervals. Since B is finite, all the bootstrap statistics are estimates and thus are random. Further, they will vary across simulations and choices of B. Thus, different researchers using the same bootstrap iterations should get slightly different results up to a simulation sampling error.

To generalize our definitions of the bootstrap, we let θ be a vector of the parameters of interest Each bootstrap draw yields an **bootstrap estimate** of θ which is denoted as $\hat{\theta}^b$. The **bootstrap mean** is

$$\bar{\theta}_B = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^b.$$

The **variance** of the estimator is

$$\hat{Var}(\bar{\theta}_B) = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^b - \bar{\theta}_B)(\hat{\theta}^b - \bar{\theta}_B)'.$$

The **standard error** of the estimator is

$$\hat{se}(\bar{\theta}_B) = \sqrt{\hat{Var}(\bar{\theta}_B)}.$$

Lastly, the **normal-approximation bootstrap confidence intervals** are

$$C^{boot} = [\bar{\theta}_B - z_{1-\alpha/2}\hat{se}(\bar{\theta}_B), \bar{\theta}_B + z_{1-\alpha/2}\hat{se}(\bar{\theta}_B)]$$

where $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard normal distribution.

Remark.

1. There are notable examples where bootstrap fails to work. One example is estimating an order statistic. In our example, consider the performance of the bootstrap when the we want the estimator to be the maximum height across students.
2. Bootstrap asymptotics are taken over the number of bootstrap iterations, i.e. as $B \to \infty$. However, we would need to be careful since the bootstrap mean has a distribution conditional to the original data.

3 Example

In the example, we will walk though a derivation of the key properties of an estimator as well as compare biased and unbiased estimators' performance in a simulation. This example serves two roles. The first is to provide an example of a theoretical treatment of deriving the properties of a proposed ML estimator. The second is to illustrate a Monte Carlo Simulation approach of examining two different estimators' properties. Please see the RMarkdown Notebook for the example.

Estimator Example

Booth Math Camp (Autumn 2021)

Walter W. Zhang

July 01, 2021

In this example, we will walk though a derivation of the key properties of an estimator as well as compare biased and unbiased estimators' performance in a simulation.[1]

Contents


```
# Load Packages
require(knitr)
require(kableExtra)
require(data.table)
```

Consider the following scenario.

Let $X_1, \ldots, X_n$ be a sequence of i.i.d. sequence of random variables with distribution $Unif(\theta, 2\theta)$ with $\theta > 0$.

[1] This example is derived from an Azeem Shaikh exercise.

MLE Estimator Properties

We first want to show the MLE estimator of θ is

$$\hat{\theta} = \frac{1}{2} \max_{1 \le i \le n} X_i.$$

For each observation,

$$p_\theta(x_i) = \begin{cases} \frac{1}{\theta} & \text{if } \theta \le x_i \le 2\theta \\ 0 & \text{otherwise} \end{cases}$$

So, the likelihood function is

$$l_n(\theta) = \prod_{i=1}^{n} p_\theta(x_i) = \prod_{i=1}^{n} \frac{1}{\theta} I\{\theta \le x_i \le 2\theta\} = \frac{1}{\theta^n} \prod_{i=1}^{n} I\{\theta \le x_i \le 2\theta\}$$

If any x_i falls out of the range $[\theta, 2\theta]$, the indication function is value 0. Thus, we want the smallest θ s.t. $\hat{\theta}_n \le X_i \le 2\hat{\theta}_n, \forall i$.

Then, for all i, we have that $\theta \le X_i \le 2\theta$, thus $\max X_i \le 2\theta$. So,

$$\frac{1}{2} \max X_i \le \theta$$

$$\frac{1}{2} \max X_i \le \theta \le X_i$$

Thus $X_i \ge \frac{1}{2} \max X_i$, the smallest $\hat{\theta}_n$ satisfies $\hat{\theta}_n \le X_i \le 2\hat{\theta}_n$ for all i is

$$\hat{\theta}_{MLE} = \hat{\theta}_n = \frac{1}{2} \max_{1 \le i \le n} X_i$$

Biased Estimator

We now show that the MLE estimator above is biased.

The highest value X_i can take is 2θ. $E[\max_{1 \le i \le n} X_i] = 2\theta$ iff $P(\max_{1 \le i \le n} X_i = 2\theta) = 1$

$$P(\max_{1 \le i \le n} X_i = 2\theta) = 1 - P(X_1, X_2, ..., X_n < 2\theta) = 1 - P(X_i < 2\theta)^n$$

Since $P(X_i < 2\theta) < 1$, we have

$$E[\hat{\theta}_n] = \frac{1}{2} E[\max_{1 \le i \le n} X_i] < \theta$$

So $\hat{\theta}_n$ is biased.

Consistency

We now want to show that $\hat{\theta}_n$ is a consistent estimator to θ.

To show that $\hat{\theta}_n$ is a consistent estimator, we need to show that as $n \to \infty$

$$P(|\hat{\theta}_n - \theta| > \epsilon) \to 0$$

Since $2\hat{\theta}_n - 2\theta \leq 0$,

$$\begin{aligned}
P(|\hat{\theta}_n - \theta| > \epsilon) &= P(|2\hat{\theta}_n - 2\theta| > 2\epsilon) \\
&= P(-2\hat{\theta}_n + 2\theta > 2\epsilon) \\
&= P(2\hat{\theta}_n < 2\theta - 2\epsilon) \\
&= P(\max_{1\leq i\leq n} X_i < 2\theta - 2\epsilon) \\
&= P(X_i < 2\theta - 2\epsilon)^n \\
&= (\frac{2\theta - 2\epsilon - \theta}{\theta})^n \\
&= (1 - \frac{2\epsilon}{\theta})^n
\end{aligned}$$

Since $0 < \frac{2\epsilon}{\theta} < 1$, we have $(1 - \frac{2\epsilon}{\theta})^n \to 0$ as $n \to \infty$ So,

$$P(|\hat{\theta}_n - \theta| > \epsilon) \to 0$$

$\hat{\theta}_n$ is a consistent estimator of θ.

Limiting Distribution

We now derive the limiting distribution for the estimator. We will show that $n(\theta - \hat{\theta}_n)$ converges in distribution to an exponential distribution parameterized by λ as $n \to \infty$. Here λ will be expressed in terms of θ, and the exponential CDF is

$$F(x) = \begin{cases} 0 & \text{if x} < 0 \\ 1 - \exp(-x/\lambda) & \text{if x} \geq 0 \end{cases}.$$

We also note that

$$\lim_{n\to\infty} (1 - \frac{c}{n})^n = \exp(-c)$$

for some constant $c \in \mathbb{R}$.

Let $X_n = n(\theta - \hat{\theta}_n)$, $X \sim [\lambda]$. We need to show that $P(X_n \leq x) \to P(X \leq x)$

$$\begin{aligned}
P(n(\theta - \hat{\theta}_n) \leq x) &= P(\theta - \hat{\theta}_n \leq \frac{x}{n}) \\
&= P(\hat{\theta}_n \geq \theta - \frac{x}{n}) \\
&= P(\frac{1}{2}\max_{1\leq i\leq n} X_i \geq \theta - \frac{x}{n}) \\
&= 1 - P(\max_{1\leq i\leq n} X_i \leq 2\theta - \frac{2x}{n}) \\
&= 1 - P(X_i \leq 2\theta - \frac{2x}{n})^n \\
&= 1 - (\frac{2\theta - \frac{2x}{n} - \theta}{\theta})^n \\
&= 1 - (1 - \frac{\frac{2x}{\theta}}{n})^n
\end{aligned}$$

Since

$$\lim_{n\to\infty} (1 - \frac{c}{n})^n = \exp(-c)$$

We have

$$P(n(\theta - \hat{\theta}_n) \leq x) \to 1 - \exp(-\frac{2x}{n})$$

Set $\lambda = \frac{\theta}{2}$, we have $n(\theta - \hat{\theta}_n)$ converges in distribution to an exponential distribution with $\lambda = \frac{\theta}{2}$.

Confidence Interval

We now want to derive the 95% confidence interval (CI) for the MLE estimator $\hat{\theta}_n$.

Define c_n to be the 95% quantile of the exponential distribution with $\lambda = \frac{\theta}{2}$.

$$P(n(\theta - \hat{\theta}_n) \leq c) \to 0.95$$

Since

$$n(\theta - \hat{\theta}_n) \xrightarrow{d} \exp(\frac{\theta}{2})$$

We have

$$P(n(\theta - \hat{\theta}_n) \leq c) = P(\theta \leq \hat{\theta}_n + \frac{c}{n}) \to 0.95$$

The 95% confidence interval is given by

$$C_n = [\hat{\theta}_n, \hat{\theta}_n + \frac{c}{n}]$$

Unbiased Estimator

We now want to suggest an unbiased estimator, $\tilde{\theta}_n$, for θ.

We know that $E[X_i] = \frac{3}{2}\theta$ from before. Then, we consider

$$\tilde{\theta}_n = \frac{2}{3}\frac{1}{n}\sum_{i=1}^{n} X_i$$

We see the estimator is unbiased since

$$E[\tilde{\theta}_n] = \frac{2}{3n}\sum_{i=1}^{n} E[X_i] = \frac{2}{3n}\frac{3}{2}\sum_{i=1}^{n}\theta = \theta.$$

Exercise

As an exercise you can show consistency of the unbiased estimator and derive its limiting distribution and 95% CI.

Simulation Study

We consider a simulation study to compare the biased ML estimator and the unbiased estimator.

Monte Carlo Simulation

The simulation will consider $\theta \in \{0.8, 1, 10\}$, $n \in \{2, 5, 25, 100\}$, and for $m = 10^4$ replications. We fix the seed before we run the simulation to ensure our results are replicable. Also, we let $\alpha = 0.05$ for the simulation.

```
# Define Simulation Parameters
theta_vec <- c(0.8, 1, 10)
n_vec     <- c(2, 5, 25, 100)
m         <- 1e4L
base_seed <- 1234L

set.seed(base_seed)
```

Our simulation will take the following steps:

i. Draw n i.i.d. observations from $Unif(\theta, 2\theta)$
ii. Compute $\hat{\theta}_n$ and $\tilde{\theta}_n$
iii. Compute the MSE: $(\hat{\theta}_n - \theta)^2$ and $(\tilde{\theta}_n - \theta)^2$
iv. Compute the MAE: $|\hat{\theta}_n - \theta|$ and $|\tilde{\theta}_n - \theta|$
v. Compute $\mathbb{I}\{|\hat{\theta}_n - \theta| < |\tilde{\theta}_n - \theta|\}$, which is an indicator that is one if the MLE estimator is closer to the true θ than the unbiased estimator

```
estimator_simulation <- function(n_value, theta_value)
{
    # Step (i)
    draws <- runif(n_value, theta_value, 2 * theta_value)

    # Step (ii)
    theta_tilde <- mean(draws) * 2 / 3
    theta_mle   <- max(draws)  * 1 / 2

    # Step (iii)
    MSE_tilde <- (theta_tilde - theta_value)^2
    MSE_mle   <- (theta_mle   - theta_value)^2

    # Step (iv)
    MAE_tilde <- abs(theta_tilde - theta_value)
    MAE_mle   <- abs(theta_mle   - theta_value)

    # Step (v)
    ind_value <- ifelse(MAE_mle < MAE_tilde, 1L, 0L)

    return(data.table(theta = theta_value,
                      n     = n_value,
                      MSE_tilde = MSE_tilde,
                      MSE_mle   = MSE_mle,
                      MAE_tilde = MAE_tilde,
                      MAE_mle   = MAE_mle,
                      ind_value = ind_value))
}
```

```
results_DT <- rbindlist(lapply(n_vec, function(n_value)
    {
        rbindlist(lapply(theta_vec, function(theta_value)
            {
                rbindlist(lapply(1:m, function(iteration_value)
                {
                    estimator_simulation(n_value, theta_value)[,Iteration := iteration_value]
                }))

            }))
    }))
```

```
summary_DT <- results_DT[, list(`MSE MLE` = mean(MSE_mle),
                                `MSE Tilde` = mean(MSE_tilde),
                                `MAE MLE` = mean(MAE_mle),
                                `MAE Tilde` = mean(MAE_tilde),
                                `Prob` = mean(ind_value)
                                ), by = c("theta", "n")]
```

From our simulations, we get the following tables for each θ value. Recall $\hat{\theta}_n$ is our MLE Estimator and $\tilde{\theta}_n$ is our unbiased estimator. MSE here implies the mean squared error and MAE implies the mean absolute deviation/error.

For the table columns, we define:

- `MSE MLE` $\iff E[(\hat{\theta}_n - \theta_0)^2]$
- `MSE Tilde` $\iff E[(\tilde{\theta}_n - \theta_0)^2]$
- `MAE MLE` $\iff E[|\hat{\theta}_n - \theta_0|]$
- `MAE Tilde` $\iff E[|\tilde{\theta}_n - \theta_0|]$
- `Prob` $\iff P\{|\hat{\theta}_n - \theta_0| < |\tilde{\theta}_n - \theta_0|\}$ which is the probability of $\hat{\theta}_n$ being closer to θ than $\tilde{\theta}_n$.

```
# theta = 0.8
kable(summary_DT[theta == 0.8, !c("theta"), with = FALSE],
      escape = FALSE, caption = "theta = 0.8", digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "bordered"),
                full_width = FALSE,
                latex_options = "hold_position")
```

Table 1: theta = 0.8

n	MSE MLE	MSE Tilde	MAE MLE	MAE Tilde	Prob
2	0.026	0.011	0.132	0.087	0.283
5	0.008	0.005	0.067	0.056	0.415
25	0.000	0.001	0.015	0.025	0.667
100	0.000	0.000	0.004	0.012	0.804

```
# theta = 1
kable(summary_DT[theta == 1, !c("theta"), with = FALSE],
      escape = FALSE, caption = "theta = 1", digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "bordered"),
                full_width = FALSE,
                latex_options = "hold_position")
```

```
# theta = 10
kable(summary_DT[theta == 10, !c("theta"), with = FALSE],
```

Table 2: theta = 1

n	MSE MLE	MSE Tilde	MAE MLE	MAE Tilde	Prob
2	0.041	0.018	0.166	0.110	0.276
5	0.012	0.008	0.085	0.070	0.416
25	0.001	0.001	0.019	0.031	0.658
100	0.000	0.000	0.005	0.015	0.812

```
    escape = FALSE, caption = "theta = 10", digits = 3) %>%
kable_styling(bootstrap_options = c("striped", "bordered"),
              full_width = FALSE,
              latex_options = "hold_position")
```

Table 3: theta = 10

n	MSE MLE	MSE Tilde	MAE MLE	MAE Tilde	Prob
2	4.210	1.876	1.670	1.120	0.287
5	1.154	0.725	0.821	0.687	0.418
25	0.073	0.148	0.194	0.307	0.652
100	0.005	0.037	0.049	0.154	0.807

From our results, we see that for smaller values of n, we would prefer to use the unbiased estimator and for larger values of n we would prefer to use the MLE estimator. Note that for a small n value, the biased of the MLE estimator is very large and the MLE estimator performs worse than the unbiased estimator. However, for larger n, we see that the MLE estimator has a much smaller MSE and MAE than those for the unbiased estimator.

Thus, for smaller n, we prefer the unbiased estimator and for large n, we prefer the biased estimator. Further, unbiasedness may not always be desirable as for large n our MLE estimate is much closer to the true θ value.

Theoretical Justification

We can justify the above results on theoretical grounds. From the CLT, the unbiased estimator has the property that

$$\sqrt{n}(\tilde{\theta}_n - \theta_0) \xrightarrow{d} N(0, \frac{\theta^2}{12})$$

As in the previous problem, we know the MLE has the property that it is n-consistent

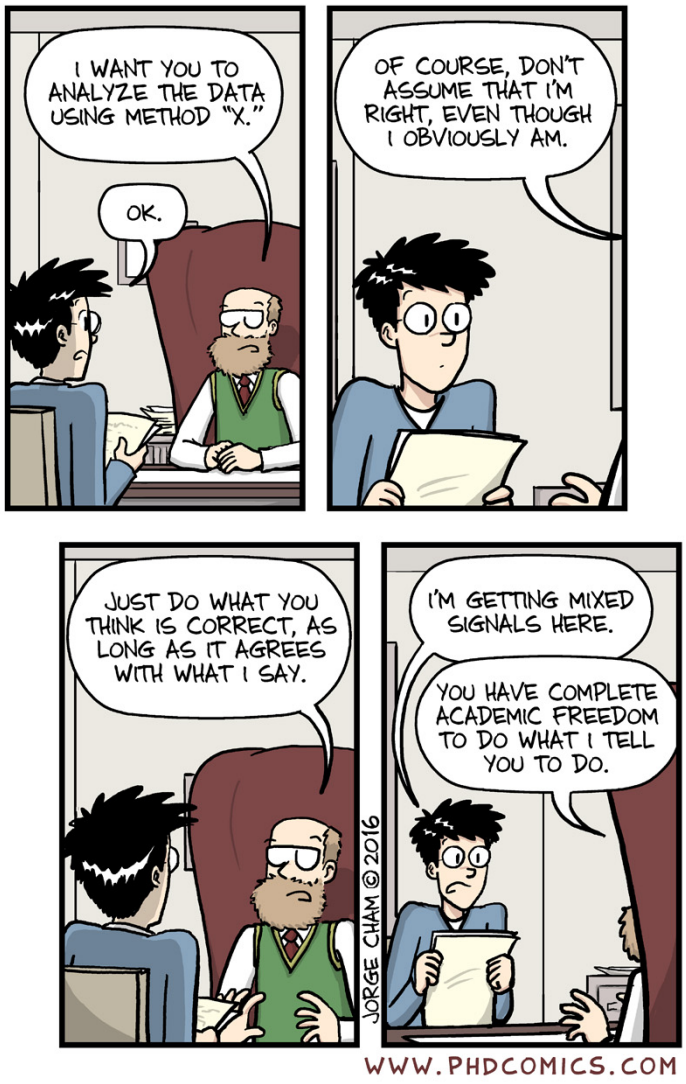
$$n(\hat{\theta}_n - \theta_0) = O_p(1)$$

Note that the unbiased estimator is only $\sqrt{n}$-consistent. Hence, we expect the MLE to converge to θ_0 at a faster rate than the unbiased estimator (or as $n \to \infty$ the MLE estimator gets more concentrated around θ_0), but in small samples it may have performance due to the MLE estimator's large bias.

Part II

Statistical Inference III - Methods

This section provides an overview of the commonly used econometric methods used in the first year courses. These concepts should provide background for the third quarter econometrics courses at Booth or the Economics department. The companion RMarkdown notebook provides an implementation of various methods for a Heterogeneous Treatment Effects estimation in a simulated RCT.



4 Panel Data Methods

A set of **panel data** is a set of observations $\{(X_{i,t}, Y_{i,t})\}_{(i,t)\in I}$ indexed by both i and t, where $X_{i,t}$ has k dimensions. A **balanced panel** means that there are N and T such that the index set $I = \{1,\dots,N\} \times \{1,\dots,T\}$. We will focus on the balanced panel case. For the three following sub-sections, we will consider the same statistical model:

$$Y_{it} = \alpha_i + X_{it}'\beta + U_{it},$$

which we call the **linear fixed effects** model.

4.1 Fixed Effects

For each individual i, let $\bar{z}_i$ be the within-individual average for $z \in \{Y, X, U\}$, i.e. $\bar{z}_i = T^{-1}\sum_{t=1}^{T} z_{it}$. Let $\dot{z}_{it}$ be the demeaned value, $\dot{z}_{it} = z_{it} - \bar{z}_i$. Then, the linear fixed effects model becomes

$$\dot{Y}_{it} = \dot{X}_{it}'\beta + \dot{U}_{it}.$$

By pooling data over i and t and then applying OLS, we propose the **fixed-effect estimator**

$$\hat{\beta}_{FE} = \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}'\right)^{-1}\left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\dot{X}_{it}\dot{Y}_{it}\right).$$

There is a numerically equivalent representation of FE. For each i and t, let D_{it} be an indicator vector of N dimensions such that D_{it} has one on its ith entry and zero everywhere else. Let $Z_{it} = (X_{it}', D_{it}')'$ which contains the covariates and the *fixed effect* D_{it}. Now we regress Y_{it} on Z_{it} with the pooled data, and obtain

$$\hat{\theta}_{FE} = \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}Z_{it}Z_{it}'\right)^{-1}\left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}Z_{it}Y_{it}\right).$$

The first k entries of $\hat{\theta}_{FE}$ will be numerically equivalent to $\hat{\beta}_{FE}$. For either method, make sure X does not include a constant, otherwise you will have multicollinearity.

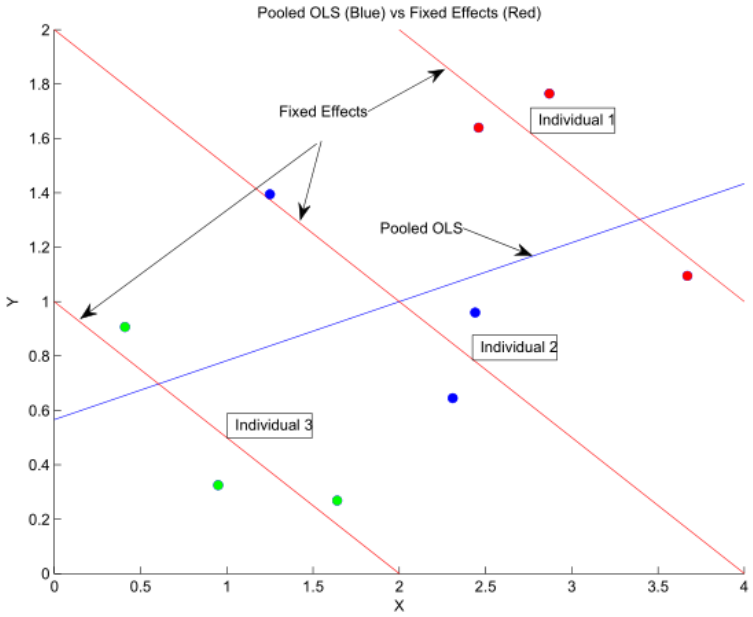
The statistical properties of the fixed-effect estimator will vary depending on the the assumptions you are willing to make.

Case 1: Assume (Y_{it}, X_{it}, U_{it}) is i.i.d. for each i and t, then we can treat $\hat{\theta}_{FE}$ as an OLS estimator and derive standard OLS properties under regularity conditions.

Case 2: Assume (Y_{it}, X_{it}, U_{it}) is independent across i, but is correlated within i. That is, we assume $E[U_{it}U_{js}] = 0$ for $i \neq j$, but we might have $E[U_{it}U_{is}] \neq 0$ for some t and s. Also, assume $N \to \infty$ and T is fixed. In this case, the standard OLS mean-independence condition $E[U_{it}|X_{it}] = 0$ is not sufficient for consistency. We need some form of "*strict exogeneity*". An example is:

[SE]: (strict exogeneity) $E[U_{it}|\alpha_i, X_{i1}, X_{i2}, \dots, X_{iT}] = 0$ for each i and t.

Figure 1: Fixed effects v.s. Pooled OLS



The consistency of the fixed-effect estimator can be established by

$$\begin{aligned}
\hat{\beta}_{FE} &= \left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}'\right)\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{Y}_{it}\right)\right) \\
&= \beta + \left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}'\right)\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{U}_{it}\right)\right) \\
&\xrightarrow{p} \beta + E\left[\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{X}_{it}'\right]^{-1} E\left[\frac{1}{T}\sum_{t=1}^{T}\dot{X}_{it}\dot{U}_{it}\right] \\
&= \beta.
\end{aligned}$$

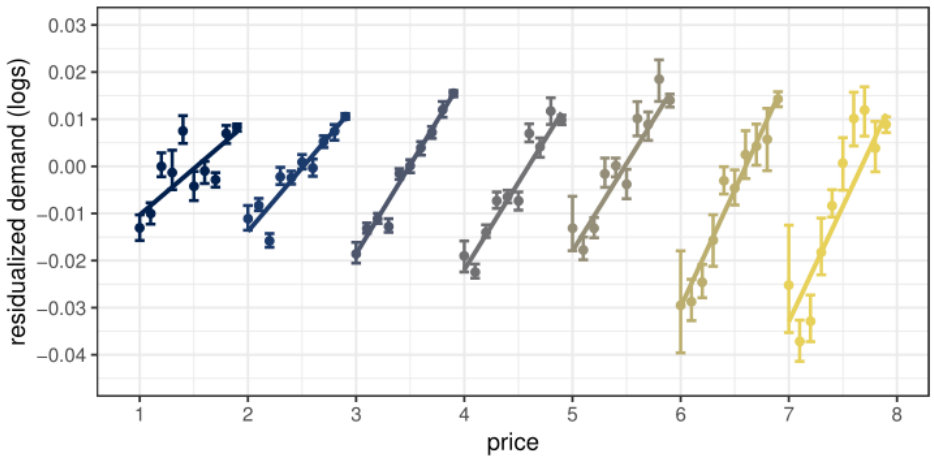
To justify the convergence in probability, we need to check conditions of WLLN and invertibility. The last equality is due to **[SE]**.

We can follow the similar procedure from before to obtain the asymptotic distribution of $\hat{\beta}_{FE}$. Note that there are other popular sets of assumptions in addition to our two cases.

Remark.

1. Figure 1 illustrates the intuition for the fixed effects and pooled OLS estimator. This figure also demonstrates a version of Simpson's Paradox.
2. Figure 2 from Strulov-Shlain (2019) demonstrates the same pattern appearing in left digit bias in consumer demand.

Figure 2: Left digit bias



4.2 First Differences

Another way to cancel out the individual effect is to take **first differences**. In other words, let $\tilde{z}_{i,t} = z_{i,t} - z_{i,t-1}$ for $z \in \{Y, X, U\}$. Then the linear fixed effects model is

$$\tilde{Y}_{it} = \tilde{X}_{it}'\beta + \tilde{U}_{it}.$$

Pooled OLS can be used now to estimate β, i.e.

$$\hat{\beta}_{FD} = \left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\tilde{X}_{it}\tilde{X}_{it}'\right)^{-1}\left(\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\tilde{X}_{it}\tilde{Y}_{it}\right),$$

where we assume data of $t = 0$ is also available for notational simplicity. Again, different sets of assumptions yields different statistical properties.

4.3 Fama-MacBeth Regression

The Fama-MacBeth estimator (FM) is especially popular in accounting and financial research. It can take various forms, but the idea is to first divide observations into G groups, then estimate the parameter in each subgroup, and take the average of the G estimators as the final estimator. A typical FM estimator in the context of the linear fixed effect model is given as follows:

For each i, first run OLS of $\{Y_{it}\}_{t=1}^{T}$ on $\{X_{it}\}_{t=1}^{T}$ and a constant. Ignore the constant's estimator and let the coefficients of X be $\hat{\beta}_i$. Then, the Fama-MacBeth estimator is given by

$$\hat{\beta}_{FM} = \frac{1}{N}\sum_{i=1}^{N}\hat{\beta}_i.$$

The Fama-MacBeth estimator was proposed in 1973 and the theory behind it was established relatively recently. It is conceptually simple and easy to implement. Surprisingly, it has very nice statistical proper-

ties. The intuition behind this is that we can treat each $\hat{\beta}_i$ as being drawn from some distribution. As long as they are approximately independent, their average will have nice properties.

5 Treatment Effects Overview

5.1 Theory

5.1.1 Neyman-Rubin Causal Model

The Neyman-Rubin Causal Model underlies most of the causal inference research in applied economics. The other approach is the Judea Pearl's causal graphs and direct acyclical graphs (DAGs) formulation of approaching the problem of causal inference. Note that the two approaches are not exclusive and often researchers can map their problems from one to another. We will focus on the Neyman-Rubin Causal Model for these set of notes.

Consider the following scenario. We want to determine the causal impact of math camp (or listening to Karthik's curated Spotify playlist) on the first year student's first quarter GPA. The population here is all of the incoming Booth PhD students in math camp this year. For each individual i, the no treatment case ($W_i = 0$) case would be that the individual does attend to math camp and the treatment case ($W_i = 1$) would be that the individual attends math camp. The outcome of interest is the first quarter students' GPA. There are two treatment levels and individuals are assigned to the treatment or no treatment case.

Then, causal effect (or treatment effect) of treatment W_i on individual i is the difference in the potential outcomes or,

$$\tau_i = Y_i(W_i = 1) - Y_i(W_i = 0).$$

This is a model of parallel worlds where the treatment effect, τ_i, is the difference between the two worlds. In our example, the individual treatment effect would be the difference in GPA if that individual attended math camp to not attending math camp. Note that we can never know the true treatment effect because we never observe the other potential outcome. An incoming Booth PhD student either was allowed to go to math camp or did not go, and we cannot see the other potential outcome for person i. This is known as the **fundamental problem of causal inference**. Lastly, we can generalize this to many treatment arms (e.g. $W_i \in \{0, 1, \ldots, k\}$).

Since we only have the **realized outcome** (or observed outcome) for each individual i,

$$Y_i^{obs} = \begin{cases} Y_i(W_i = 0) & \text{if } W_i = 0 \text{ (Did not attend math camp)} \\ Y_i(W_i = 1) & \text{if } W_i = 1 \text{ (Attended math camp)} \end{cases}$$

which is one of the two potential outcomes, and we can never directly estimate the individual causal effect or individual treatment effect of

$$Y_i(W_i = 1) - Y_i(W_i = 0).$$

Instead of estimating the individual treatment effects, we can estimate the **average treatment effect (ATE)** in the population

$$ATE = E[Y_i(W_i = 1) - Y_i(W_i = 0)].$$

In our example, the ATE is the *average difference* in first quarter students' GPA (*outcome variable*) between those who attended math camp to those who did not attend math camp (*across treatment groups*). To estimate the ATE, we can use the potential outcomes framework,

$$\tau = E[Y_i(W_i = 1) - Y_i(W_i = 0)] = E[Y_i^{obs}|W_i = 1] - E[Y_i^{obs}|W_i = 0].$$

A major concern with the causal estimates are **confounds**. Confounds are can be observable or unobservables that led to certain individuals to *select* on the treatment. These confounders can lead estimates to not be the average treatment effect (ATE). For example, PhD students who are better at math may decide to skip math camp entirely (or select out of the math camp treatment), so the average treatment effect will be lower than the true average treatment effect. If we observe student's math ability before math camp, then we can try to control for the math ability by proposing an instrumental (IV) that captures the effect. Alternatively, there could some unobservable characteristics that leads to selection on treatment. Controlling for these would require a stronger argument on how your model relates to the underlying data-generating process.

5.1.2 Randomization and Randomized Control Trials (RCT)

One way to try to avoid confounders is introducing randomization to the treatment assignment. Instead of allowing for self-selection, we can try to impose the treatment arms on the treatment groups. In our example, we would randomly assign each person in the incoming class to be treated. Then, assuming compliance to the treatment, we can just estimate $E[Y_i^{obs}|W_i = 1] - E[Y_i^{obs}|W_i = 0]$, or that the difference in means of the outcome variable of interest to attain the ATE. If there are multiple treatment arms, we can generalize the procedure of the differences in the means of the outcome variables of the treatment groups to the non-treatment group.

However, for randomization to give us an estimate the average treatment effect, we need the following conditions to hold.

1. **Unconfoundedness**: The potential outcomes $Y_1(0)$ and $Y_i(1)$ are statistically independent of the treatment W_i,
$$(Y_i(0), Y_1(1)) \perp W_i$$
2. **Overlap**: The probability of receiving the treatment is between 0 and 1,
$$0 < Pr\{W_i = 1\} < 1$$
3. **Stable unit treatment value assumption (SUTVA)**: The treatment assignment for individual i does not affect the treatment assignment for individual $k \neq i$

Remark.

- Unconfoundedness and overlap are *directly* under the RCT designer's control. If the treatment is randomized correctly for some probability of treatment between 0 and 1, then will be satisfied.
- SUTVA rules out social interactions or economic equilibrium effects.

Proposition 2. *If these three conditions are satisfied, then we have*

$$\tau = E[Y_i^{obs}|W_i = 1] - E[Y_i^{obs}|W_i = 0] = ATE.$$

Proof. Let us denote Y_i as the shorthand for Y_i^{obs}. Then,

$$\begin{aligned} E[Y_i(1) - Y_i(0)] &= E[Y_i(1)] - E[Y_i(0)] \\ &= E[Y_i(1)|W_i = 1] - E[Y_i(0)|W_i = 0] \\ &= E[Y_i|W_i = 1] - E[Y_i|W_i = 0] \end{aligned}$$

where the first line follows from the the linearity of the expectation operator. The second line follows from the unconfoundedness assumption. This assumption is key and allows us to see how we can use randomization to infer the ATE. In our recurrent math camp example, we expect $E[Y_i(0)|W_i = 0] > E[Y_i(0)]$ without randomization (and compliance to the treatment), and this occurs from **selection bias**. The last line states that a conditional on assignment treatment w, the observed outcomes is the same as the potential outcome $Y_i(w)$ that corresponds to this treatment. □

Remark. To recap the intuition for an RCT, we note that if the RCT with two treatment arms was ran correctly then:

- Random assignment yields *two copies of the same population* of individuals
- The *only difference* between the two populations is the treatment assignment or *experimental manipulation*
- Thus, the difference in outcomes between the two outcomes is *caused* by the treatment

We can check differences in the population of individuals in the treatment arms by examining **covariate balance**, which can involve comparing histograms of the pre-treatment variables or more formal statistical tests (i.e. an unpaired t-test or a KS-test).

We can also use standard regression analysis to estimate the average treatment effect. Let $W_i \in \{0, 1\}$ be a dummy variable indicating treatment status. The regression model to estimate the ATE, τ, is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK} + \tau W_i + \epsilon_i$$

where $X_{i1,} \ldots, X_{iK}$ are the **pre-treatment variables** or pre-treatment covariates. In our example, these could be whether an incoming student has a master's degree or not, whether she has worked as an RA before or not, etc.

Question 1. *Do we need to control for pre-treatment variables $X_{i1,} \ldots, X_{iK}$ to estimate the average treatment effect τ?*

Proof. Let W_i be randomly assigned and all covariates $X_1, \ldots, X_{iK}$ be independent. Consider the regression

$$Y_i = \beta_0 + \tau W_i + \tilde{\epsilon}_i$$

where the error term implicitly includes the omitted covariates. Let us define $\mu_k = E[X_{ik}]$ and $\delta =$

$\sum_{k=1}^{K} \beta_k \mu_k$. Then, we can write the error term as

$$\begin{aligned}\tilde{\epsilon}_i &= \beta_1 X_{i1} + \cdots + \beta_k X_{iK} + \epsilon_i \\ &= \delta + \sum_{k=1}^{K} \beta_k (X_{ik} - \mu_k) + \epsilon_i \\ &= \delta + \eta_i.\end{aligned}$$

From the unconfoundedness assumption, we have that

$$E[X_{ik} - \mu_k | W_i] = E[X_{ik}|W_i] - \mu_k = E[X_{ik}] - \mu_k = 0$$

so then

$$\begin{aligned}E[\eta_i|W_i] &= E[\sum_{k=1}^{K} \beta_k (X_{ik} - \mu_k) + \epsilon_i | W_i] \\ &= \sum_{k=1}^{K} \beta_k E[X_{ik} - \mu_k | W_i] + E[\epsilon_i | W_i] \\ &= 0.\end{aligned}$$

Then, the original regression in the form

$$Y_i = (\beta_0 + \delta) + \tau W_i + \eta_i$$

and the intercept will be $\beta_0 + \delta$. Since $E[\eta_i|W_i] = 0$ the regression will yield an unbiased estimate of the ATE. Thus, we do not need to control for $X_1, \ldots, X_{iK}$ to estimate the ATE. □

Remark.

- In applied research, the observed X_k's are usually included in the regression. Inclusion of the observed covariate will reduce the error's variance and the estimated coefficients will then be more precise.
- The regression's estimate will not yield a causal interpretation unless $E[\epsilon|X] = 0$. If the RCT is run correctly and assuming SUTVA, then $E[\epsilon|X] = 0$ should hold.

5.1.3 Observational Studies

We saw that if treatment is randomly assigned, then we can recover an unbiased estimate of the CATE. But in observational studies, where the researcher is not given the agency to introduce randomization in population, can we recover the causal estimate? It turns out we can, but we generally need stronger assumptions.

Consider the binary treatment $W_i = \{0, 1\}$ scenario. Even when W_i is not fully randomly assigned, it may be random conditional on the observed pre-treatment variables $X_{i1}, \ldots, X_{iK}$. This leads to the general unconfoundedness assumption which is weaker than in the RCT case.

1. **Unconfoundedness** (general): The potential outcomes $Y_1(0)$ and $Y_i(1)$ are statistically independent of the treatment W_i conditional on $X_{i1}, \ldots, X_{iK}$

$$\{(Y_i(0), Y_1(1)) \perp W_i\} | X_{i1}, \ldots, X_{iK}$$

In colloquial terms, this means that the treatment W_i is *as good as random.* If the general version of unconfoundedness assumption is satisfied, then we can recover the ATE

$$ATE = E[Y_i^{obs} | W_i = 1] - E[Y_i^{obs} | W_i = 0].$$

Note that in regression formulation for estimating the ATE, we now must control for the pre-treatment covariates $X_{i1}, \ldots, X_{iK}$ in order to recover the ATE. The regression function then becomes,

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_k X_{iK} + \tau W_i + \epsilon_i.$$

The regression model can also be extended to multi-level treatments or a continuous treatment variable W_i.

Revisiting our example for determining the causal impact of math camp, we may believe after controlling whether an incoming PhD student has gotten a masters, has been an RA before, her GRE math scores, and her undergraduate GPA may be enough for general unconfoundedness to hold.

Remark. Making this claim that general unconfoundedness holds will depend on how convincing the story you tell is. In the above example, if there is some unobservable characteristic that leads to selection bias, then the unconfoundedness assumption will not hold. While running an RCT gives you unconfoundedness and overlap "for free", the convincing story instead needs be on whether the experiment was run correctly and whether SUTVA hold in your scenario.

5.1.4 Conditional Average Treatment Effect (CATE)

We can extend the average treatment effects framework to get the **conditional average treatment effect (CATE)**, which is just the average treatment effect conditional on the some covariates. In our example, we may want to estimate the average treatment effect of math camp for students who have done a master's program before. Then, the estimate of interest is the conditional average treatment effect where we are conditioning on the whether an individual has completed a master's program before starting the PhD.

The CATE is represented as

$$\begin{aligned} CATE(x_i) &= \tau(x_i) \\ &= E[Y_i(W_i = 1) - Y_i(W_i = 0) | X_i = x_i] \\ &= E[Y_i(W_i = 1) | X_i = x_i] - E[Y_i(W_i = 0) | X_i = x_i]. \end{aligned}$$

The CATE can be different across customers with varying characteristics and thus is a **heterogeneous treatment effect (HTE)**. If we assume a linear model for the CATE, then we have the proposed regression

model

$$Y_i = \beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} + \delta_0 W_i + \sum_{k=1}^{p} \delta_k (x_{ik} W_i) + \epsilon_i$$

and the interaction terms are $x_{ik}W_i$. The regression model will let us estimate the conditional expectation function

$$E[Y_i|\boldsymbol{x}_i, W_i] = \beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} + \delta_0 W_i + \sum_{k=1}^{p} \delta_k (x_{ik} W_i)$$

Our goal is to estimate the CATE,

$$\begin{aligned}\tau_i &= E[Y_i(W_i = 1) - Y_i(W_i = 0)|\boldsymbol{x}_i] \\ &= E[Y_i|\boldsymbol{x}_i, W_i = 1] - E[Y_i|\boldsymbol{x}_i, W_i = 0].\end{aligned}$$

We can use the regression function to predict the CATE,

$$\begin{aligned}\tau_i &= E[Y_i|\boldsymbol{x}_i, W_i = 1] - E[Y_i|\boldsymbol{x}_i, W_i = 0] \\ &= \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} + \delta_0 + \sum_{k=1}^{p} \delta_k x_{ik})\right) - \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik}\right) \\ &= \delta_0 + \sum_{k=1}^{p} \delta_k x_{ik}\end{aligned}$$

and we will attain consistent estimates of the parameters δ_k when the three assumptions from before are satisfied (unconfoundedness, overlap, and SUTVA).

Remark. To attain the ATE from the CATE, you integrate out the $\boldsymbol{x}_i$. In other words, you evaluate the CATE with $\boldsymbol{x}_i = \bar{\boldsymbol{x}}_i$ to recover the ATE. This also means that in the last line of the above equation, δ_0 is not the ATE.

5.2 An Applied Toolbox

Instrumental variables (IV), RCTs, DiD, RDD, and matching are commonly used tools by applied econometricians. However, all these methods all depend on the assumptions underpinning the models. Ultimately, to make a convincing argument that your method captures the causal effect, one needs to tell a *credible story* about how the model fits the *underlying* **data-generating process (DGP)**. If one's audience is not convinced by the story, then no matter what econometric technique is presented, the causal estimate would be viewed in suspect.

Even for the "gold standard" of methods, the RCT, a story needs to be told to justify the RCT was run correctly. Otherwise, there could be selection on unobservables or confounders that plague the RCT estimates. Thus, regardless of the econometric method, a credible story of the relation of how the technique relates to the DGP is required.

A separate question is the **external validity** of the estimates. The question asks whether the estimates *generalize* to a larger population or only adequately describe the people in one's data sample. Ultimately, for a **normative** economics perspective, externality validity is required for decision making, such as for policy

or business decisions. If the researcher is focused on a **positive** economics perspective, then treatment effect on the estimated population is itself interesting enough.

Other than IV and RCTs, commonly used techniques by applied econometricians are matching estimators, differences-in-differences, and regression discontinuity designs. We will cover the latter three in this section.

5.2.1 Matching

In essence, the matching estimator examines people in a treatment group and finds the closest person or people in the other treatment group. Then, taking the differences in the outcome variables will yield the treatment effect. When one person is matched to many in another treatment group, there can be **matching with replacement** or **matching without replacement**. If a model of the **propensity score** (a model of the probability of being treated conditional on observed covariates) exists, then matching can be also done on the propensity score estimates. Matching tries to approximate the individual level treatment effect by finding someone in the other group that is similar to the person of interest.

Consider the case with binary treatment $W_i \in \{0, 1\}$. We first denote that person i matches with M_i to be the matched people in the other treatment arm. Often M_i is chosen by the researcher and the number of people matched with is constant across i. Then the matching estimator the other potential outcomes for individual i by constructed by averaging the realized outcomes across M_i

$$\tau_i = \begin{cases} Y_i^{obs} - \frac{1}{|M_i|} \sum_{l \in M_i} Y_l^{obs} & \text{if } W_i = 1 \\ \frac{1}{|M_i|} \sum_{l \in M_i} Y_l^{obs} - Y_i^{obs} & \text{if } W_i = 0 \end{cases}.$$

To find $m \in M_i$ for each individual i, researchers often choose the individuals in the other treatment arm that have similar $X_{m1}, \ldots, X_{mK}$ to that individual i. Alternatively if a propensity score model, or $Pr(W_i = 1|X)$, has been estimated, then the researcher may choose the $m \in M_i$ who have similar propensity scores to individual i.

Matching with replacement means that one person can be matched to many people. When $|M_i| > 1$ this will be the case by construction. Matching without replacement means that each person can only be matched a unique person in the other treatment arm. In this case, the order of the matching assignment will play a factor.

Exercise 7. In the matching with replacement scenario, when $|M_i|$ gets very large will the treatment effect estimates have high variance or high bias? How would you choose the size of $|M_i|$?

Remark.

- Matching with covariates is often considered a "pure" method in that it does not require knowledge of the outcome variable to form matches. Thus, matches can be formed before the experiment is run if the characteristics of the participants are a priori known.
- Applied econometricians do not like matching outside of RCT settings due to possible selection on unobservables.

5.2.2 Differences-in-Differences (DiD)

Differences-in-differences (DiD) uses panel data and two groups to estimate the causal effect. Consider control group C and treatment group T as well as pre-treatment time period 0 and post-treatment time period 1. To give an example, let's modify our math camp example such that before math camp there is a math ability test and another one after math camp ends. Then the two groups would be the people who attended or did not attended math camp, the outcome of interest is the math ability test score, and the time periods would be before and after math camp. The goal is to estimate the causal effect of the treatment of math camp on the math ability test scores.

The DiD causal effect estimator is then

$$\hat{\beta}_{DiD} = (\bar{y}_{T1} - \bar{y}_{T0}) - (\bar{y}_{C1} - \bar{y}_{C0})$$

where $\bar{y}_{g,t}$ is the sample average in the outcome variable for group g at time period t. Taking expected values, we attain

$$\begin{aligned} E[\hat{\beta}_{DiD}] &= (E[\bar{y}_{T1}] - E[\bar{y}_{T0}]) - (E[\bar{y}_{C1}] - E[\bar{y}_{C0}]) \\ &= (\beta_{TE} + \delta_{T1} + \alpha_T - \alpha_T) - (\delta_{C1} + \alpha_C - \alpha_C) \\ &= \beta_{TE} + (\delta_{T1} - \delta_{C1}) \end{aligned}$$

where β_{TE} is the treatment effect, α_T, α_C are the baseline averages in groups T and C, and δ_{C1}, δ_{T1} are the change in the average outcomes for group C and T from time 0 to time 1 respectively.

If we consider a **common trends assumption**, $\delta_{C1} = \delta_{T1}$, which says the time trends in both groups are the same in absence of treatment, then

$$E[\hat{\beta}_{DiD}] = \beta_{TE}$$

and the DiD estimator is an unbiased estimate the to treatment effect. We can also formulate the DiD in a regression model,

$$y_{it} = \alpha + \alpha_T T_{it} + \delta D_{it} + \beta T_{it} D_{it} + \epsilon_{it}$$

where $T_{it} = 1$ if individual i is in group T (treatment group dummy variable) and $D_{it} = 1$ if time $t = 1$ (post-treatment dummy).

Remark. The model implies the following:

1. $E[y_{it}|i \in C, t = 0] = \alpha$
2. $E[y_{it}|i \in C, t = 1] = \alpha + \delta$
3. $E[y_{it}|i \in T, t = 0] = \alpha + \alpha_T$
4. $E[y_{it}|i \in T, t = 1] = \alpha + \alpha_T + \delta + \beta$

Then, from the four implications, we have that

$$\beta = (E[y_{it}|i \in T, t = 1] - E[y_{it}|i \in T, t = 0]) - (E[y_{it}|i \in C, t = 1] - E[y_{it}|i \in C, t = 0]) = \beta_{TE}.$$

Thus, from the model we have $\hat{\beta}_{OLS} = \hat{\beta}_{DiD}$.

We can extend the standard DiD model in a few ways. First, we can control for exogenous covariates $\boldsymbol{x}_{it}$, which assumes *common trends after controlling for* $\boldsymbol{x}_{it}$, and yields the regression equation,

$$y_{it} = \alpha + \alpha_T T_{it} + \delta D_{it} + \beta T_{it} D_{it} + \boldsymbol{x}_{it}'\gamma + \epsilon_{it}.$$

There can also be M total treatment and control groups. The DiD setup assumes that the treatment effect is homogeneous across groups and common trends are assumed after controlling for $\boldsymbol{x}_{it}$. This yields the regression equation,

$$y_{it} = \left(\sum_{l=1}^{M} \alpha_j d_{i\in l}\right) + \delta D_{it} + \beta T_{it} D_{it} + \boldsymbol{x}_{it}'\gamma + \epsilon_{it}$$

where $d_{i\in l}$ is dummy variable that is 1 if individual i is in group l.

Lastly, we can extend the model so there could be treatment at different time periods $(0, 1, \ldots, T)$ for various cross-section groups (M total groups) with homogeneous treatment effects across groups and common trends are controlling for $\boldsymbol{x}_{it}$. This yields the regression equation,

$$y_{it} = \left(\sum_{l=1}^{M} \alpha_j d_{i\in l}\right) + \left(\sum_{s=1}^{T} \delta_s D_{is}\right) + \beta \tilde{T}_{it} + \boldsymbol{x}_{it}'\gamma + \epsilon_{it}$$

where $d_{i\in l}$ is dummy variable that is 1 if individual i is in group l, D_{is} is a dummy variable that is 1 if treatment $t = s$, and $\tilde{T}_{it}$ is a dummy variable that is 1 is individual i is treated at time t.

Remark. Note that the last extension to DiD is the same as a fixed-effects panel data model where α_j is the time invariant heterogeneity (or the individual fixed effect) for an individual j and δ_s are cross-section invariant heterogeneity (or the time fixed effect).

5.2.3 Regression Discontinuity Design (RDD)

The **regression discontinuity design (RDD)** is a quasi-experimental method that recovers a causal effect from interventions by examining the individuals around the threshold induced by an intervention. Then, by examining the observations close to the threshold on both sides, the average treatment effect can be evaluated. RDD can be used on observational data or in scenario where randomization in not feasible. For example, a threshold altered by a policy change can be used for an RDD design (i.e. a change to a bracket of the tax law)

The RDD has a non-parametric and a parametric estimation strategy. The **parametric RDD** imposes a rigid functional form (usually a polynomial) on the estimation equation. One example can be using fifth order polynomials,

$$Y = \beta_0 + \beta_1 x_i + \beta_2 c_i + \beta_3 c_i^2 + \beta_4 c_i^3 + \beta_5 c_i^4 + \beta_6 c_i^5 + \epsilon.$$

where Y is the outcome variable, c_i is the treatment variable (or running variable/assignment variable), $\tilde{c}$ is the cut-off, and

$$x_i = \begin{cases} 1 & \text{if } c_i \geq \tilde{c} \\ 0 & \text{if } c_i < \tilde{c} \end{cases}.$$

Non-parametric RDD often uses local linear regression for estimation. Local linear regression is used

because it has low bias and well-known convergence properties. A common estimation equation is

$$Y = \beta_0 + \tau D + \beta_1(X - \tilde{c}) + \beta_2 D(X - \tilde{c}) + \epsilon$$

where $\tilde{c}$ is the the treatment cutoff and $D = \mathbf{1}\{X \geq \tilde{c}\}$. For given bandwidth h, we have that $c-h \leq X \leq c+h$ and usually a rectangular kernel is used. Note that different slopes and intercepts are fit on the data before and after the threshold.

Remark. Non-parametric RDD is one of the few popular applications of non-parametric regression in the applied literature.

Remark. The assumptions for RDD to produce a valid estimate are the following.

1. All relevant variables, including treatment and outcome variables, are continuous up to where the treatment and outcome discontinuities occur.
2. If the treatment assignment is *as good as random* at the threshold of treatment, then those who just are past the threshold are close to those under the threshold.
3. Treatment being as good as random also implies that individuals near the threshold cannot select into treatment. In the tax bracket example, individuals near threshold would disallowed from purposefully reducing income to be in the lower tax bracket. If not, this would lead to selection bias.

6 Machine Learning Introduction and Example

See the slides and companion video for a machine learning introduction with the focus on causal estimation. The first half of the slides focus on using machine learning techniques for estimating heterogeneous treatment effects in an RCT setting. The second half of slides takes a more normative perspective and examines ways to evaluate different targeting policies as well as how to construct the optimal targeting policy from an randomized control trial.

In the example, we will examine heterogeneous treatment effects (HTE) estimation using various machine learning methods for a simulated RCT. In the RCT framework, HTE estimation becomes a prediction problem – we just need to predict the unobserved potential outcome to estimate the HTE. This fact occurs because if the RCT was performed correctly, then we would satisfied the unconfoundedness and the overlap conditions. If we further that assume SUTVA holds, then we get a pure prediction problem. Please see the RMarkdown notebook for the example.

Example: Heterogeneous Treatment Effects

Booth Math Camp (Autumn 2021)

Walter W. Zhang

01 July 2021

This RMarkdown Notebook walks through heterogeneous treatment effects estimation for simulated randomized control trial (RCT). The notebook also introduces different machine learning estimators and how to implement them in R. To run the notebook, you may need to install RTools on Windows or Xcode on a Mac if you have not done so already.

Contents


```
# Load packages
require(data.table)
require(ggplot2)
require(grf)
require(gamlr)
require(knitr)
require(parallel)
require(DiagrammeR)
require(ranger)
require(rpart)

# Number of cores to use
ncores <- detectCores() - 1L
```

Data Simulation

We simulate a field experiment that includes the following variables:

Outcome:

- `spend` is observed dollar spending.

Features:

- `recency` is the customer recency status (in months), ranging from 1 to 18.
- `email` is a dummy variable that indicates if a customer signed up to company's email list host.

Randomized Treatment:

- `target` is the treatment indicator, a dummy variable indicating if a customer was targeted with a catalog. Note that this assignment is random.

We model the purchase probability, p, as follows:

- Targeted consumers:
  - For `recency` between 1 and 6, $p = 0$.
  - For `recency` between 7 and 12, p increases in `recency` and takes the value $p = 0.03(recency - 6)$.
  - For `recency` greater than 12, $p = 0.18$.
- Not targeted consumers:
  - If `email` is true, then p is 1.25 times the baseline purchase probability.
  - If `email` is false, then p is twice the the baseline purchase probability.

Spending conditional on a purchase is uniformly distributed between 80 and 120 (Mean of 100). - Expected spending is then $100p$.

The treatment effect is non-linear in `recency`. The treatment effect is larger for customers who are not a `email` and have `recency` 7 or higher.

```
set.seed(1234L)

n_obs  <- 100000        # Training Observations
n_pred <- 100000        # Prediction Observations
n <- n_obs + n_pred

customer_DT <- data.table(target  = rbinom(n, 1, 0.5),
                          recency = sample(1:18, n, replace = TRUE),
                          email   = rbinom(n, 1, 1/3))

# Define the purchase probability p
customer_DT[recency <= 6,                 p := 0.0]
customer_DT[recency >  6 & recency <= 12, p := 0.03*(recency - 6)]
customer_DT[recency > 12,                 p := 0.03*6]

customer_DT[email == 1 & target == 1,     p := 1.25*p]
customer_DT[email == 0 & target == 1,     p := 2*p]

# Simulate spending data
customer_DT[, purchase   := runif(n) <= p]
customer_DT[, cond_spend := sample(80:120, n, replace = TRUE)]
customer_DT[, spend      := purchase*cond_spend]

training_DT <- customer_DT[1:n_obs]
pred_DT     <- customer_DT[(n_obs+1):n]
```

Heterogeneous Treatment Effect Estimation

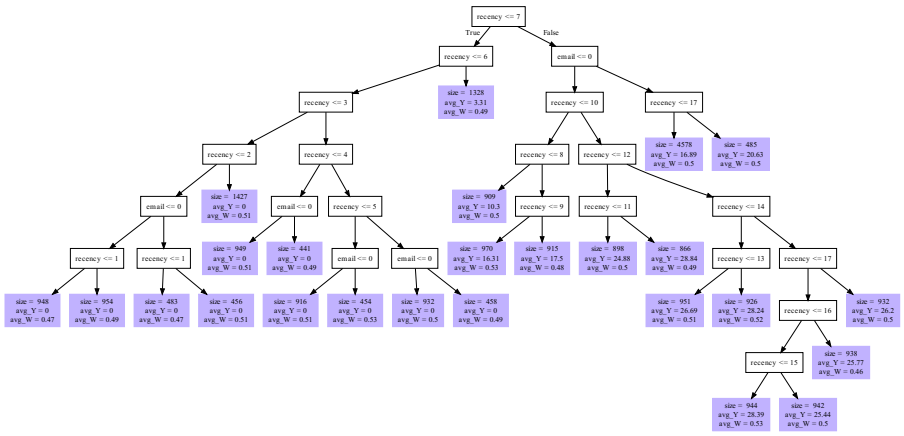
Causal Forest

We estimate the causal forest.

```
X_mat <- as.matrix(training_DT[, .(recency, email)])
Y_vec <- training_DT$spend
W_vec <- training_DT$target

CF_fit <- causal_forest(X = X_mat,
                        Y = Y_vec,
                        W = W_vec,
                        num.trees   = 1000L,
                        num.threads = ncores,
                        seed        = 5678)
```

Plot the first tree in the forest.

```
plot(get_tree(CF_fit, 1))
```



\begin{center}

Predict spending in the prediction sample.

```
pred_DT[, TE_CF := predict(CF_fit, as.matrix(pred_DT[, .(recency, email)]), num.threads = ncores)]
```

Examine table with the true, observed, and predicted CATE (τ) for all values of `email` and `recency`.

```
summary_DT <- pred_DT[, list(tau         = 100*(mean(p[target==1]) - mean(p[target==0])),
                             tau_obs     = mean(spend[target==1])  - mean(spend[target==0]),
                             tau_pred_CF = mean(TE_CF)),
                      keyby = .(email, recency)]
```

```
# Email == 0
kable(summary_DT[email == 0], digits = 2)
```

email	recency	tau	tau_obs	tau_pred_CF
0	1	0	0.00	0.46
0	2	0	0.00	0.46
0	3	0	0.00	0.46
0	4	0	0.00	0.46
0	5	0	0.00	0.46
0	6	0	0.00	0.46
0	7	3	3.06	2.31
0	8	6	5.50	5.79
0	9	9	10.12	11.41
0	10	12	12.40	13.77
0	11	15	15.85	15.38
0	12	18	16.04	16.06
0	13	18	21.68	17.53
0	14	18	18.20	18.14
0	15	18	18.25	17.62
0	16	18	18.15	17.24
0	17	18	17.93	15.75
0	18	18	15.96	16.93

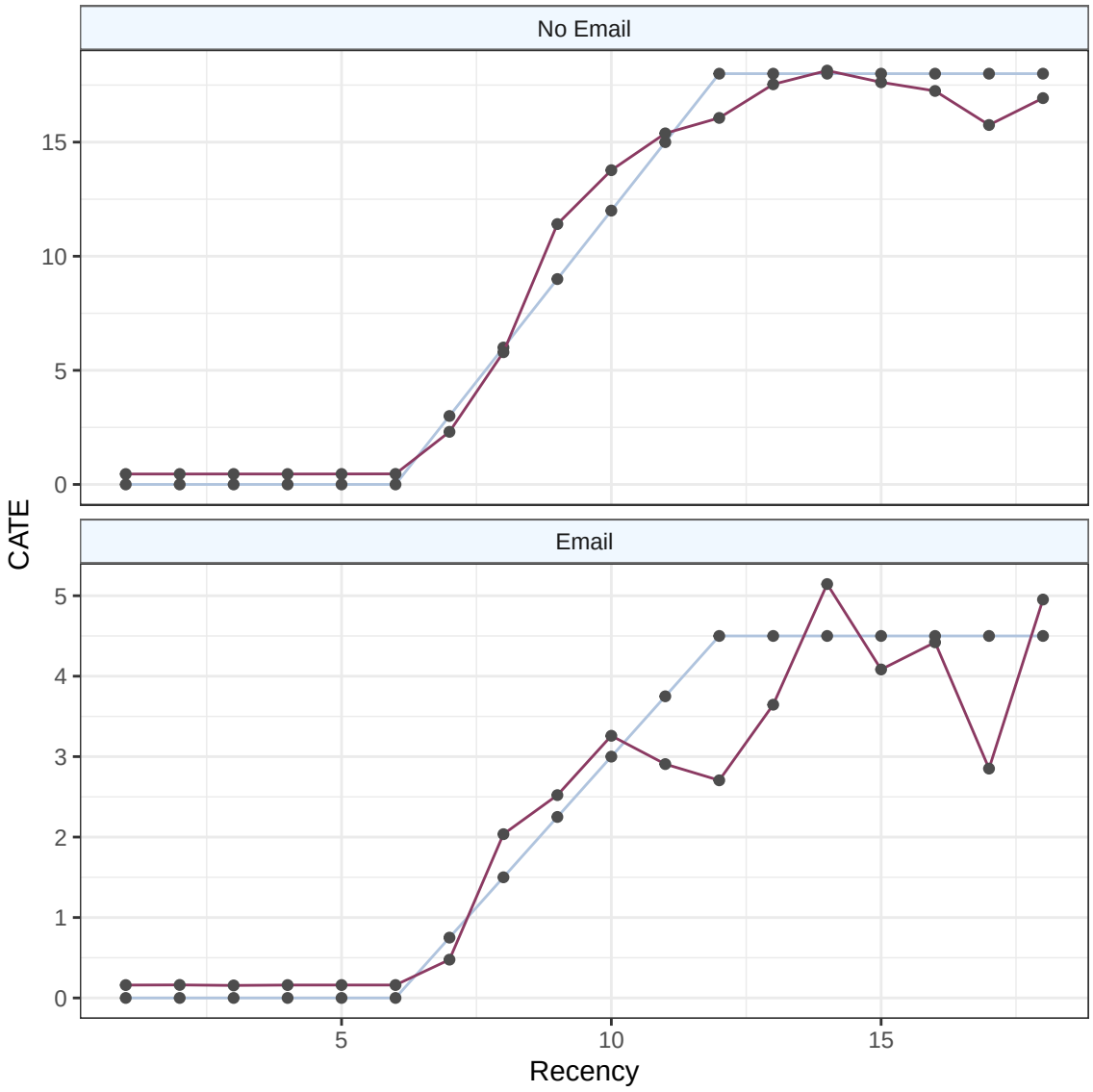
```
# Email == 1
kable(summary_DT[email == 1], digits = 2)
```

email	recency	tau	tau_obs	tau_pred_CF
1	1	0.00	0.00	0.16
1	2	0.00	0.00	0.16
1	3	0.00	0.00	0.16
1	4	0.00	0.00	0.16
1	5	0.00	0.00	0.16
1	6	0.00	0.00	0.16
1	7	0.75	0.24	0.48
1	8	1.50	-0.66	2.04
1	9	2.25	0.15	2.52
1	10	3.00	1.81	3.26
1	11	3.75	1.71	2.91
1	12	4.50	4.27	2.71
1	13	4.50	3.86	3.65
1	14	4.50	2.13	5.15
1	15	4.50	2.99	4.08
1	16	4.50	3.74	4.42
1	17	4.50	1.18	2.85
1	18	4.50	1.32	4.95

Plot the true and predicted CATE (τ) for all values of `email` and `recency`.

```
graph_DT <- copy(summary_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
```

```
  geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
  geom_line(aes(x = recency, y = tau_pred_CF),
            color = "hotpink4", size = 0.5) +
  geom_point(aes(x = recency, y = tau_pred_CF),
             color = "gray30", fill = "hotpink4", size = 1.5) +
  ylab("CATE") + xlab("Recency") +
  facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
  theme_bw() +
  theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```



OLS

```
OLS_DT <- training_DT[, .(spend, recency, email, target)]

fit_OLS <- lm(spend ~ . + .*target + email:recency*target,
              data = OLS_DT)

pred_DT[, spend_OLS := predict(fit_OLS, pred_DT)]

summary_OLS_DT <- pred_DT[, .(tau_pred_OLS = mean(spend_OLS[target==1])
                                           - mean(spend_OLS[target==0])),
                            keyby = .(email, recency)]
summary_OLS_DT <- merge(summary_OLS_DT, summary_DT[, .(email, recency, tau)],
                        by = c("email", "recency"))
```

```
# Email == 0
kable(summary_OLS_DT[email == 0], digits = 2)
```

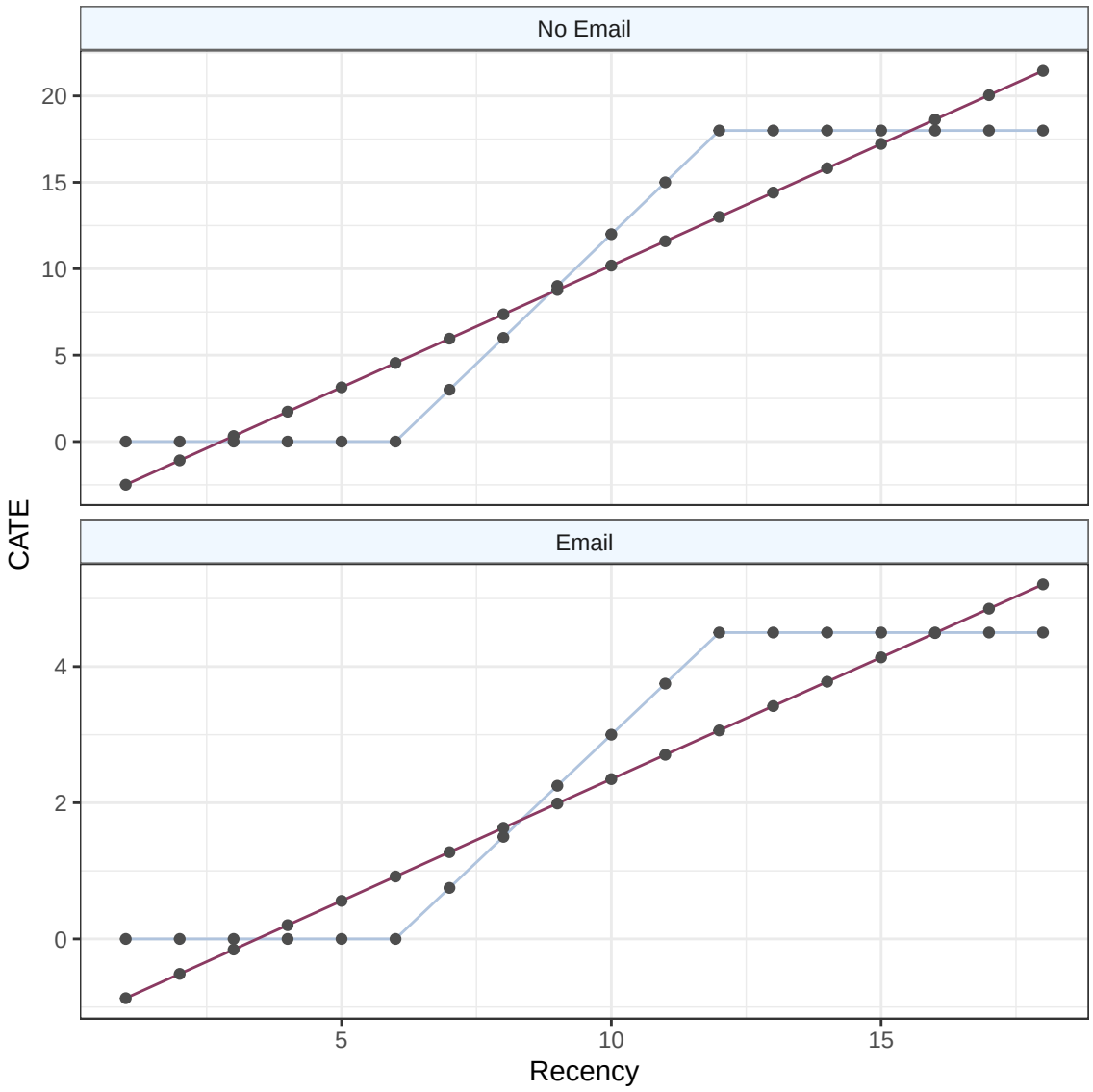
email	recency	tau_pred_OLS	tau
0	1	-2.50	0
0	2	-1.09	0
0	3	0.32	0
0	4	1.73	0
0	5	3.14	0
0	6	4.55	0
0	7	5.96	3
0	8	7.36	6
0	9	8.77	9
0	10	10.18	12
0	11	11.59	15
0	12	13.00	18
0	13	14.41	18
0	14	15.82	18
0	15	17.22	18
0	16	18.63	18
0	17	20.04	18
0	18	21.45	18

```
# Email == 1
kable(summary_OLS_DT[email == 1], digits = 2)
```

email	recency	tau_pred_OLS	tau
1	1	-0.87	0.00
1	2	-0.51	0.00
1	3	-0.16	0.00
1	4	0.20	0.00
1	5	0.56	0.00
1	6	0.92	0.00
1	7	1.27	0.75
1	8	1.63	1.50
1	9	1.99	2.25
1	10	2.35	3.00

email	recency	tau_pred_OLS	tau
1	11	2.70	3.75
1	12	3.06	4.50
1	13	3.42	4.50
1	14	3.78	4.50
1	15	4.14	4.50
1	16	4.49	4.50
1	17	4.85	4.50
1	18	5.21	4.50

```
graph_DT <- copy(summary_OLS_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
   geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
   geom_line(aes(x = recency, y = tau_pred_OLS),
             color = "hotpink4", size = 0.5) +
   geom_point(aes(x = recency, y = tau_pred_OLS),
              color = "gray30", fill = "hotpink4", size = 1.5) +
   ylab("CATE") + xlab("Recency") +
   facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
   theme_bw() +
   theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```




```
summary(fit_OLS)
```

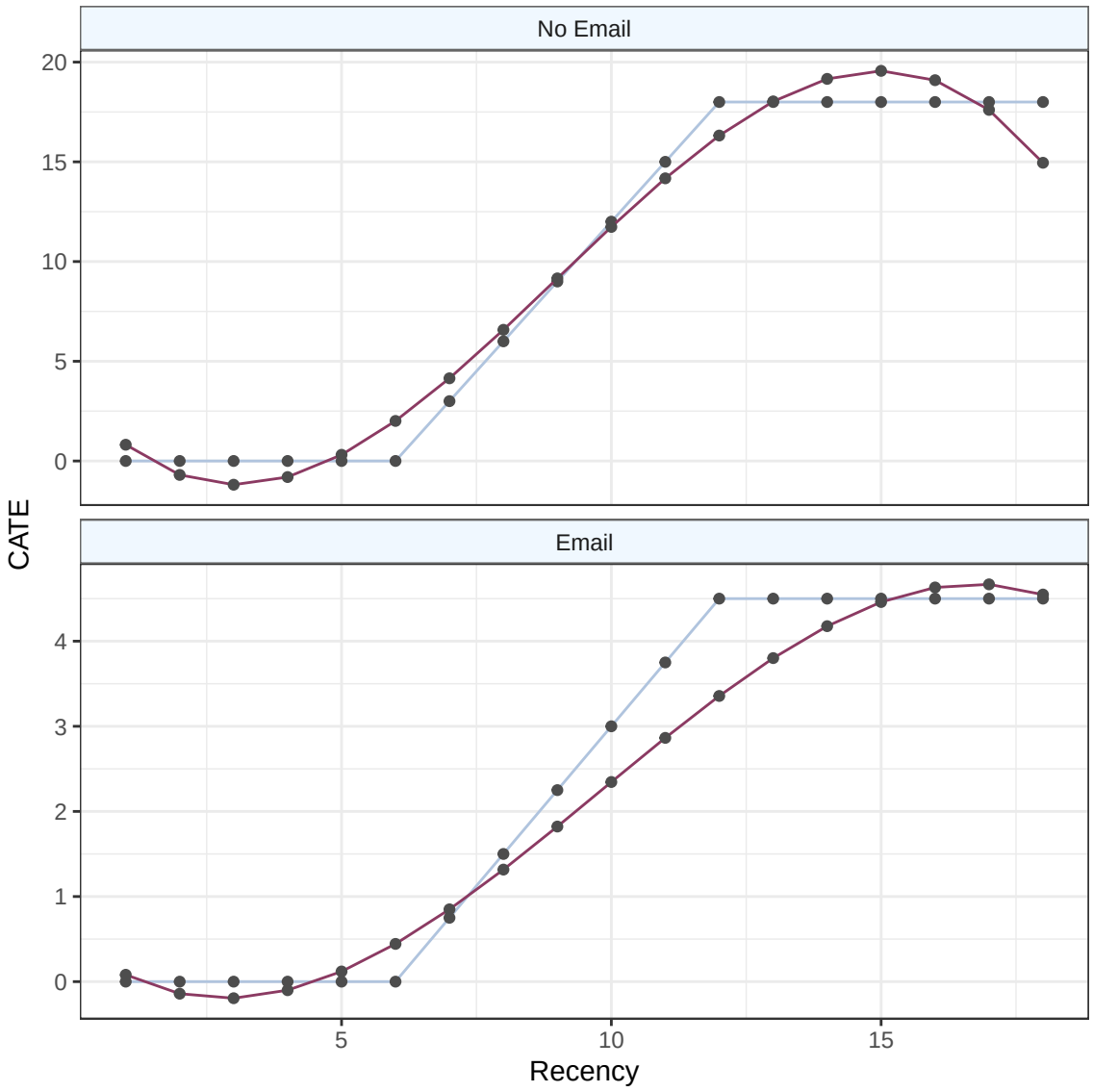
```
Call:
lm(formula = spend ~ . + . * target + email:recency * target,
    data = OLS_DT)

Residuals:
    Min      1Q  Median      3Q     Max
-43.135 -17.380  -7.339  -0.028 114.096

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)       -4.13787    0.36284 -11.404 < 0.0000000000000002 ***
recency            1.43455    0.03370  42.573 < 0.0000000000000002 ***
email             -0.02472    0.63135  -0.039              0.96877
target            -3.90365    0.51357  -7.601   0.0000000000000296 ***
recency:target     1.40856    0.04751  29.650 < 0.0000000000000002 ***
```

```
email:target          2.67532    0.89245   2.998              0.00272 **
recency:email         0.01406    0.05829   0.241              0.80934
recency:email:target -1.05100    0.08244 -12.748 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 31.92 on 99992 degrees of freedom
Multiple R-squared:  0.1167,    Adjusted R-squared:  0.1166
F-statistic:  1887 on 7 and 99992 DF,  p-value: < 0.00000000000000022
```

OLS with Polynomials

```
OLS_DT <- training_DT[, .(spend, recency, email, target)]

fit_OLS <- lm(spend ~ target*email + poly(recency,3)*email + poly(recency,3):target*email,
              data = OLS_DT)

pred_DT[, pred_spend_OLS_poly := predict(fit_OLS, pred_DT)]

summary_OLS_DT <- pred_DT[, .(tau_pred_OLS = mean(pred_spend_OLS_poly[target==1])
                                           - mean(pred_spend_OLS_poly[target==0])),
                          keyby = .(email, recency)]
summary_OLS_DT <- merge(summary_OLS_DT, summary_DT[, .(email, recency, tau)],
                        by = c("email", "recency"))
```

```
graph_DT <- copy(summary_OLS_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
   geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
   geom_line(aes(x = recency, y = tau_pred_OLS),
             color = "hotpink4", size = 0.5) +
   geom_point(aes(x = recency, y = tau_pred_OLS),
              color = "gray30", fill = "hotpink4", size = 1.5) +
   ylab("CATE") + xlab("Recency") +
   facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
   theme_bw() +
   theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```




```
summary(fit_OLS)
```

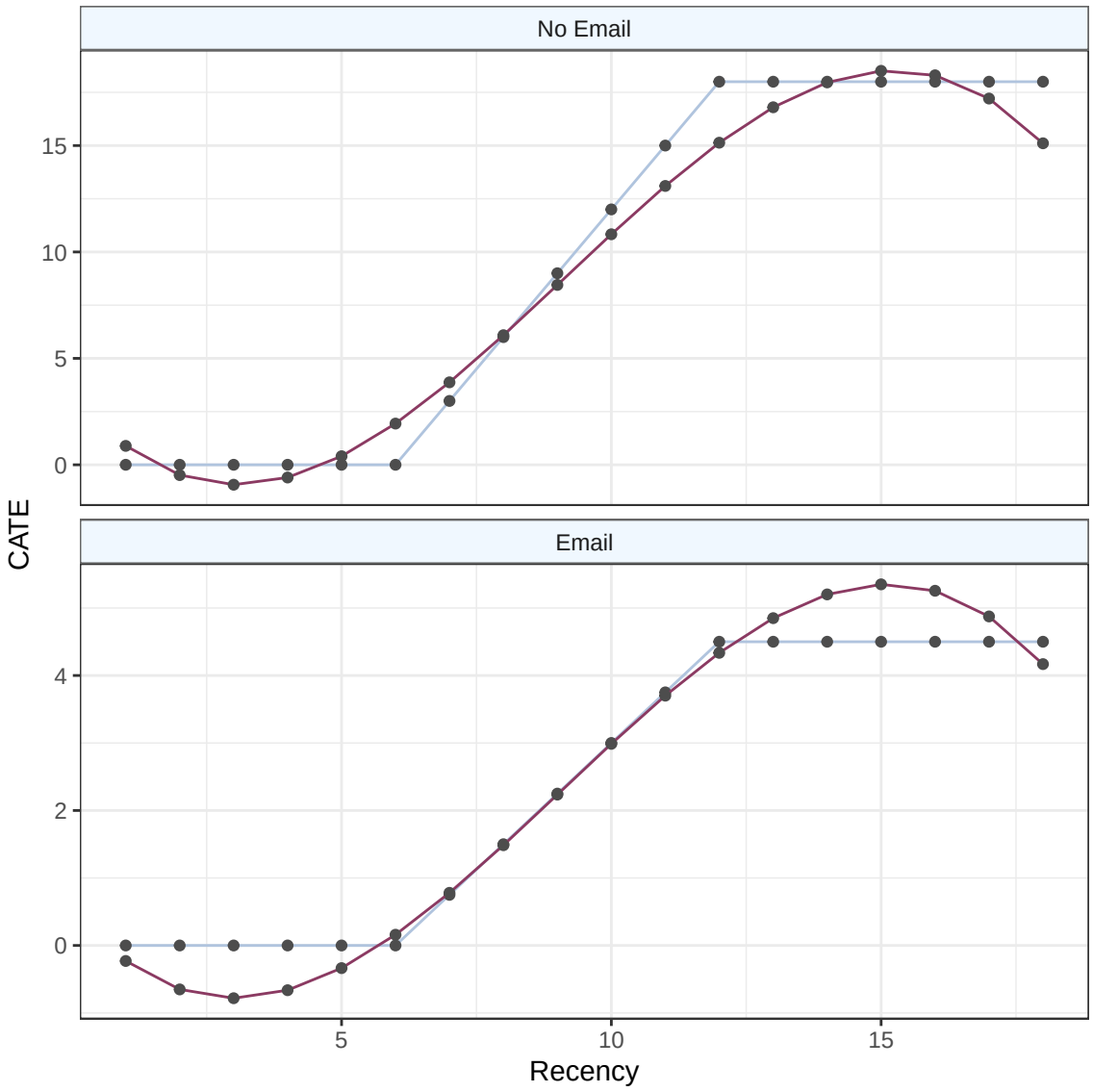
```
Call:
lm(formula = spend ~ target * email + poly(recency, 3) * email +
    poly(recency, 3):target * email, data = OLS_DT)

Residuals:
    Min      1Q  Median      3Q     Max
-38.970 -18.297  -4.989   0.672 115.868

Coefficients:
                              Estimate Std. Error t value
(Intercept)                     9.4592     0.1738  54.413
target                          9.4716     0.2456  38.570
email                           0.1350     0.3011   0.448
poly(recency, 3)1            2355.3579    55.0554  42.782
poly(recency, 3)2            -172.1075    54.9950  -3.130
```

```
poly(recency, 3)3               -727.5815    54.9744 -13.235
target:email                      -7.3071     0.4257 -17.165
email:poly(recency, 3)1           35.0505    95.2393   0.368
email:poly(recency, 3)2            6.1174    94.9908   0.064
email:poly(recency, 3)3          -18.8939    95.1966  -0.198
target:poly(recency, 3)1        2321.2524    77.6171  29.906
target:poly(recency, 3)2        -268.5050    77.6918  -3.456
target:poly(recency, 3)3        -829.9534    77.6244 -10.692
target:email:poly(recency, 3)1 -1747.2688   134.6962 -12.972
target:email:poly(recency, 3)2   293.0562   134.5754   2.178
target:email:poly(recency, 3)3   706.4091   134.6818   5.245
                                          Pr(>|t|)
(Intercept)                    < 0.0000000000000002 ***
target                         < 0.0000000000000002 ***
email                                     0.654012
poly(recency, 3)1              < 0.0000000000000002 ***
poly(recency, 3)2                         0.001751 **
poly(recency, 3)3              < 0.0000000000000002 ***
target:email                   < 0.0000000000000002 ***
email:poly(recency, 3)1                   0.712855
email:poly(recency, 3)2                   0.948652
email:poly(recency, 3)3                   0.842676
target:poly(recency, 3)1       < 0.0000000000000002 ***
target:poly(recency, 3)2                  0.000548 ***
target:poly(recency, 3)3       < 0.0000000000000002 ***
target:email:poly(recency, 3)1 < 0.0000000000000002 ***
target:email:poly(recency, 3)2            0.029435 *
target:email:poly(recency, 3)3          0.000000157 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 31.71 on 99984 degrees of freedom
Multiple R-squared:  0.1279,    Adjusted R-squared:  0.1277
F-statistic: 977.2 on 15 and 99984 DF,  p-value: < 0.00000000000000022
```

Lasso with Polynomials

```
LASSO_DT <- training_DT[, .(spend, recency, email, target)]

# No intercept here
X_mat <- model.matrix(~ 0 + target*email + poly(recency,3)*email + poly(recency,3):target*email, data =
Y_vec <- training_DT$spend

set.seed(12345)
fit_LASSO <- cv.gamlr(x = X_mat,
                      y = Y_vec,
                      nfold = 10L)

X_mat_pred <- model.matrix(~ 0 + target*email + poly(recency,3)*email + poly(recency,3):target*email, da

pred_DT[, pred_spend_Lasso := predict(fit_LASSO, X_mat_pred,  select="min")[,1]]

summary_LASSO_DT <- pred_DT[, .(tau_pred_LASSO = mean(pred_spend_Lasso[target==1])
                                                 - mean(pred_spend_Lasso[target==0])),
                            keyby = .(email, recency)]
summary_LASSO_DT <- merge(summary_LASSO_DT, summary_DT[, .(email, recency, tau)],
                          by = c("email", "recency"))
```

```
graph_DT <- copy(summary_LASSO_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
   geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
   geom_line(aes(x = recency, y = tau_pred_LASSO),
             color = "hotpink4", size = 0.5) +
   geom_point(aes(x = recency, y = tau_pred_LASSO),
              color = "gray30", fill = "hotpink4", size = 1.5) +
   ylab("CATE") + xlab("Recency") +
   facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
   theme_bw() +
   theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```



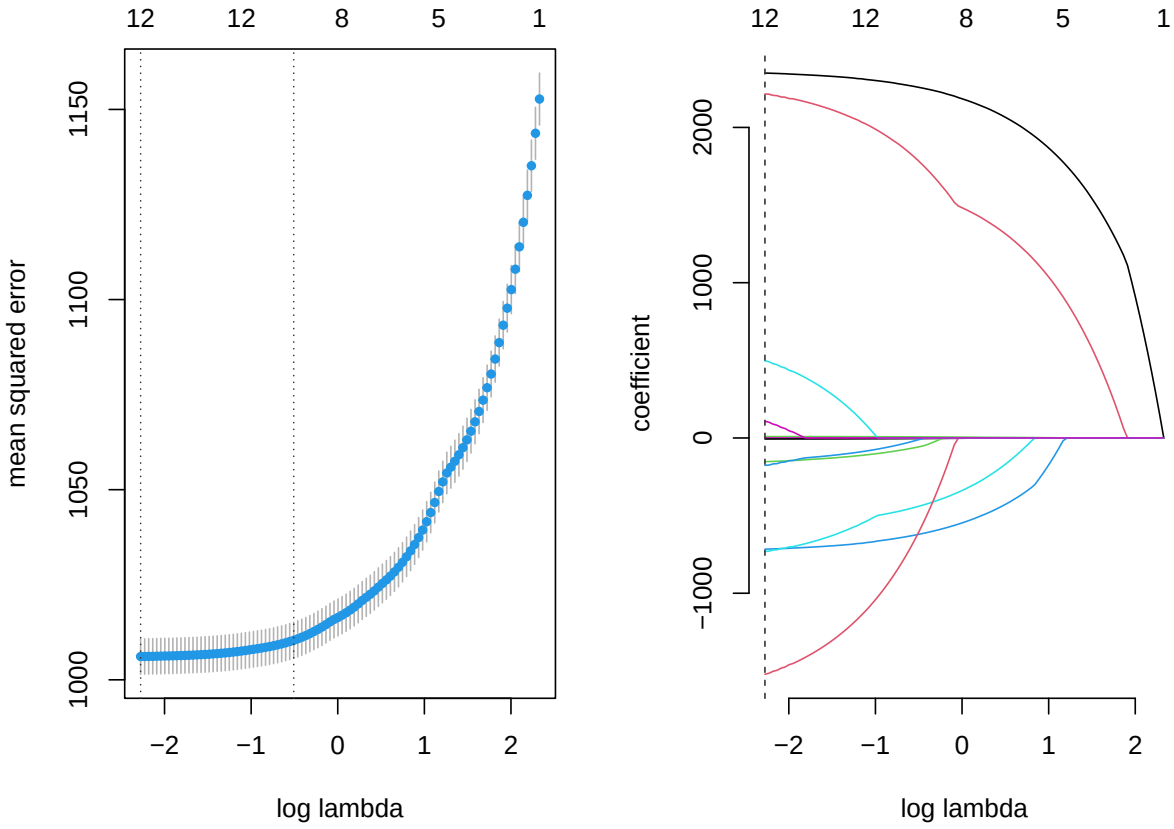

```
coef(fit_LASSO, select = "min")
```

```
16 x 1 sparse Matrix of class "dgCMatrix"
                                    seg100
intercept                         9.610541
target                            9.048303
email                             .
poly(recency, 3)1              2350.132873
poly(recency, 3)2              -152.593385
poly(recency, 3)3              -716.236884
target:email                     -6.669777
email:poly(recency, 3)1           .
email:poly(recency, 3)2           .
email:poly(recency, 3)3           .
target:poly(recency, 3)1       2215.650063
target:poly(recency, 3)2       -176.311541
target:poly(recency, 3)3       -730.868992
```

```
target:email:poly(recency, 3)1 -1521.641996
target:email:poly(recency, 3)2   107.705104
target:email:poly(recency, 3)3   497.253281
```

Lasso coefficients look similar to those of the OLS with polynomials.We can also look at the cross-validation out-of-sample error and its regularization path.

```
par(mfrow=c(1,2))
# Lasso CV Error
plot(fit_LASSO)
# Lasso regularization path for selected
plot(fit_LASSO$gamlr)
par(mfrow=c(1,1))
```

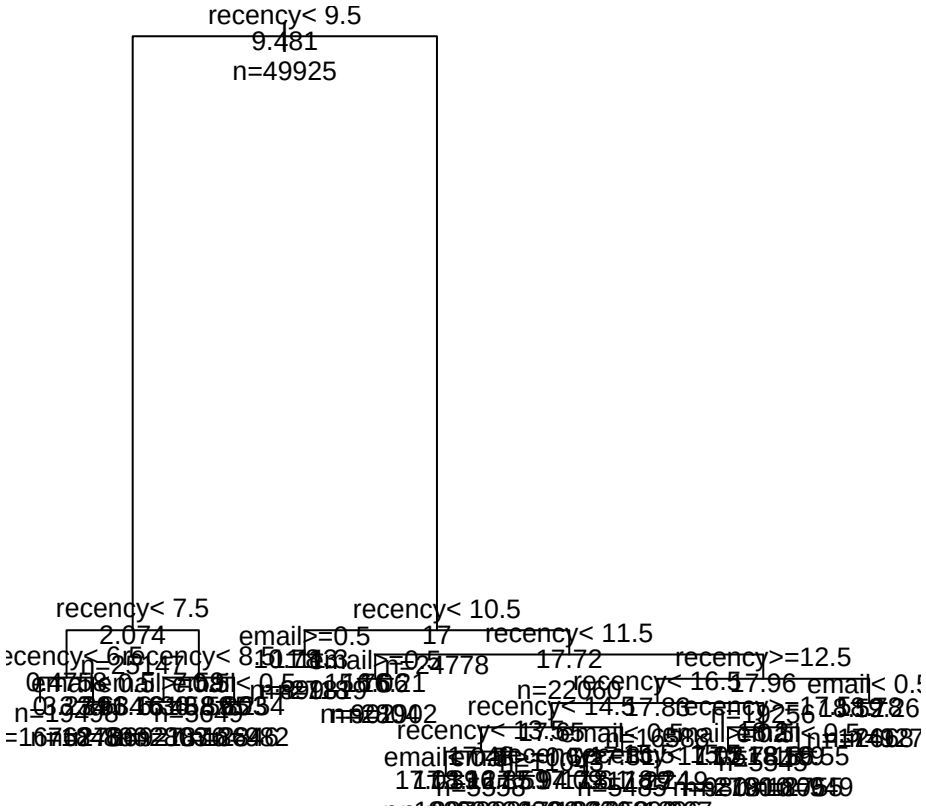


Two Trees

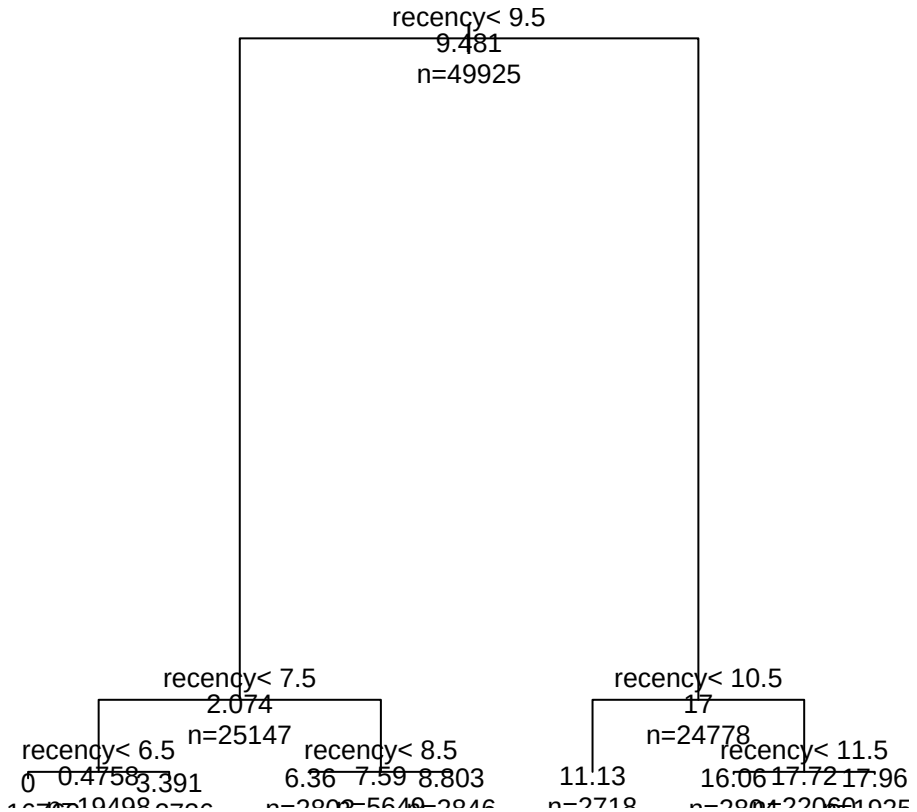
```
set.seed(12345)
fit_CART_0 <- rpart(spend ~ recency + email,
                    data = training_DT[target == 0],
                    method = "anova", # Regression tree
                    control = rpart.control(cp = 0, minsplit = 10))

fit_CART_1 <- rpart(spend ~ recency + email,
                    data = training_DT[target == 1],
                    method = "anova", # Regression tree
                    control = rpart.control(cp = 0, minsplit = 10))
```

We can examine the two regression trees.

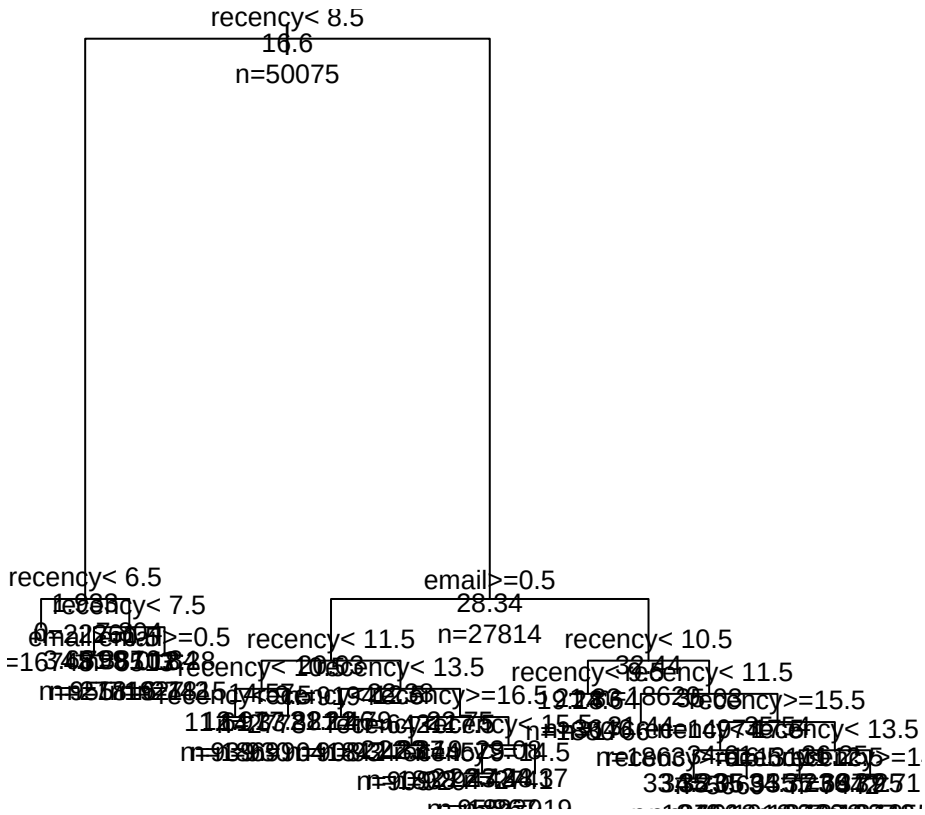
```
# Not targeted
plot(fit_CART_0)
text(fit_CART_0, use.n=TRUE, all=TRUE, cex=.8)
```



```
# Prune the tree (not targeted)
bestcp_0 <- fit_CART_0$cptable[which.min(fit_CART_0$cptable[,"xerror"]),"CP"]
fit_CART_0 <- prune(fit_CART_0, cp = bestcp_0)
plot(fit_CART_0)
text(fit_CART_0, use.n=TRUE, all=TRUE, cex=.8)
```



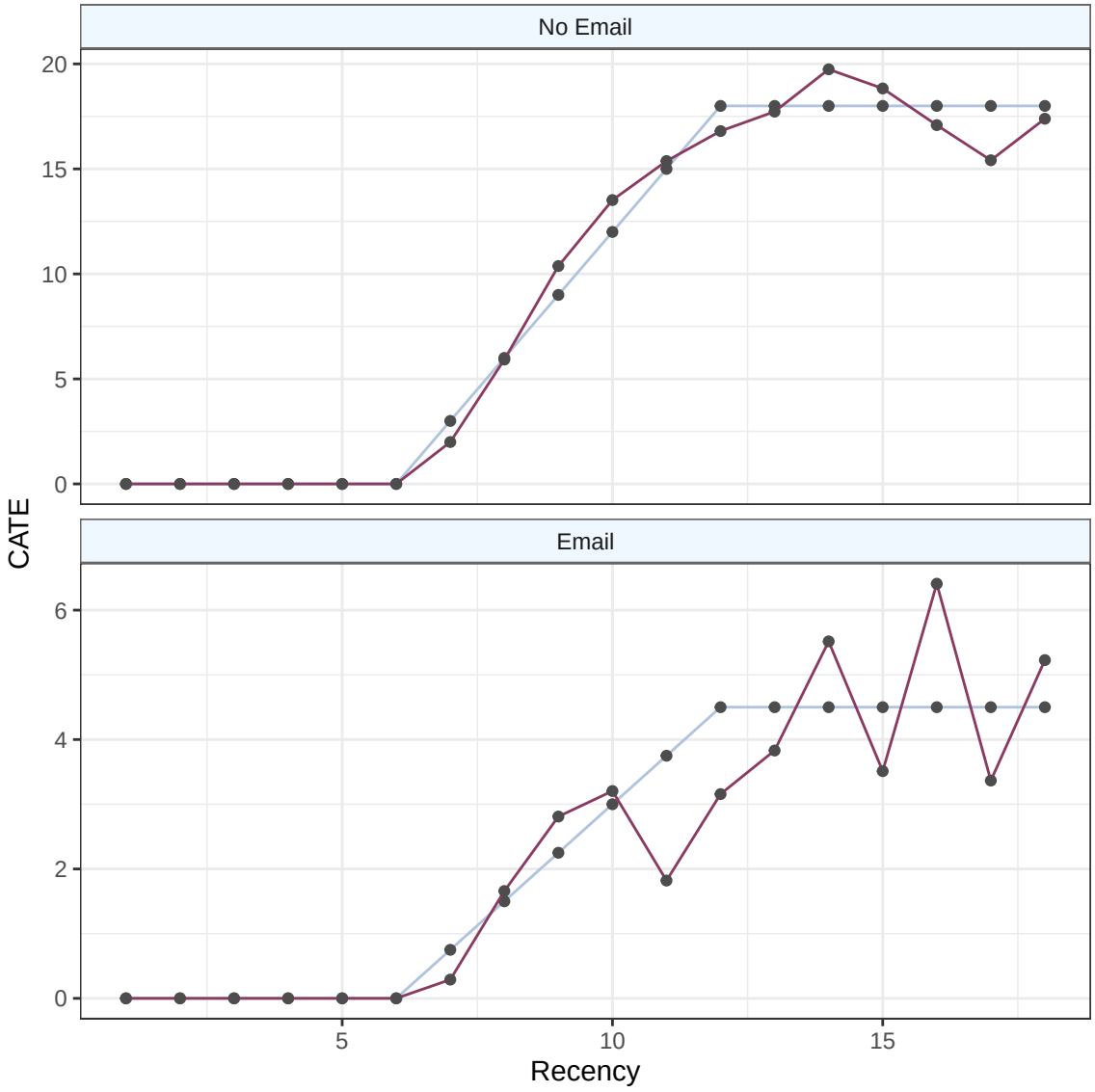

```
# Targeted
## Prune the tree (targeted)
bestcp_1   <- fit_CART_1$cptable[which.min(fit_CART_1$cptable[,"xerror"]),"CP"]
fit_CART_1 <- prune(fit_CART_1, cp = bestcp_1)
plot(fit_CART_1)
text(fit_CART_1, use.n=TRUE, all=TRUE, cex=.8)
```




```
# Create counterfactual data sets
X_pred_DT_0 <- copy(pred_DT[, .(recency, email)])
X_pred_DT_0[, target:= 0]

X_pred_DT_1 <- copy(pred_DT[, .(recency, email)])
X_pred_DT_1[, target:= 1]

pred_DT[, TE_CART_TT := predict(fit_CART_1, X_pred_DT_1) -
                        predict(fit_CART_0, X_pred_DT_0)]

summary_TT_DT <- pred_DT[, list(tau_pred_CART_TT = mean(TE_CART_TT)),
                         keyby = .(email, recency)]
summary_TT_DT <- merge(summary_TT_DT, summary_DT[, .(email, recency, tau)],
                       by = c("email", "recency"))
```

```
graph_DT <- copy(summary_TT_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
```

```
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
   geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
   geom_line(aes(x = recency, y = tau_pred_CART_TT),
             color = "hotpink4", size = 0.5) +
   geom_point(aes(x = recency, y = tau_pred_CART_TT),
              color = "gray30", fill = "hotpink4", size = 1.5) +
   ylab("CATE") + xlab("Recency") +
   facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
   theme_bw() +
   theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```

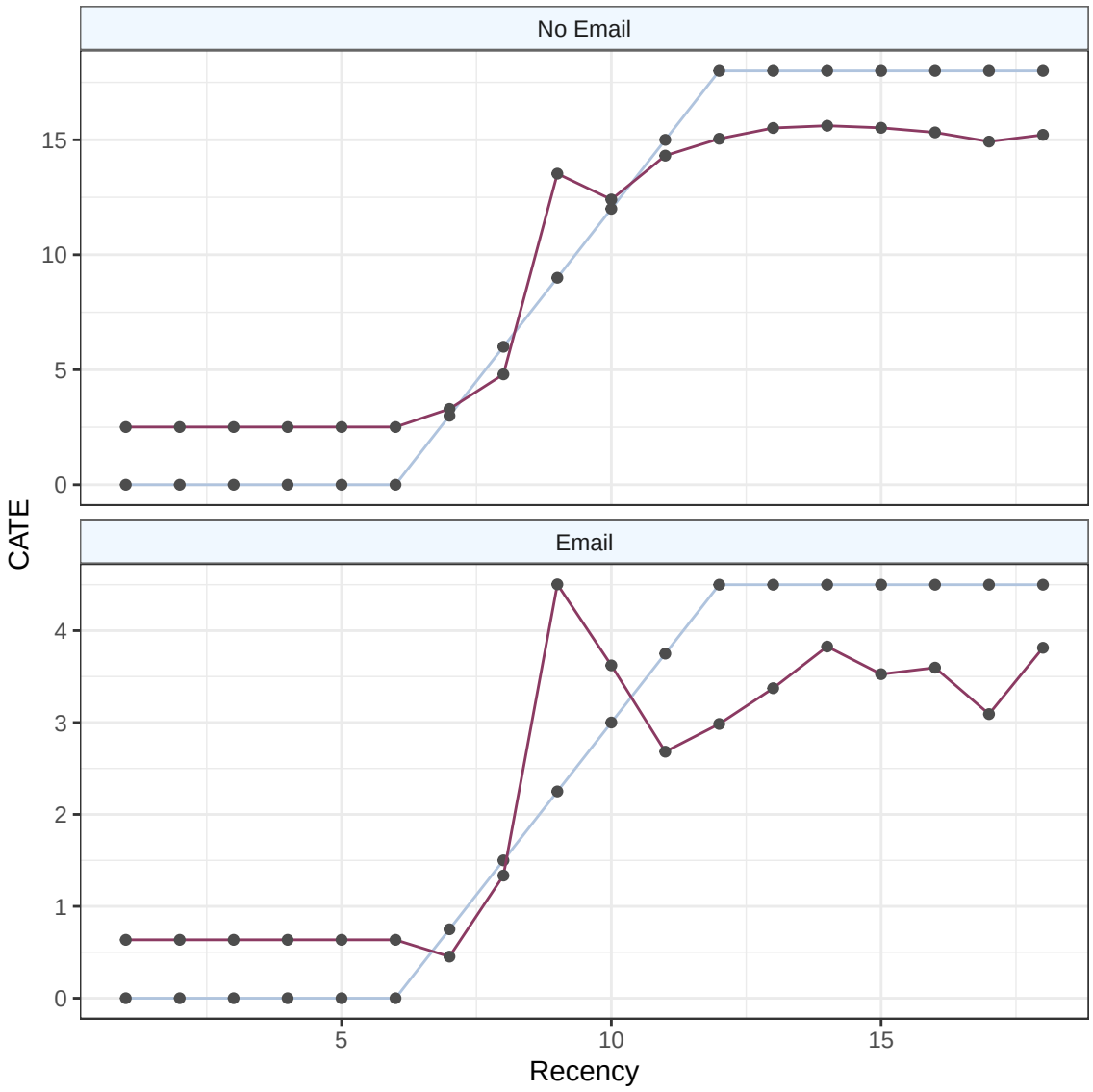


Two Forests

Recall that we train a random forest on the treated data and another on the untreated data. The intuition here is that if we trained one forest on the whole feature set including the treatment variable, we force the tree to first split on the treatment variable.

```
fit_RF_0 <- ranger(spend ~ recency + email,
                   data = training_DT[target == 0],
                   num.trees   = 1000,
                   num.threads = ncores,
                   seed = 12345L)

fit_RF_1 <- ranger(spend ~ recency + email,
                   data = training_DT[target == 1],
                   num.trees   = 1000,
                   num.threads = ncores,
                   seed = 12345L)

# Create counterfactual data sets
X_pred_DT_0 <- copy(pred_DT[, .(recency, email)])
X_pred_DT_0[, target:= 0]

X_pred_DT_1 <- copy(pred_DT[, .(recency, email)])
X_pred_DT_1[, target:= 1]


pred_DT[, TE_RF_TF := predict(fit_RF_1, X_pred_DT_1, num.threads = ncores)$predictions -
                        predict(fit_RF_0, X_pred_DT_0, num.threads = ncores)$predictions]

summary_TF_DT <- pred_DT[, list(tau_pred_RF_TF = mean(TE_RF_TF)),
                         keyby = .(email, recency)]
summary_TF_DT <- merge(summary_TF_DT, summary_DT[, .(email, recency, tau)],
                       by = c("email", "recency"))
```

```
graph_DT <- copy(summary_TF_DT)
graph_DT[, email_flag := ifelse(email == 1, "Email", "No Email")]
graph_DT[, email_flag := factor(email_flag, c("No Email", "Email"))]

ggplot(graph_DT, aes(x = recency, y = tau)) +
   geom_line(color = "lightsteelblue", size = 0.5) +
   geom_point(color = "gray30", fill = "lightsteelblue", size = 1.5) +
   geom_line(aes(x = recency, y = tau_pred_RF_TF),
             color = "hotpink4", size = 0.5) +
   geom_point(aes(x = recency, y = tau_pred_RF_TF),
              color = "gray30", fill = "hotpink4", size = 1.5) +
   ylab("CATE") + xlab("Recency") +
   facet_wrap(~ email_flag, nrow = 2, scales = "free_y") +
   theme_bw() +
   theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```

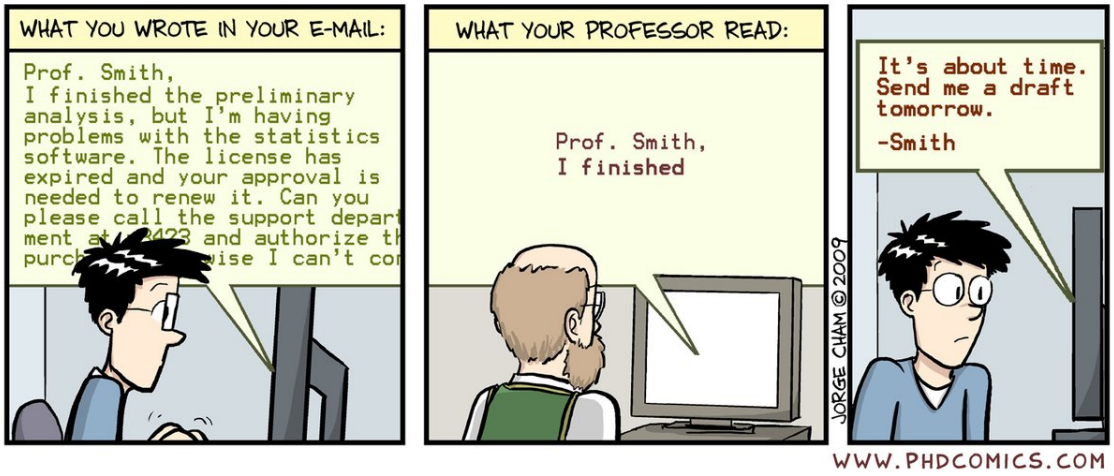

No Email
Email
CATE
Recency
0
5
10
15
1
2
3
4

Next Steps

Exercise: Try playing around with the non-linear treatment effect part of the simulation. Can you construct a scenario where the OLS outperforms the Causal Forest in the validation set? What do the treatment effects need to look like for that to occur?

Part III
Optimization Theory

This section provides an overview of the commonly used optimization methods. These concepts should provide background for the first year operation courses, courses at the statistics department, and the third quarter price theory course at the Economics department. The companion RMarkdown notebook provides a testbed of different optimizers on the Rosenbrock Banana Function.



7 Fundamentals

> "An equation of cruel optimization exists when something you model is actually a limit on your behavior. It might involve profit, or a kind of utility; it might be a fantasy of game theoretic conduct, or a political project. It might rest in something simpler, too, like new jargon that promises to induce in you a tenure-tracked way of being. These kinds of optimization are not inherently cruel. They become cruel only when the object that draws your analysis actively impedes the aim that brought you to it initially." (JWK 2022)

Optimization theory, or *cruel optimization*[3], underlies all the different fields within the business school. We provide an overview of unconstrained, constrained, and convex optimization in this chapter and allude to these core concepts throughout the rest of the notes.

7.1 Unconstrained Optimization

We will focus on the the **maximization** problem. Any **minimization** problem can be transformed into a maximization problem by flipping the sign. An **unconstrained optimization** problem is:

For some $S \subset \mathbb{R}^k$ and some function $f : S \to \mathbb{R}$,

$$\max_{x \in S} f(x),$$

which we call the unconstrained optimization problem (or program) through this section. Let x^* be one element in the set of the maximizers, then we write

$$x^* \in \arg\max_{x \in S} f(x).$$

If the optimization problem has a unique solution, we write

$$x^* = \arg\max_{x \in S} f(x).$$

Example 18. (Monopolist profit maximization) A monopolist firm solves the profit maximization problem

$$\max_{p \in [0,\infty)} p \cdot q(p) - c(q(p)),$$

where p is price, $q(\cdot)$ is the demand function, and $c(\cdot)$ is the cost function.

Definition 4. For an optimization problem, let S be the set of points that satisfy all constraints. Then, $x^* \in S$ is a **local maximum** if $\exists \epsilon > 0$, such that $\|x^* - x\| < \epsilon$ implies $f(x^*) \geq f(x)$. We say $x^* \in S$ is a **strict local maximum** if $\exists \epsilon > 0$, such that for $x \neq x^*$, $\|x^* - x\| < \epsilon$ implies $f(x^*) > f(x)$. Also, $x^* \in S$ is a **global maximum** if $\forall x \in S$, $f(x^*) \geq f(x)$; accordingly, x^* is said to be a **strict global maximum** if $\forall x \in S$, $f(x^*) > f(x)$.

We introduce a set of *necessary* conditions for local maxima.

[3] In reference to concept of *cruel optimism*: `https://www.dukeupress.edu/cruel-optimism`

For some differentiable function $f : S \to \mathbb{R}$ with $S \subset \mathbb{R}^k$, the gradient of f is

$$\nabla f(x_1, \ldots, x_k) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x_1, \ldots, x_k) \\ \vdots \\ \frac{\partial}{\partial x_k} f(x_1, \ldots, x_k) \end{bmatrix}.$$

Theorem 6. *(First Order Condition/FOC) We say $x \in S$ is an **interior point** of S if there exists an open set $U \subset S$ such that $x \in U$. Let f be differentiable at some $x^* \in S$. For the unconstrained optimization problem, if x^* is an interior point of S and a local maximum of f, then we must have $\nabla f(x^*) = 0$.*

Proof. Suppose $\nabla f(x^*) \neq 0$. WLOG, let $f_1(x^*) > 0$. Pick ϵ such that $0 < \epsilon < f_1(x^*)$. Note that

$$f_1(x^*) = \lim_{h \downarrow 0} \frac{f(x^* + he_1) - f(x^*)}{h},$$

so there exists $\delta > 0$ such that $|h| < \delta$ implies

$$\left| \frac{f(x^* + he_1) - f(x^*)}{h} - f_1(x^*) \right| < \epsilon.$$

This implies for each $0 < h < \delta$,

$$f_1(x^*) - \epsilon < \frac{f(x^* + he_1) - f(x^*)}{h} < f_1(x^*) + \epsilon,$$

i.e.

$$f(x^* + he_1) > f(x^*) + (f_1(x^*) - \epsilon)h.$$

Thus, within any neighborhood of x^* with radius d, pick h such that $0 < h < \min\{\delta, d\}$ and we have

$$f(x^* + he_1) > f(x^*),$$

contradicting x^* being local maximum. □

Definition 5. We call $x \in S$ a **critical point** if $\nabla f(x) = 0$. That means, when solving the unconstrained optimization problem, we only need to check all the critical points and boundary points of S.

If f is twice-differentiable, we write the second partial derivative for some $i, j \in \{1, 2, \ldots, k\}$ as

$$f_{ij}(x_1, \ldots, x_k) = \frac{\partial^2}{\partial x_i \partial x_j} f(x_1, \ldots, x_k) = \frac{\partial}{\partial x_j} \left(\frac{\partial}{\partial x_i} f(x_1, \ldots, x_k) \right).$$

Definition 6. We define the **Hessian** of f at $x = (x_1, \ldots, x_k)$ as

$$\nabla^2 f(x_1, \ldots, x_k) = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1k} \\ f_{21} & f_{22} & \cdots & f_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ f_{k1} & f_{k2} & \cdots & f_{kk} \end{bmatrix}.$$

This matrix is also denoted as the *Hessian matrix* of f.

Definition 7. A function $f : S \to \mathbb{R}$ is **continuously differentiable** if the partial derivative $\partial f / \partial x_i$ is continuous for each i. We say f is **twice-continuously differentiable** if for each $i, j \in \{1, 2, \ldots, k\}$, $f_{i,j} : S \to \mathbb{R}$ is a continuous function.

Theorem 7. *(Second Order Condition/SOC) Suppose f is twice-continuously differentiable in the unconstrained optimization problem. If x^* is an interior point of S and a local maximum, then $\nabla^2 f(x^*)$ is negative semi-definite.*

Remark. Unless the problem is univariate, we rarely use the SOC because it is tedious to check whether a matrix is negative semi-definite.

Theorem 8. *(Envelope Theorem) Let $f(x; a)$ be a function of $x \in S \subset \mathbb{R}^k$ and $a \in A \subset \mathbb{R}$. Assume $f(x; a)$ is continuously differentiable in $x \in S$ for each $a \in A$ and S is open. For each $a \in A$, let $x^*(a) = \arg\max_{x \in S} f(x; a)$ and assume $x^*(a)$ is continuously differentiable in a. Then,*

$$\frac{d}{da} f(x^*(a); a) = \frac{\partial}{\partial a} f(x^*(a); a) = \left. \frac{\partial}{\partial a} f(x; a) \right|_{x = x^*(a)}.$$

Proof. Write out the total derivative and apply the first order condition. □

Example 19. Suppose the profit of a firm is determined by

$$\pi(x; p) = px - c(x),$$

where x is quantity, p is the price of the product, and $c(x)$ is differentiable. We assume the firm is a price-taker such that we treat p as given. Assume $c(x)$ is smooth. The FOC of the profit maximization problem requires $p = c'(x^*)$ (or equivalently that marginal revenue equals to marginal cost), where x^* is the maximizer. Then, the Envelope Theorem says

$$\frac{d}{dp} \pi(x^*(p); p) = x^*(p).$$

One popular approach to computationally determining the optimal point is gradient descent and stochastic gradient descent.

7.2 Constrained Optimization

A constrained optimization problem is just an unconditional optimization problem subject to constraints. If the constraints bind, or play a role in the optimization problem, then for the same objective function, the constrained solution generally deviates from the unconstrained problem.

Example 20. (Consumer utility maximization) A consumer facing a budget constraint solves the optimization problem

$$\begin{aligned} \max_{x_1,\ldots,x_n} \; & u(x_1,\ldots,x_n) \\ s.t. \; & p_1x_1 + \cdots + p_nx_n \leq m \\ & x_i \geq 0 \text{ for } i = 1,\ldots,n, \end{aligned}$$

where $u(\cdot)$ is the utility function, p_i the price for x_i, and m is the budget.

Example 21. (Monopolist profit maximization with a price cap) Revisit our example monopolist program from before, we assume the monopolist firm solves the profit maximization problem but faces a price cap of p^0 imposed by the government.

$$\begin{aligned} \max_{p\in[0,\infty)} \; & p \cdot q(p) - c(q(p)) \\ s.t. \; & p \leq p^0 \end{aligned}$$

where p is price, $q(\cdot)$ is the demand function, and $c(\cdot)$ is the cost function.

Remark.

1. Most microeconomics problems are written as a constrained optimization problem. Usually some form of a constraint exists in reality that impacts the optimization program (i.e. economies of scale occur up to a point or scarce resources exist).
2. The "economics" of the optimization problem can be boiled down to the researcher's choice of the objective function and the constraint. For example, for the consumer's problem, consumer preferences are pinned down the objective function and the problem's setting are pinned down by the constraint.

More generally, a **constrained optimization** is defined as for some open set $S \subset \mathbb{R}^k$,

$$\begin{aligned} \max_{x\in S} \quad & f(x) \\ s.t. \quad & g_i(x) \geq 0, \; \forall i = 1,\ldots,n \\ & h_j(x) = 0, \; \forall j = 1,\ldots,m. \end{aligned}$$

The **Lagrangian** of this optimization problem is

$$\mathcal{L}(x,\lambda,\mu) = f(x) + \sum_{i=1}^{n} \lambda_i g_i(x) + \sum_{j=1}^{m} \mu_j h_j(x) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

where $\lambda = (\lambda_1, \ldots, \lambda_n)^T$, $\mu = (\mu_1, \ldots, \mu_m)^T$, $g(x) = (g_1(x), \ldots, g_n(x))^T$, $h(x) = (h_1(x), \ldots, h_m(x))^T$. Here λ and μ are defined as the **Lagrangian Multipliers**.

The **Karush-Kuhn-Tucker** conditions (KKT) are given by

1. (primal constraints) $g_i(x) \geq 0, \forall i = 1, \ldots, n$; $h_j(x) = 0, \forall j = 1, \ldots, m$.
2. (dual constraints) $\lambda_i \geq 0, \forall i = 1, \ldots, n$.
3. (complementary slackness) $\lambda_i g_i(x) = 0, \forall i = 1, \ldots, n$.
4. (vanishing gradient) $\nabla_x \mathcal{L} = 0$.

KKT is a set of conditions that is necessary for optimality under regularity conditions (say, f, g_i, h_j are continuously differentiable).

Example 22. Consider the following program.

$$\begin{aligned} \max_{x,y} \; & x - y^2 \\ s.t. \; & x \geq 0, y \geq 0 \\ & x^2 + y^2 = 4. \end{aligned}$$

KKT conditions specify:

$$\begin{aligned} \text{Primal} &: x \geq 0, y \geq 0, x^2 + y^2 = 4 \\ \text{Dual} &: \lambda_1 \geq 0, \lambda_2 \geq 0 \\ \text{Complementary Slackness} &: \lambda_1 x = 0, \lambda_2 y = 0 \\ \text{Gradient} &: 1 + \lambda_1 + 2\mu x = 0, -2y + \lambda_2 + 2\mu y = 0, \end{aligned}$$

which yields $(x^*, y^*) = (2, 0)$.

Example 23. (Simplified utility maximization) Suppose there are two goods and ignore the positivity constraint. A consumer facing a budget constraint solves the optimization problem

$$\begin{aligned} \max_{x_1,x_2} \; & u(x_1, x_2) \\ s.t. \; & p_1 x_1 + p_2 x_2 \leq m \end{aligned}$$

The KKT conditions specify:

$$\begin{aligned} \text{Primal} &: m - p_1 x_1 - p_2 x_2 \geq 0 \\ \text{Dual} &: \lambda \geq 0 \\ \text{Complementary Slackness} &: \lambda(m - p_1 x_1 - p_2 x_2) = 0 \\ \text{Gradient} &: \partial u/\partial x_1 - \lambda p_1 = 0, \partial u/\partial x_2 - \lambda p_2 = 0, \end{aligned}$$

implying

$$\lambda = \frac{\partial u/\partial x_1}{p_1} = \frac{\partial u/\partial x_2}{p_2},$$

i.e. utility increment of spending one more dollar on good x_1 is equal to that of good x_2.

7.3 Convex Optimization

Definition 8. A set $X \subset \mathbb{R}^k$ is a **convex set** if for each $\theta \in (0,1)$ and $x_1, x_2 \in X$, $\theta x_1 + (1-\theta)x_2 \in X$, where $x = \theta x_1 + (1-\theta)x_2$ is called a **convex combination** of x_1 and x_2.

In other words, a convex set is closed under convex combination. The **convex hull** of a set S is the set of all convex combinations of points in S.

Proposition 3. *Suppose $A, B \subset X$. If A and B are convex, then $A \cap B$ is convex.*

Proof. Use the prior definition. □

Definition 9. A set X is called a **hyperplane** if $X = \{x \in \mathbb{R}^k | a^T x = b\}$ for some nonzero $a \in \mathbb{R}^k$ and $b \in \mathbb{R}$.

Theorem 9. *(Separating Hyperplane Theorem) If $C, D \subset \mathbb{R}^k$ are nonempty disjoint convex sets, there exists a nonzero vector $a \in \mathbb{R}^k$ and $b \in \mathbb{R}$ such that $a^T x \leq b$ for each $x \in C$ and $a^T x \geq b$ for each $x \in D$.*

Theorem 10. *(Supporting Hyperplane Theorem) A supporting hyperplane to a set C at a boundary point x_0 is a hyperplane X such that $X = \{x \in \mathbb{R}^k | a^T x = a^T x_0\}$ for some nonzero vector $a \in \mathbb{R}^k$, and $a^T x \leq a^T x_0$ for each $x \in C$. If C is a convex set, then there exists a supporting hyperplane to C at each boundary point of C.*

Remark.

1. The separating hyperplane theorem says that if there are two disjoint, convex sets in finite dimensional Euclidean space, then you can draw a hyperplane (or "line") that separates the two.
2. The supporting hyperplane theorem says that for a convex set, there exists a hyperplane that intersects the a point on the boundary of the set once. This hyperplane is the "supporting hyperplane". Note that a point on the boundary of a convex set can have many supporting hyperplanes. Further, the convexity of the set is importance because otherwise the supporting hyperplane for boundary point can intersect the set somewhere else (and would no longer be a "supporting hyperplane")
3. The best way to get intuition for these two theorems is to draw $2-D$ convex sets with the hyperplane being a line.

Definition 10. A function $f : X \to \mathbb{R}$ is a **convex function** if X is convex and for each $\theta \in (0,1)$ and $x_1, x_2 \in X$, $f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$. We say f is **strictly convex** if X is convex and for each $\theta \in (0,1)$ and $x_1, x_2 \in X$, $f(\theta x_1 + (1-\theta)x_2) < \theta f(x_1) + (1-\theta) f(x_2)$.

Definition 11. A function $f : X \to \mathbb{R}$ is **concave** if $-f$ is convex. We say f is **strictly concave** if $-f$ is strictly convex.

Lemma 3. *Let $X \subset \mathbb{R}^k$. If $f : X \to \mathbb{R}$ is convex, then $C = \{(x, y) \in \mathbb{R}^{k+1} : y \geq f(x), x \in X\}$ is convex.*

Proof. Let $z_1 = (x_1, y_1)$, $z_2 = (x_2, y_2) \in C$. Then we have $y_1 \geq f(x_1)$ and $y_2 \geq f(x_2)$. We want to show convex combination $z = \lambda z_1 + (1-\lambda) z_2 \in C$ for $\lambda \in (0,1)$. Note that by convexity,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \leq \lambda y_1 + (1-\lambda) y_2$$

so in turn,

$$z = (\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2) \in C.$$

□

Theorem 11. *(Jensen's Inequality) If $S \subset \mathbb{R}^k$ and $f : S \to \mathbb{R}$ is convex, then for some random variable X such that $\Pr(X \in S) = 1$ and $E[|X|] < \infty$, we have $f(E[X]) \leq E[f(X)]$.*

Proof. Let $z_0 = (E[X]^T, f(E[X]))^T$. By the previous lemma and the supporting hyperplane theorem, at $z_0 \in \mathbb{R}^{k+1}$, there exists a supporting hyperplane to the set $C = \{(x, y) \in \mathbb{R}^{k+1} : y \geq f(x), x \in S\}$ such that $Z = \{z \in \mathbb{R}^{k+1} : a^T z = a^T z_0\}$, where we write $a = (a_x^T, a_y)^T$ to denote the x part and y part, separately. Note $a_y \neq 0$, unless S is singleton, in which case Jensen's inequality trivially holds.

Then for each $(x^T, y)^T \in Z$, we can write $y = -a_y^{-1} a_x^T x + a_y^{-1} a^T z_0$ using the supporting hyperplane construction and note $y \leq f(x)$ for points x and outcome y on the supporting hyperplane. Therefore,

$$E[f(X)] \geq E[-a_y^{-1} a_x^T X + a_y^{-1} a^T z_0] = -a_y^{-1} a_x^T E[X] + a_y^{-1} a^T z_0 = f(E[X]).$$

The inequality holds since we are taking expectations over $y \leq f(x)$ and for point $(X^T, y)^T \in Z$ on the supporting hyperplane, we see $a_x^T X + a_y y = a^T z_0$ (and we can rearrange the equation to isolate y. Then, the first equality is because of linearity and the last equality holds because $z_0 \in Z$ so $a_x^T E[X] + a_y f(E[X]) = a^T z_0$ (and we can rearrange the equation to isolate $f(E[X])$). □

A **convex optimization problem** is a constrained optimization problem where f and $g_1, \ldots, g_n$ are all concave functions, and $h_1, \ldots, h_m$ are affine functions.

Lemma 4. *If $g : S \to \mathbb{R}$ is concave, then $\{x \in S : g(x) \geq 0\}$ is convex where $g(x)$ are the inequality constraints in the optimization problem.*

Proof. Use the prior definition. □

Theorem 12. *A local maximum of a convex optimization problem is a global maximum.*

Proof. Let S be the set of points that satisfy all constraints. Then S is convex by previous lemma and proposition. The optimization problem becomes $\max_{x \in S} f(x)$, for S convex and f concave. Let x_0 be a local maximization to this problem, and suppose by way of contradiction that there exists $x_1 \in S$ such that $f(x_1) > f(x_0)$. For each $\epsilon > 0$, pick λ such that

$$0 < \lambda < \frac{\epsilon}{\|x_0 - x_1\|}.$$

Let the convex combination be $z = \lambda x_1 + (1-\lambda)x_0$, and we have

$$\|x_0 - z\| = \lambda \|x_0 - x_1\| \leq \epsilon$$

so $z \in B_\epsilon(x_0)$. But $f(z) \geq \lambda f(x_1) + (1-\lambda) f(x_0) > f(x_0)$, contradicting x_0 being local maximum. □

8 Topics

8.1 Duality Gap

There are primal and dual representation of an optimization problem or program. In our KKT formulation, we wrote the **primal** representation of a constrained optimization problem. For some open set $S \subset \mathbb{R}^k$, the primal problem is,

$$\begin{aligned} \max_{x \in S} \quad & f(x) \\ s.t. \quad & g_i(x) \geq 0, \ \forall i = 1, \ldots, n \\ & h_j(x) = 0, \ \forall j = 1, \ldots, m. \end{aligned}$$

The Lagrangian of the primal problem is

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{n} \lambda_i g_i(x) + \sum_{j=1}^{m} \mu_j h_j(x) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

where $\lambda = (\lambda_1, \ldots, \lambda_n)^T$, $\mu = (\mu_1, \ldots, \mu_m)^T$, $g(x) = (g_1(x), \ldots, g_n(x))^T$, $h(x) = (h_1(x), \ldots, h_m(x))^T$, and λ and μ are the Lagrangian Multipliers. The optimal value of the primal problem, denoted as p^* satisfies

$$p^* = \sup_x \inf_{\lambda \geq 0, \mu} L(x, \lambda, \mu).$$

Given the Lagrangian, the *Lagrange dual function* is

$$d(\lambda, \mu) = \sup_x L(x, \lambda, \mu)$$

The equivalent **dual** representation of the problem is,

$$\begin{aligned} & \min_{\lambda, \mu} d(\lambda, \mu) \\ & s.t. \ \lambda_i \geq 0, \ \forall i = 1, \ldots, n \end{aligned}$$

or equivalently

$$\begin{aligned} & \min_{\lambda, \mu} \sup_{x \in S} f(x) + \lambda^T g(x) + \mu^T h(x) \\ & \quad s.t. \ \lambda_i \geq 0, \ \forall i = 1, \ldots, n \end{aligned}$$

where $\lambda = (\lambda_1, \ldots, \lambda_n)^T, \mu = (\mu_1, \ldots, \mu_m)^T$, $g(x) = (g_1(x), \ldots, g_n(x))^T$, $h(x) = (h_1(x), \ldots, h_m(x))^T$, and λ and μ are **dual variables**. The dual representation is inherently related to the Lagrangian approach to solving a primal problem and the latter is often termed the **Lagrangian dual problem**. The optimal value of the dual problem, denoted as d^*, satisfies,

$$d^* = \inf_{\lambda \geq 0, \mu} \sup_x L(x, \lambda, \mu).$$

For a convex optimization problem, we almost always have **strong duality**, or that the solution to primal problem and the dual problem are equivalent. For a general optimization problem, this does not necessarily hold, and the **duality gap** is the difference between the primal and the dual solutions or $p^* - d^*$. We denote the primal solution as p^* and the dual solution as d^*.

When the duality gap is negative, then **weak duality** holds. We can show $p^* \leq d^*$ or the primal solution will always be less or equal to the dual solution for our maximization problem. Note that for a minimization problem, the primal solution will always be greater than or equal to the dual solution. The proof of this can be shown with the minimax inequality.

8.2 Optimal Transport and the Monge-Kantorovich Problem

In essence, optimal transport is about moving mass from one probability distribution to another subject to some transportation cost. Common optimization, computer science, and economics problems can be written in the optimal transport framework and computationally solved. Recent developments in computational solutions algorithms have made optimal transport problems that were previously too difficult to now be solvable.

We have the problem of assigning possibly infinite number of workers and firms. Each worker works for one firm and each firm cam hire one worker. Workers and firms have heterogeneous characteristics, $x \in \mathcal{X}$ for workers and $y \in \mathcal{Y}$ for firms, and we let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$. Workers and firms are in equal mass, and we normalize the total mass of workers and firms to 1. The distribution of workers is P over $\mathcal{X}$ and the distribution firms is Q over $\mathcal{Y}$.

We define a **coupling** to determine which workers are assigned to which firms. If we had a finite number of workers and firms, we just need to count the workers of type x that are matched to firms of type y. The coupling more generally is defined as the probability measure π of the occurrence of worker-firm pairs. If $(X, Y) \sim \pi$ is a joint random pair, then $X \sim P$ and $Y \sim Q$ where $X \sim P$ means X has distribution P. The first margin of π is P and the second margin is Q.

Definition 12. The set of couplings of probability distributions P and Q is the set of probability distributions over $\mathcal{X} \times \mathcal{Y}$ with first and second margins P and Q. The set is denoted as $\mathcal{M}(P, Q)$. In other words, a probability measure π over $\mathcal{X} \times \mathcal{Y}$ is in $\mathcal{M}(P, Q)$ if and only if

$$\pi(A \times \mathcal{Y}) = P(A) \text{ and } \pi(\mathcal{X} \times B) = Q(B)$$

holds for every subset A of $\mathcal{X}$ and B of $\mathcal{Y}$. By extension, a random pair $(X, Y) \sim \pi$, where $\pi \in \mathcal{M}(P, Q)$ will also be called a coupling of P and Q.

The simplest coupling is the *independent coupling* (or sometimes called random matching). Here, we let $\pi(A \times B) = P(A)Q(B)$ so that if $(X, Y) \sim \pi$, then $X \sim P$ and $Y \sim Q$. While this coupling ensures the set $\mathcal{M}(P, Q)$ is non-empty, this also means that we have random matches between firms and workers.

We can also consider couplings (X, Y) such that $Y = T(X)$ is a deterministic function of X. This ensures that workers of type x are assigned to the same type of firm $T(x) \in \mathcal{Y}$. This type of assignment is called pure assignment or is called the Monge Coupling. The constraint on $T(\cdot)$ that ensures $(X, T(X))$ is a coupling of P and Q is equivalent to the five following conditions.

1. If $X \sim P$, then $T(X) \sim Q$.

2. Equality $P(T^{-1}(B)) = Q(B)$ holds for every subset B of $\mathcal{Y}$.
3. The coupling $\pi(x, y) = P(x)\delta(y - T(x))$ is in $\mathcal{M}(P, Q)$ where $\delta(\cdot)$ is the Dirac delta function.
4. For any $\phi \in L^1(Q)$ (or ϕ is an function that is integrable with respect to P), $E_P[\phi(T(X))] = E_Q[\phi(Y)]$.
5. When P and Q have respective densities f_p and f_Q, and when T is smooth, then the Monge-Ampère Equation holds, or $f_P(x) = |\det DT(X)| f_Q(T(x)$, which is just a multivariate change of variables.

These equivalent conditions are denoted as

$$T\#P = Q$$

where $T\#P$ is the distribution of $T(X)$ when $X \sim P$ and is called the *push-forward* probability of probability distribution P by map T. In the probability literature, this relation is also denoted as $PT^{-1} = Q$.

Outside of the Monge couplings, couplings can generally be characterized by a family of conditional probability distributions. These are Markov kernels $\pi(dy|x)$ such that $\int_{\mathcal{X}} \pi(B|x)dP(x) = Q(B), \forall B \subseteq \mathcal{Y}$.

We assume if worker x workers for firm y, this generates output $\Phi(x, y)$ which is measured in dollars. The *social planner's problem* is how to assign workers to firms to maximize the total output. This is the problem addressed in the Monge-Kantorovich Problem.

The **Monge problem** only looks at the possible pure assignments of firms to maximize the overall average surplus,

$$\max_{T(\cdot)} E_P[\Phi(X, T(X))]$$
$$\text{s.t. } T\#P$$

One possible function form for the total surplus is $\Phi(x, y) = -|x - y|$, which is a form of the negative *earth mover's distance.* If x, y are locations of the workers and firms, the total surplus is higher is the assigned worker is closely located to the her assigned firm.

The Monge Problem remained unsolved until Kantorovich's work with linear programming relaxation. Rather than assign x deterministically to work with firm $T(x)$, we can introduce randomization the assignment mechanism. So instead of maximizing over the deterministic maps $T(x)$, we can maximize over the conditional probabilities, or Markov kernels, of assigning worker x to firm y. Then we replace $\Phi(x, y)$ in the Monge Problem with $E_\pi[\Phi(x, Y)|X = x]$, where $\pi \in \mathcal{M}(P, Q)$ is coupling between probabilities P and Q that is not necessary pure. Then, we get the **Kantorovich problem**,

$$\max_{\pi \in \mathcal{M}(P,Q)} E_\pi[\Phi(X, Y)]$$

which has some difference to the Monge problem. The Kantorovich problem (1) is a linear programming problem and (2) will have a solution π. The former will lead us to a duality result and the latter is nice because the Monge problem does not always have a solution. Often optimal transport problems are set up such that the Monge solution exists and coincides with the Kantorovich solution. Then, the Kantorovich relaxation solves the Monge problem.

Theorem 13. *(Monge-Kantorovich Duality) Let $\mathcal{X}$ and $\mathcal{Y}$ be subsets of $\mathbb{R}^d$ and P, Q be probability measures on $\mathcal{X}, \mathcal{Y}$ respectively. We have $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \cup \{-\infty\}$ be a upper semi-continuous surplus function bounded from above. Then, we have*

(1) the value of the primal Monge-Kantorovich problem

$$\max_{\pi \in \mathcal{M}(P,Q)} E_\pi[\Phi(X,Y)]$$

coincides with the value of the dual

$$\inf_{u,v} E_P[u(x)] + E_Q[v(Y)]$$
$$s.t.\ u(x) + v(y) \geq \Phi(x,y)$$

where u, v are measurable and integrable functions. The inequality constraint should hold for almost every x and almost every y

(2) an optimal solution to the primal Monge-Kantorovich problem exists.

We have strong duality for this infinite dimensionality linear programming problem from (1) and (2) tells us the solution to the primal problem exists. We can interpret $u(x)$ and $v(y)$ as the equilibrium payoffs that a worker x and firm y will get at equilibrium. Then, the total surplus is the dual problem is the sum of the worker's surplus and the firm's surplus. While the theorem does not state the dual solution (u, v) exists, assume that it does for the following proposition.

Proposition 4. *If (u, v) is a solution to the dual of the Kantorovich problem, then we can redefine u and v such that they take $+\infty$ for values outside of their supports P and Q. Then,*

$$u(x) = \sup_y(\Phi(x,y) - v(y))$$
$$v(y) = \sup_x(\Phi(x,y) - u(y))$$

should hold almost surely with respect to probabilities P and Q respectively.

We can interpret $u(x)$ as the market wage of the worker x and the $v(y)$ as the indirect surplus of the firm y. We can interpret the second equation as the firm will not hire x unless hiring x yields a profit that is equal to $v(y)$. The first equation then describes the worker's problem of choosing the firm optimally. Then, our theorem acts like a welfare theorem. The primal solution, or the social planner's problem solution, will coincide with the dual solution, which is the decentralized equilibrium. We see that the decentralized solution is further both Pareto efficient and in the utilitarian sense as it optimized the total surplus $\Phi(x, y)$ over the possible matches.

9 Example

In the example, we will examine the performance of different optimizers on the well-known Rosenbrock Banana Function. Hopefully, this should build intuition on the performance of different optimizers and shed some light into the black-box optimizers that are built-in to modern statistical software programs. Please see the RMarkdown notebook for the example.

Optimization Example

Booth Math Camp (Autumn 2021)

Walter W. Zhang

July 01, 2021

In this example, we will examine the performance of different optimizers on the well-known Rosenbrock Banana Function. Hopefully, this should build intuition on the performance of different optimizers and shed some light into the black-box optimizers that are built-in to modern statistical software programs. In R, the common, off-the-shelf optimizers have been wrapped up in the `optim` function.

Contents


```
# Load Packages
require(ggplot2)
require(scatterplot3d)
require(scales)
```

We consider a 2-D Rosenbrock Banana function,

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

which has the analytic gradient,

$$\nabla f(x) = \begin{pmatrix} 2(200x_1^3 - 200x_1x_2 + x_1 - 1) \\ 200(x_2 - x_1^2) \end{pmatrix}.$$

Looking at the function, the minimum occurs at $(x_1, x_2) = (1, 1)$ and the function has a value of 0. We will examine the performance of the BFGS, Nelder-Mead, and Conjugate Gradient optimization algorithms.

Setup

```
# x is a vector with two elements
objective <- function(x)
{
```

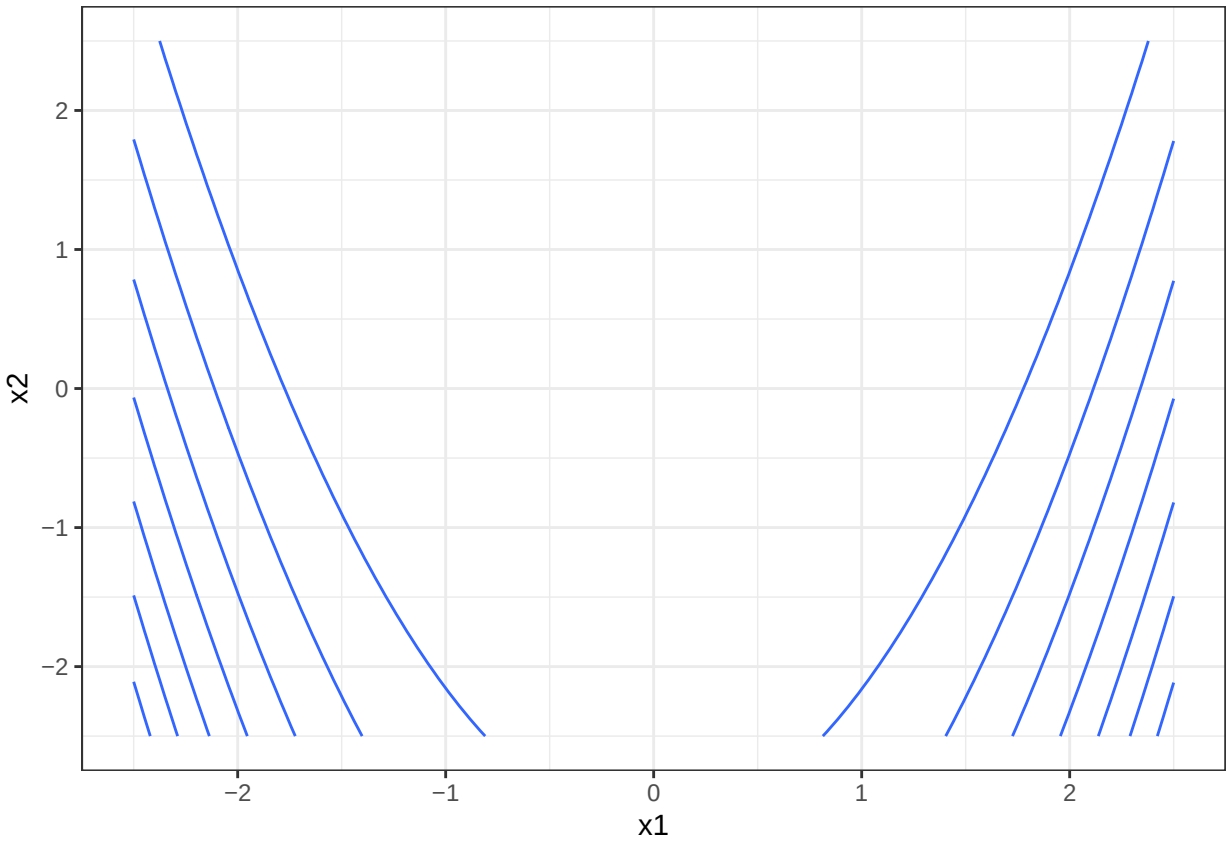
```
  obj <-  100 * (x[2] - x[1]^2 )^2 + (x[1] - 1)^2
  return(obj)
}

objective(c(1,1))
```

```
## [1] 0
```

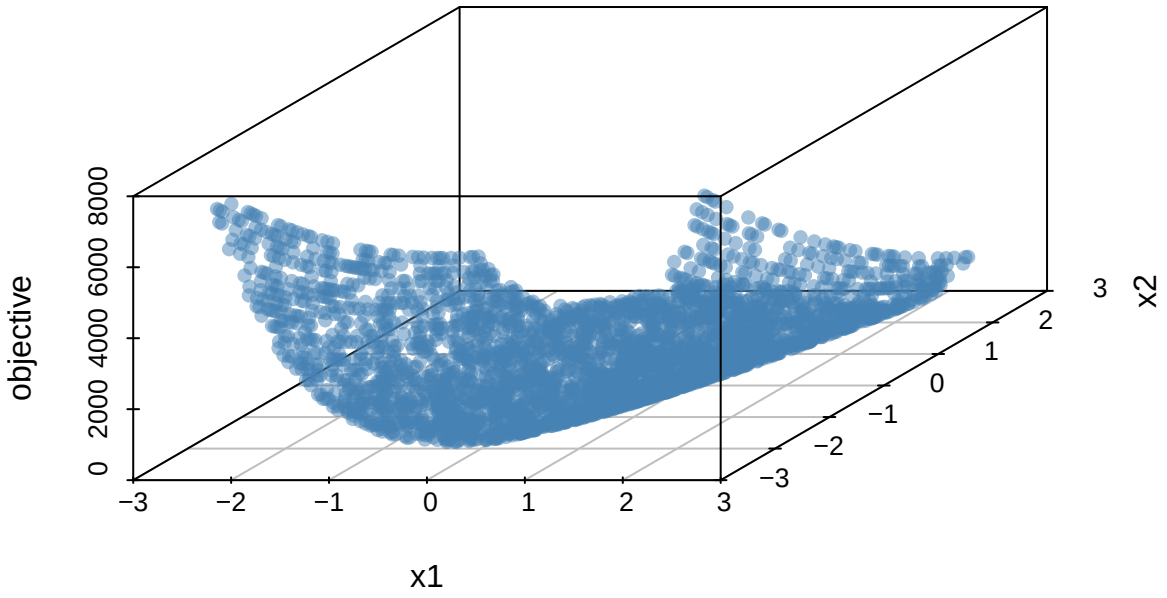
```
gradient <- function(x)
{
  grad <- c(2*(200*x[1]^3 - 200*x[1]*x[2] + x[1] - 1),
            200*(x[2]-x[1]^2))
  return(grad)
}
```

```
# Simulate the data
lower_bound <- c(-2.5)
upper_bound <- c( 2.5)
sample_num  <- 100L

set.seed(1234L)
data_mat <- expand.grid(x1 = (seq(lower_bound, upper_bound, length.out = sample_num)),
                        x2 = (seq(lower_bound, upper_bound, length.out = sample_num)))
obj_vec  <- apply(data_mat, 1, objective)
grad_vec <- apply(data_mat, 1, gradient)
```

```
# Plot the surface plot
graph_DT <- data.frame(x1 = data_mat[,1],
                       x2 = data_mat[,2],
                       objective = obj_vec)

ggplot(graph_DT, aes(x1, x2, z = objective)) +
  stat_contour() +
  xlim(lower_bound, upper_bound) +
  ylim(lower_bound, upper_bound) +
  theme_bw() +
  theme(strip.background = element_rect(colour = "gray40", fill = "aliceblue"))
```




```
# Plot a 3-D Scatter
scatterplot3d(graph_DT[sample(1:nrow(graph_DT), 3000),],
              pch = 16, color=alpha("steelblue", 0.5) )
```



The Rosenbrock function is a non-convex function and the global minimum is within a long and narrow parabolic shaped value. While finding the valley is relatively easy, pinning down the global minimum within the valley is more difficult.

BFGS

We first examine the performance of BFGS (or the Broyden–Fletcher–Goldfarb–Shanno Algorithm). The algorithm is an iterative quasi-Newton method that approximates the Hessian from the Gradient and whose necessary condition for optimality is that the gradient is zero at the optimum. From a supplied starting point or initial guess it does the following:

1. Compute the direction to move the gradient by numerically approximating the Hessian
2. Determine the acceptable step size given the direction found in the first step
3. Move in the new direction given the direction and step size
4. Update the Hessian approximation for the new point

The procedure terminates when the norm of the gradient is under some tolerance value (i.e. 10^{-6})

```
# BFGS
x0 <- c(-1, 2)

init_time <- Sys.time()
res_bfgs <- optim(x0,
                  objective,
                  gradient,
                  method  = "BFGS",
```

```
                  control = list(trace = 10))
```

```
## initial  value 104.000000
## iter  10 value 6.536201
## iter  20 value 3.223389
## iter  30 value 0.000481
## iter  40 value 0.000000
## final  value 0.000000
## converged
```

```
end_time <- Sys.time()
BFGS_time <- format(end_time - init_time, digits = 3)
```

Nelder-Mead

We then examine the Nelder-Mead performance. Nelder-Mead is a downhill simplex method that is a direct search method based on the function value. However, the procedure is *heuristic* search method and can get stuck on non-stationary points. We first denote a simplex as a $n+1$ dimensional polytope on a n dimensional surface. For the 1-D case this is a line segment, and in the 2-D case this is a triangle. The simplex then moves as the algorithm iterates. From the simplex constructed around the proposed starting value, the general approach of the procedure for minimization is as follows (use a triangle as the simplex the 2-D plane for intuition):

1. *Ordering*: Order the vertices of the simplex by their function value (rank the edge of the triangle by the function value)
2. Compute the centroid of the points excluding the vertex with highest function value
3. *Reflection*: Reflect one of the points on the simplex and see if has a smaller function value than the excluded point (move a edge of the triangle across the triangle). If it is, then replace the excluded point and form a new simplex and go to step 1. If the new point has the smallest function value for all points on the simplex go to the next step.
4. *Expansion/Extension*: Move along the direction from the centroid to the new point (move out along the triangle's centroid to a edge point). This new point is the expanded point and if the expanded point is better than the worst point on the simplex then move to the step 1. If not then go to the next step.
5. *Contraction/Reduction*: Move along the direction from the new point to the centroid (move in towards the triangle's centroid to a edge point). This new point is the contracted point and if the contracted point is better than the worst point on the simplex then move to the step 1. If not then go to the next step.
6. *Shrinkage*: Shrink the simplex by replacing the point except the best performing point (make the triangle smaller). Then move to step 1.

The algorithm terminates when the simplex updates are small enough given some tolerance value (triangle barely moves). Then the lowest point on the simplex (triangle) is returned as the optimum for the minimization problem.

Note that this approach neither uses the gradient nor the hessian explicitly. If provided, the gradient can help with the individual line searches for moving simplex vertices.

```
# Nelder-Mead
x0 <- c(-1, 2)

init_time <- Sys.time()
res_nm <- optim(x0,
                objective,
                gradient,
                method = "Nelder-Mead",
                control= list(trace = 10))
```

```
##   Nelder-Mead direct search function minimizer
## function value for initial parameters = 104.000000
##   Scaled convergence tolerance is 1.54972e-06
## Stepsize computed as 0.200000
## BUILD              3 188.200000 104.000000
## EXTENSION          5 148.000000 17.320000
## REFLECTION         7 104.000000 7.720000
## LO-REDUCTION       9 17.320000 7.720000
## HI-REDUCTION      11 13.520000 7.720000
## HI-REDUCTION      13 8.178125 6.564221
## REFLECTION        15 7.720000 6.328167
## HI-REDUCTION      17 6.564221 6.191275
## REFLECTION        19 6.328167 5.879893
## HI-REDUCTION      21 6.191275 5.879893
## REFLECTION        23 5.957893 5.729399
## EXTENSION         25 5.879893 5.378041
## REFLECTION        27 5.729399 5.287197
## EXTENSION         29 5.378041 4.420000
## LO-REDUCTION      31 5.287197 4.420000
## LO-REDUCTION      33 4.704885 4.336536
## EXTENSION         35 4.420000 3.908420
## HI-REDUCTION      37 4.336536 3.908420
## EXTENSION         39 4.204598 3.546309
## LO-REDUCTION      41 3.908420 3.546309
## REFLECTION        43 3.717900 3.470733

.....

## REFLECTION       167 0.024408 0.015836
## REFLECTION       169 0.022062 0.014837
## REFLECTION       171 0.015836 0.011379
## REFLECTION       173 0.014837 0.009836
## LO-REDUCTION     175 0.011379 0.006739
## HI-REDUCTION     177 0.009836 0.006739
## HI-REDUCTION     179 0.007256 0.006434
## EXTENSION        181 0.006739 0.004402
## HI-REDUCTION     183 0.006434 0.004402
## EXTENSION        185 0.005283 0.003210
## EXTENSION        187 0.004402 0.000739
## HI-REDUCTION     189 0.003210 0.000739
## LO-REDUCTION     191 0.002540 0.000739
## EXTENSION        193 0.001442 0.000026
## LO-REDUCTION     195 0.000739 0.000026
## HI-REDUCTION     197 0.000141 0.000026
## HI-REDUCTION     199 0.000131 0.000026
## LO-REDUCTION     201 0.000029 0.000017
## HI-REDUCTION     203 0.000026 0.000001
## HI-REDUCTION     205 0.000017 0.000001
## LO-REDUCTION     207 0.000008 0.000001
## HI-REDUCTION     209 0.000003 0.000001
## Exiting from Nelder Mead minimizer
##     211 function evaluations used
```

```
end_time <- Sys.time()
NM_time <- format(end_time - init_time, digits = 3)
```

Conjugate Gradient (CG)

The conjugate gradient method was initially developed to solve for the numerical solution of a system of linear equations, but has been adapted for unconstrained optimization. It is usually implemented as an iterative algorithm and is approximate optimization routine. At each step, the method uses conjugate gradients to determine the step direction, and it then moves along the chosen direction. The benefit of this algorithm over standard Newton methods is that it explicitly avoids matrix inversion and will be better computationally feasible for larger problems.

```
# Conjugate Gradient
x0 <- c(-1, 2)

init_time <- Sys.time()
res_cg <- optim(x0,
                objective,
                gradient,
                method = "CG",
                control= list(trace = 10))
```

```
##   Conjugate gradients function minimizer
## Method: Fletcher Reeves
## tolerance used in gradient test=3.63798e-12
## 0 1 104.000000
## parameters   -1.00000    2.00000
## ***** i> 1 8 48.945435
## parameters   -1.12672    1.93600
##  i> 2 10 8.462664
## parameters   -1.40325    1.80518
## ***** i> 3 17 6.082506
## parameters   -1.37227    1.81568
##  i> 4 19 5.571783
## parameters   -1.33984    1.82630
## ***** i> 5 26 5.528302
## parameters   -1.34368    1.82431
##  i> 6 28 5.515486
## parameters   -1.34820    1.82145
## **** i< 7 34 5.512028
## parameters   -1.34609    1.82084
##  i> 8 36 5.502421
## parameters   -1.34333    1.81513
## **** i< 9 42 5.499507
## parameters   -1.34414    1.81344
##  i> 10 44 5.489721
## parameters   -1.34270    1.80668

.....

## **** i< 89 362 4.989181
## parameters   -1.23248    1.52623
##  i> 90 364 4.980618
## parameters   -1.23148    1.51989
## **** i< 91 370 4.976431
## parameters   -1.22923    1.51935
##  i> 92 372 4.967880
## parameters   -1.22560    1.51417
## **** i< 93 378 4.963809
## parameters   -1.22678    1.51224
##  i> 94 380 4.955307
## parameters   -1.22581    1.50590
```

```
## **** i< 95 386 4.951062
## parameters   -1.22354    1.50538
##  i> 96 388 4.942570
## parameters   -1.21988    1.50023
## **** i< 97 394 4.938438
## parameters   -1.22106    1.49829
##  i> 98 396 4.929998
## parameters   -1.22012    1.49195
## **** i< 99 402 4.925694
## parameters   -1.21784    1.49143
##  i> 100 404 4.917261
## parameters   -1.21413    1.48631
```

```
end_time <- Sys.time()
CG_time <- format(end_time - init_time, digits = 3)
```

Comparisons

We compare the three routines by optimum value, function value, and run time.

```
res_DT <- data.frame(Name        = c("BFGS", "NM", "CG"),
                     Objective   = c(res_bfgs$value, res_nm$value, res_cg$value),
                     Convergence = c(res_bfgs$convergence, res_nm$convergence, res_cg$convergence),
                     X1_opt      = c(res_bfgs$par[1], res_nm$par[1], res_cg$par[1]),
                     X2_opt      = c(res_bfgs$par[2], res_nm$par[2], res_cg$par[2]),
                     Time        = c(BFGS_time, NM_time, CG_time))

res_DT
```

```
##   Name    Objective Convergence     X1_opt    X2_opt       Time
## 1 BFGS 3.615234e-18           0  1.0000000 1.0000000 0.003 secs
## 2   NM 4.568377e-07           0  0.9997026 0.9994661 0.002 secs
## 3   CG 4.917261e+00           1 -1.2178392 1.4914286 0.002 secs
```

We see that BFGS and Nelder-Mead basically converged to the optimum while the Conjugate Gradient method failed to converge. Since Nelder-Mead is a heuristic optimization algorithm, we see that the optimal values are further away than BFGS.

Remarks

- Generally, Nelder-Mead and BFGS are used for applied researchers because they support constrained optimization
- Feeding in the analytically gradient function often will drastically speed up the optimization routine. Try running the optimizers without the gradient specified. Some routines will compute the numerical gradient and thus will be slower and be subject to numerical approximation noise
- The choice of the starting value can really help the optimizer along. Try choosing starting values that are far from the optimal value and see how long the routines will take.
- You can mix optimizers. If you want to run an MLE estimator, you can speed things along by first running Nelder-Mead. Then take the Nelder-Mead estimates and plug it into BFGS and ask for the numerical Hessian (so you can compute SE). This will probably be faster than just running BFGS from the start.
- Lastly, for many complicated models it is often difficult to determine whether you reached a local or global optima. Ideally your final optimal value should make intuitive sense given the application. If you have lots of computing resources to spare, you can also run many versions of the optimizer for a grid of starting point and choose the best optimal point over the set of optimal points.

Part IV

Dynamic Programming

This section provides an introduction of dynamic programming and a background for the first year macroeconomics, operations, and marketing courses. Generally, these concepts will be widely used in a structural economics course. The companion RMarkdown notebook provides an introduction of the computation aspect by working through the Rust (1987) dynamic discrete choice and optimal renewal problem.

10 Fundamentals

Before we get started with dynamic programming, we will cover some fundamentals to review the tool set needed to solve these types of problems.

10.1 Fixed Points

In mathematical economics, we usually defer the existence of an equilibrium to a fixed point theorem. As long as we can show a fixed point exists, then we can design an algorithm that converges to the fixed point in our estimation. There are three main families of fixed point theorems in economics.

1. The *metric* approach with *Banach's* fixed point theorem. Contraction mapping in a metric space yields a unique fixed point. Convex analysis and linear programming fall here.
2. The *order-theoretic* approach with *Tarski's* fixed point theorem. An isotone mapping in a lattice contains a nonempty set of fixed points that are itself a lattice. Supermodularity and isotonicity fall here.
3. The *topological* approach with *Schauder's* fixed point theorem. A continuous mapping on a convex compact space yields a nonempty, closed set of fixed points. Existence of mixed equilibrium Nash strategies fall here.

The first approach contains the bulk of applied research. Generally, the first two approaches have been used computationally. The first uses an convex optimization framework and the second uses a Nash equilibrium setup in a supermodular game.

In dynamic programming, the fixed point theorem guarantees an existence of an optimal solution given some conditions. We will often leverage the convex optimization framework to solve the dynamic program computationally.

More formally, a fixed point is a point that maps to itself for a given function f. In the simplest case, $x \in \mathbb{R}$ is a fixed point if continuous function f has the property that $f(x) = x$. Naturally, further compositions of function evaluated at x yields the fixed point ($f(f(x)) = x$). We generalize this concept with the following.

Theorem 14. *(Brouwer's fixed-point theorem) Any continuous function f mapping a compact, convex set to itself will have a a point x such that $f(x) = x$.*

There are two classic examples of Brouwer's Theorem.

- (2D case) Take out a map of the country that you reside in. There will always be a point on the map that represents the exact place of the map in the country.
- (3D case) Consider stirring a tea in a teacup. No matter how long you stir the tea, there will always be a point in the tea cup that was at the original spot before you started stirring.

Generally, a nonlinear equation of form $f(x) = 0$ can be rewritten as $g(x) = x$ where x is the fixed point of some function $g(x)$. We further can solve the for the fixed point using a fixed-point iteration algorithm. To build intuition, we first consider functions on $\mathbb{R}$.

Theorem 15. *Let g be a continuous function on $[a, b]$. If $g(x) \in [a, b]$ and for each $x \in [a, b]$, then g has a fixed point in $[a, b]$. Furthermore, if g is differentiable on (a, b) and there exists a constant $k < 1$ such that $|g'(x)| < k$, then g has a unique fixed point in $[a, b]$.*

Algorithm 3 Fixed point iteration

We let g be a continuous function on $[a, b]$. The following algorithm produces a value $x^* \in (a, b)$ that is a solution to $g(x) = x$.

initialization: Guess $x_0 \in [a, b]$. Choose $K \in \mathbb{N}$ and some tolerance level *tol*
for $k = 0, 1, \ldots, K$:

1. $x_{k+1} = g(x_k)$
2. **if** $(|x_{k+1} - x_k| < tol)$ then
   (a) **break** the for loop iteration
   (b) **end**
3. **end**

define $x^* = x_{k+1}$
return x^*

Exercise 8. Why do we need $|g'(x)| < k$ for $k < 1$? Appeal to Taylor's Theorem and note that for $e_k = x_k - x^*$ we have $e_{k+1} \approx g'(x^*)e_k$. What happens to the algorithm's performance as $|g'(x)| \to 1$?

Theorem 16. *We let g be a continuous function on $[a, b]$. Then if $g(x) \in [a, b]$ for each $x \in [a, b]$ and there exists some constant $k < 1$ such that $|g'(x)| \leq k, \forall x \in (a, b)$, then the sequence $\{x_k\}_{k=0}^{\infty}$ produced from the algorithm converges to the unique fixed point $x^* \in [a, b]$ for any initial guess $x_0 \in [a, b]$.*

Example 24. We want to solve $\cos(x) - x = 0$. We can to compute the fixed point of $g(x) = \cos(x)$ in the interval $[0, 1]$. We know $\cos(x) : [0, 1] \mapsto [0, 1]$, $|cos(X)| \leq 1$, and $\cos(x)$ is continuous so $g(x)$ has a fixed point in $[0, 1]$. Further $|g'(x)| = |-\sin(x)| \leq 1$ on $[0, 1]$ so the fixed point is unique. Then, we can apply the fixed point iteration algorithm for some initial point $x_0 \in [0, 1]$ to achieve the fixed point.

Exercise 9. Code the previous example up. Show that number of time lengths it takes to converge to the fixed point changes with the starting value. For example, starting at the fixed point itself should lead the algorithm to terminate after one step.

10.2 Gradient-based Optimization

We consider ways to minimize some continuous loss function $J : \mathbb{R}^d \to \mathbb{R}$ whose gradient is well defined. These gradient based methods are often used in complicated optimization problems that occur in dynamic programming and in reinforcement learning. We first define the **gradient flow** equation

$$\frac{du}{dt} = -DJ(u), \quad u(0) = u_0$$

where $u(t)$ is a function of time period t.

We know that

$$\frac{d}{dt}(J(u)) = \langle DJ(u), \frac{du}{dt} \rangle = -|DJ(u)|^2.$$

Since the time derivative of $u(t)$ gives the tangent to the trajectory, moving in the negative gradient of $J(u)$ will lead $J(u(t))$ to be non-increasing as a function of time. Further $J(u(t))$ will decrease until u is at a non-critical point of $J(\cdot)$, at which the gradient is 0.

To turn the gradient flow into an optimization algorithm, we can discretize it by using Euler's method with a variable time-step α_l to obtain **gradient descent** or

$$u_{l+1} = u_l - \alpha_l DJ(u_l)$$

where l represents the discrete time interval. Note that our choice of the variable time-step α_l can speed up or slow down the convergence to the critical point. Ideally, we want to choose α_l such that we jump as close as possible to the critical point in one step.

Lastly, we can consider **stochastic gradient descent**. We now let

$$J(u) = \int_B F(u, z)\eta(z)dz$$

where $B \subseteq \mathbb{R}^q$ and η is the PDF of a random variable $z \in B$. Then, the goal is to determine

$$u^* = \arg\min_{u \in \mathbb{R}^d} J(u).$$

Stochastic gradient descent tries to solve this problem when explicit evaluations of $J(u)$ and $DJ(u)$ are not possible due to its integration over B. However, we assume $D_u F(u, z)$ can be evaluated. Then, the algorithm becomes

$$u_{l+1} = u_l - \alpha_l D_u F(u_l, z_l)$$

where $z_l \in B$ is drawn i.i.d. from the PDF η. Recently, stochastic gradient descent methods have been used for problems where standard gradient descent methods are computationally feasible. The former is computationally less taxing (and requires less memory) and its randomness allows escaping of local minima and faster traversing of saddle point neighborhoods.

Exercise 10. Consider a strictly convex objective function. Will gradient descent and stochastic gradient descent methods always achieve the global minima on a convex set? Assuming no computational feasibility issues, will stochastic gradient descent be better performing than gradient descent here?

11 Introduction

To build an intuition for dynamic programming, we will walk through different set ups to the classic cake eating problem. Suppose after a long Stokey lecture, you acquire a whole cake of size W_1 from the Booth Cafe (or the Divinity School cafe or Plein Air) to stress eat while solving her latest problem set. For each time period $t = 1, \ldots, T$, you can eat some of the cake and will save the rest, and the problem set is due at time $T + 1$. You want to finish eating the whole cake by the time you hand in the problem set. After all, you cannot stress eat without the stress!

We let c_t be the consumption in period t and $u(c_t)$ be the **flow utility** from c_t. Note that preferences are stationary ($u(c_t)$ is not indexed by time), $u(\cdot)$ is real-valued differentiable, strictly increasing, and strictly concave, and that $\lim_{c \to 0} u'(c) \to \infty$ (infinite marginal utility at no consumption). Then, your lifetime

utility in this period is

$$\sum_{t-1}^{T} \beta^{(t-1)} u(c_t)$$

where $0 \leq \beta \leq 1$ is the **discount factor**. We further assume that the cake neither depreciates nor grows over time. Then, the cake's evolution, or transition formula, is

$$W_{t+1} = W_t - c_t, \ \forall t = 1, \ldots, T. \tag{1}$$

We want to find our optimal path of consumption in this period $\{c_t^*\}_{t=1}^T$.

11.1 Brute-force Approach

We can brute force the finite dimensional optimization problem by taking the **sequence problem** approach. We have the program

$$\max_{\{c_t\}_{t=1}^T, \{W_t\}_{t=2}^{T+1}} \sum_{t=1}^{T} \beta^{(t-1)} u(c_t) \tag{2}$$

subject to the cake transition formula from Equation 1. We further assume that there are *non-negativity constraints* given by $c_t \geq 0$ and $W_t \geq 0$ and W_1 is the size of the initial cake and is given. Here, we can think of W_1 as a *boundary condition* of our problem.

We can collapse the transitions equations to get the *flow constraint* by writing

$$W_{T+1} = W_T - c_T = -\sum_{t=1}^{T} c_t + W_1 \tag{3}$$

and the non-negativity constraints can be written as $c_t \geq 0, \forall t \in \{1, \ldots, T\}$ and $W_{T+1} \geq 0$ (*resource constraint*). We see that in our program, we have a concave and continuous objective function and a compact constraint set and as a result we will have a solution to our program. Setting λ as the Lagrange Multiplier for the flow constraint ($W_1 - \sum_{t=1}^T c_t - W_{T+1} = 0$) and ϕ as the multiplier on the resource constraint ($W_{T+1} \geq 0$), we have

$$\mathcal{L} = \left(\sum_{t=1}^{T} \beta^{(t-1)} u(c_t) \right) + \lambda \left(W_1 - W_{T+1} - \sum_{t=1}^{T} c_t \right) + \phi \left(W_{T+1} \right)$$

and the FOC are

$$\begin{aligned} \beta^{t-1} u'(c_t) &= \lambda \\ \lambda &= \phi. \end{aligned}$$

Since the MU of consumption at $c_t = 0$ is infinity, we can ignore the non-negativity constraint on c_t. Since

the FOC to c_t holds for all t, we see that

$$\beta^{t-1}u'(c_t) = \lambda$$
$$\beta^{t}u'(c_{t+1}) = \lambda$$

where we moved the time index from t to $t+1$ to get the second equation. Equating the two equations, we attain the **Euler equation**,

$$u'(c_t) = \beta u'(c_{t+1}) \tag{4}$$

which links consumption across any two period. Note that this is a *necessary condition for optimality for any t* and if it does not hold then we can always do better by adjusting our choice of c_t and c_{t+1}. However, this is condition is not a *sufficient condition* as deviations that occur more than one period away will not covered by a single Euler equation. We can combine the Euler equations to cover these by

$$u'(c_t) = \beta^2 u'(c_{t+2})$$

and so on. Thus, for a finite horizon problem, the Euler equation will rule out deviations from a candidate solution that satisfies the equations. But, this is still not a sufficient condition as a candidate solution can have $W_T > c_T$ so there is cake left on the table that is never eaten and thus the solution is not efficient. Hence, for an efficient solution, the non-negativity constraint must bind for W_{T+1} (or $W_{T+1} = 0$). With the initial condition (W_1 is given), the terminal condition ($W_{T+1} = 0$), and the set of $(T-1)$ Euler equations the optimal path of consumption will be pinned down.

We denote a solution to problem as $V_T(W_1)$ where T is the horizon of the problem and W_1 is the initial size of the cake. We further define $V_T(W_1)$ as the **value function**. Note that here an small increase in the size of the cake will lead the lifetime utility to increase of the marginal utility, $u'(c_t)$, in *any* period t, or

$$V_T'(W_1) = \lambda = \beta^{t-1}u'(c_t),\ \forall t = \{1, \ldots, T\}.$$

The parallel here is a income shock to a consumer that can choose to allocate it across many discrete goods.

11.2 Dynamic Programming Approach

Before we introduce the dynamic programming approach, we add a period 0 to the problem and set the initial cake size to W_0. The dynamic programming approach rewrites the finite horizon problem as two-period problem by rewriting the objective function. We now consider the program,

$$\max_{c_0} u(c_0) + \beta V_T(W_1) \tag{5}$$

where $W_1 = W_0 - c_0$ and W_0 is given.

Note that we now choose the level of consumption at time 0, or c_0, and cake size at time period 1, or W_1. Then with these two initial values, the value of the problem for the rest of the time periods is given by $V_T(W_1)$. In the dynamic programming approach, the $V_T(W_1)$ function fully summarizes optimal behavior from period 1 to period T. Note that it does not matter how the cake is consumed after period 0, as long as it yields a indirect utility level of $V_T(W_1)$. We denote this as the **principle of optimality**.

Assuming that $V_T(W_1)$ is differentiable, we see that the FOC is

$$u'(c_0) = \beta V_T'(W_1)$$

which says that the MU of consumption is related to the discounted derivative of the value function. From our results from the brute force approach, we know

$$V_T'(W_1) = u'(c_1) = \beta^t u'(c_{t+1}),\ \forall t = 1, \ldots, T-1$$

and combining the two results, we get

$$u'(c_t) = \beta u'(c_{t+1}),\ \forall t = 1, \ldots, T-1$$

which is our necessary condition for an optimal solution. Since our dynamic programming approach nests the Euler equations, the two approaches will yield the same results. Generally, when $u(c)$ is strictly concave, the two solutions will be identical in the two different approaches.

Note that we solved the problem assuming that we know $V_T(W_1)$. We can solve for it recursively by first solving the single period problem to get $V_1(W_1)$. Then, we can solve Equation 5 to construct $V_2(W_1)$, and so on. We show this in the following example.

Example 25. Recall that T represents the terminal period. Suppose $u(c) = \ln(c)$ and $T = 1$. Then $V_1(W_1) = \ln(W_1)$.

Then for $T = 2$, the FOC from Equation 2 yields

$$\frac{1}{c_1} = \beta \frac{1}{c_2}$$

with the resource constraint

$$W_1 = c_1 + c_2.$$

Together they give us,

$$c_1 = \frac{W_1}{1+\beta}, \quad c_2 = \frac{\beta W_1}{1+\beta}$$

and we can solve for the value function of the two-period problem

$$V_2(W_1) = \ln(c_1) + \beta \ln(c_2) = \aleph_2 + \beth_2 \ln(W_1) \tag{6}$$

where

$$\aleph_2 = \ln(\frac{1}{1+\beta}) + \beta \ln(\frac{\beta}{1+\beta}), \quad \beth_2 = 1 + \beta.$$

Note that Equation 6 does not contain the max operator because we are evaluating it at its optimal values.

We now solve out the $T = 3$ value function, $V_3(W_1) = \max_{W_2} \ln(W_1 - W_2) + \beta V_2(W_2)$, where the choice variable the size of the cake in the second period. We want to find the optimal levels of consumption c_t for the three periods as well as the form for our value function. Taking the first order condition of $V_3(W_1)$

to W_2, we get

$$\frac{1}{W_1 - W_2} = \beta V_2'(W_2)$$

and we have resource constraints

$$W_1 = c_1 + c_2 + c_3$$
$$W_2 = c_2 + c_3$$

which tells us that $c_1 = W_1 - W_2$. Plugging this to the FOC, we get

$$\frac{1}{c_1} = \beta V_2'(W_2).$$

Recall that we had $V_2(W_1) = \aleph_2 + \beth_2 \ln(W_1)$ for the two period problem. We then replace W_1 with W_2 for this $T = 2$ value function. Now $V_2(W_2)$ says we have an initial cake size W_2 and we have *two* periods to eat it. Note that the subscript to the value function is for how many time periods away from the terminal period we are at and subscript to the cake size is the current time period.

Since we have that $V_2(W_2) = \aleph_2 + \beth_2 \ln(W_2)$, so the partial derivative to W_2 is

$$V_2'(W_2) = \beth_2 \frac{1}{W_2} = (1+\beta)\frac{1}{W_2} = (1+\beta)\frac{1}{(1+\beta)c_2} = \frac{1}{c_2}$$

where we plugged in the solution to the $T = 2$ problem with the time periods shifted up by one ($c_2 = \frac{W_2}{1+\beta}$ and $c_3 = \frac{\beta W_2}{1+\beta}$). Then, the FOC becomes

$$\frac{1}{c_1} = \beta V_2'(W_2) = \beta \frac{1}{c_2}.$$

From the FOC of the $T = 2$ problem, we can shift the time index by one (as we did before when we used our $T = 2$ solution) and attain

$$\frac{1}{c_2} = \beta \frac{1}{c_3}.$$

Lastly, we recall the resource constraint of the problem,

$$c_1 + c_2 + c_3 = W_1.$$

The two FOC equations (or Euler equations) with the resource constraint gives us three equations for our problem with three unknowns (c_1, c_2, c_3) that we can solve as a function of W_1.

First, we see that the first two FOCs yield $c_2 = \beta c_1, c_3 = \beta c_2$ respectively, which combined yields $c_3 = \beta^2 c_1$. Then plugging these equations into the resource constraint, we see that

$$\begin{aligned} c_1 + \beta c_1 + \beta^2 c_1 &= W_1 \\ c_1(1 + \beta + \beta^2) &= W_1 \\ c_1 &= \frac{W_1}{(1+\beta+\beta^2)}. \end{aligned}$$

Plugging in our solved optimal value for c_1 into the two FOCs, we get

$$c_2 = \beta c_1 = \frac{\beta}{1+\beta+\beta^2} W_1$$
$$c_3 = \beta^2 c_1 = \frac{\beta^2}{1+\beta+\beta^2} W_1.$$

Now, we just need to find the form of the value function. We guess that for parameters (or constants values) $\aleph_3$ and $\beth_3$, the value function takes form

$$V_3(W_1) = \aleph_3 + \beth_3 \ln(W_1) = \ln(W_1 - W_2) + \beta V_2(W_2)$$

where we know $V_2(W_2) = \ln(c_2) + \beta \ln(c_3) = \aleph_2 + \beth_2 \ln(W_2)$ from before. Then, we have that

$$V_3(W_1) = \aleph_3 + \beth_3 \ln(W_1) = \ln(c_1) + \beta \ln(c_2) + \beta^2 \ln(c_3)$$

and plugging in our optimal values for c_1, c_2, c_3 we get

$$\begin{aligned}
\aleph_3 + \beth_3 \ln(W_1) &= \ln\left(\frac{W_1}{1+\beta+\beta^2}\right) + \beta \ln\left(\frac{\beta W_1}{1+\beta+\beta^2}\right) + \beta^2 \ln\left(\frac{\beta^2 W_1}{1+\beta+\beta^2}\right) \\
&= \ln(W_1) + \beta \ln(W_1) + \beta^2 \ln(W_1) \\
&\quad + \ln\left(\frac{1}{1+\beta+\beta^2}\right) + \beta \ln\left(\frac{\beta}{1+\beta+\beta^2}\right) + \beta^2 \ln\left(\frac{\beta^2}{1+\beta+\beta^2}\right)
\end{aligned}$$

Equating the $\ln(W_1)$ terms, we have that

$$\beth_3 = 1 + \beta + \beta^2$$

and equating the terms without $\ln(W_1)$, we have that

$$\aleph_3 = \ln\left(\frac{1}{1+\beta+\beta^2}\right) + \beta \ln\left(\frac{\beta}{1+\beta+\beta^2}\right) + \beta^2 \ln\left(\frac{\beta^2}{1+\beta+\beta^2}\right).$$

Thus, our solution to the problem is the optimal levels of consumption c_1, c_2, c_3 and the value function parameters $\aleph_3, \beth_3$ given our specified functional form of the value function.

Exercise 11. Verify our solution to the $T = 3$ value function solution using the brute force approach.

Exercise 12. Do you see a pattern in the solutions to optimal levels of consumption $\{c_t^*\}_{t=1}^T$ and value function parameters as T increases from the $T = 2$ and $T = 3$ solutions? Make a guess for the solution to the $T = 4$ problem and validate it by solving it out using the value function approach.

11.3 Dynamic Programming Extensions

11.3.1 Infinite Horizon Problem

We now extend our finite horizon problem to an infinite horizon problem. The program becomes

$$\max_{\{c_t\}_{t=1}^{\infty}\{W_t\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^t u(c_t)$$

with flow equation

$$W_{t+1} = W_t - c_t, \ \forall t = 1, 2, \ldots.$$

For this type of problem, we construct the value function as

$$V(W) = \max_{c \in [0,W]} u(c) + \beta V(W - c)$$

for all W. Here, we see that $u(c)$ is the utility of consuming c units in the current period and $V(W)$ is the value function of the infinite horizon problem starting at W. Thus, $V(W - c)$ represents the subsequent value function given that c was consumed today. We denote the next time period with primes, so $W' = W - c$ is the next period's cake size.

We denote the **state variable** as the size of the cake (W), which is given to us at the beginning of each problem and in this example is the initial cake size. The state variable *summarizes all of the information from the past* that is needed for the forward-looking program. The **control variable** is denoted as the variable chosen to solve the program, and here it is the level of consumption in each period, or c, and it lies on a compact set. The **transition equation** dictates next period's state variable given today's state variable and control variable,

$$W' = W - c.$$

We thus can reformulate the problem using the transition equation so we choose tomorrow's state variable instead of today's consumption level,

$$V(W) = \max_{W' \in [0,W]} u(W - W') + \beta V(W') \tag{7}$$

for all W. Equation 7 is a **functional equation** or **Bellman Equation**, where the unknown in the Bellman equation is the value function itself. Since we do not have terminal period to backward induct and derive the value function, we must rely on a *fixed point equation* or restriction since $V(W)$ appears on both sides of Equation 7.

Further, note that there's no time indicator in the Bellman Equation in Equation 7, so we can represent all our relations *invariant of time*. This is the **stationarity** property of the infinite horizon Bellman Equation. Stationarity is needed for us to leverage a fixed point theorem to show the existence of a value function that solves the program.

We first assume that such a solution exists and delay the discussion about its existence for later. The FOC of Equation 7 is

$$u'(c) = \beta V'(W').$$

Further assuming the value function is differentiable, we see that $V'(W) = u'(c)$ as from before. Since this holds for all W, it must hold in the subsequent time period, so

$$V'(W') = u'(c')$$

and we can combine the equations to obtain

$$u'(c) = \beta u'(c')$$

which looks like the Euler equation from before (Equation 4). Just like we had before, this Euler equation is necessary condition for an optimal solution for all W.

The relation from the level of consumption today and the next period cake size (the control variables in the two different formulations) to the size of the cake today (the state variable) is given by the **policy function**,

$$\begin{aligned} c &= \phi(W) \\ W' &= \psi(W) \equiv W - \phi(W). \end{aligned}$$

Substituting these values in to the Euler equation, we attain

$$u'(\phi(W)) = \beta u'(\phi(W - \phi(W))), \ \forall W.$$

Policy functions are often used in applied work because they map the state variables to actions. When either are observable to a researcher, they can be used to estimate the parameters of a model.

Example 26. (Guess and verify) We solve the infinite horizon problem while supposing $u(c) = \ln(c)$. From the solution to the finite horizon problem, we form the ansatz

$$V(W) = A + B\ln(W), \ \forall W$$

and we have reduced the dimensionality of the problem to just two parameters, A and B. We then try to find values of A and B that satisfy the functional equation. Plugging in our ansatz, we obtain

$$A + B\ln(W) = \max_{W'} \ln(W - W') + \beta(A + B\ln(W')), \ \forall W. \tag{8}$$

Solving for the FOC, we attain

$$W' = \psi(W) = \frac{\beta B}{1 + \beta B} W,$$

which we can plug into Equation 8 to get

$$A + B\ln(W) = \ln(\frac{W}{1 + \beta B}) + \beta\left(A + B\ln(\frac{\beta B W}{1 + \beta B})\right), \ \forall W.$$

Equating the terms with $\ln(W)$ and since the function equation holds for all W, we attain

$$B = \frac{1}{1-\beta}.$$

Similarly, we can use the equation to solve for A and then our ansatz is a solution to the function equation with our solved values of (A, B). Given our solution and from $u'(c) = V'(W)$ which we got from the differentiability of the value function, we see that $c = (1-\beta)W$ and $W' = \beta W$ (which in turn leads to $c = \frac{(1-\beta)}{\beta}W'$) so the optimal policy tells us to save a constant fraction of the cake and eat the remaining portion each period.

Exercise 13. Solve out the term for A that makes our ansatz satisfy the functional equation in the previous example's equations.

11.3.2 Uncertainty

We can add uncertainty to the problem by adding taste shocks. Adding uncertainty to the dynamic programming framework is generally straightforward if the shocks take values in a finite or countable set. In our cake eating example, we suppose the consumption utility is now

$$\epsilon u(c)$$

where ϵ is a random variable and $u(c)$ is a strictly increasing and strictly concave function. We still assume the initial cake size is W.

With uncertainty, we need to be careful formulating the problem as we need to determine if the agent can observe the taste shock when making decisions at different time periods. In our example, the agent knows the current taste shock value for contemporary decisions but the agent needs to form expectations of future values of ϵ.

We let the taste shocks take only two possible values $\epsilon \in \{\epsilon_h, \epsilon_l\}$ where $\epsilon_h > \epsilon_l > 0$. We assume the taste shock follows a **first-order Markov process**, so the probability of getting a specific ϵ in the current period *only* depends on the value of ϵ in the previous period. We define π_{ij} to be the probability that the value of ϵ goes from state i in the current period to state j in the next period. Then, in our example $\pi_{lh} \equiv Pr(\epsilon' = \epsilon_h | \epsilon = \epsilon_l)$ where ϵ' is the next period value of ϵ. We can construct Π, which is a 2×2 matrix of π_{ij}, and is denoted as the **transition matrix**.

We then rewrite the Bellman Equation as

$$V(W, \epsilon) = \max_{W'} \epsilon u(W - W') + \beta E_{\epsilon'|\epsilon}[V(W', \epsilon')], \ \forall W, \epsilon$$

where $W' = W - c$ is defined as from before. The conditional expectation $E_{\epsilon'|\epsilon}[V(W', \epsilon')]$ is given over Π and can be computed. Then, our FOC is

$$\epsilon u'(W - W') = \beta E_{\epsilon'|\epsilon}[V_1(W', \epsilon')] \ \forall W, \epsilon$$

where $V_1(W', \epsilon')$ represents the partial derivative of $V(W', \epsilon')$ to W' or $\frac{\partial V(W', \epsilon')}{\partial W'}$.We can use the functional

equation to solve for the marginal value of the cake $V_1(W', \epsilon')$, and we attain

$$\epsilon u'(W - W') = \beta E_{\epsilon'|\epsilon}[\epsilon' u'(W' - W'')] \tag{9}$$

which is the *stochastic Euler equation* for our problem. The optimal policy function is then

$$W' = \psi(W, \epsilon)$$

which lets us rewrite the Equation 9 as

$$\epsilon u'(W - \psi(W, \epsilon)) = \beta E_{\epsilon'|\epsilon}[\epsilon u'(\psi(W, \epsilon) - \psi(\psi(W, \epsilon), \epsilon')].$$

Note that since ϵ' and c' depend on the realized value of ϵ, we cannot split the expectation on the right hand side of Equation 9 into two pieces.

11.3.3 Discrete Choice

We can also analyze a discrete choice problem in this scenario. Now suppose you do not like eating every period and decide to eat the whole cake in one time period. We also allow the cake the grow or depreciate at rate ρ.

Under this framework, the problem becomes a *dynamic, stochastic discrete choice problem* and is under the class of problems called **optimal stopping problems**. Other common optimal stopping problems are when deciding when workers stop working, students to stop learning and go to the workforce, and when durable goods are adopted by consumers.

We let $V^E(W, \epsilon)$ and $V^N(W, \epsilon)$ be the values of eating size W cake now (E) or waiting (N) given the current taste shock $\epsilon \in \{\epsilon_h, \epsilon_l\}$. $V^E(W, \epsilon)$ and $V^N(W, \epsilon)$ are also called the **choice-specific value functions**. Then, we have that

$$\begin{aligned} V^E(W, \epsilon) &= \epsilon u(W) \\ V^N(W, \epsilon) &= \beta E_{\epsilon'|\epsilon}[V(\rho W, \epsilon')] \end{aligned}$$

where

$$V(W, \epsilon) = \max\{V^E(W, \epsilon), V^N(W, \epsilon)\}, \ \forall W, \epsilon.$$

Here $\epsilon u(W)$ is the direct flow utility from eating the whole cake. Once the cake has been eaten, then the problem terminates, so $V^E(W, \epsilon)$ is just a one-period return. Alternatively, if you wait, then there is no current consumption utility and the next period cake is of size ρW. Since tastes are stochastic, the you need to take expectations of the future taste shocks ϵ'. Similarly, you face the same problem of choosing to wait or consume the next period so the value of having the cake is $V(W, \epsilon)$ which it the value from maximizing over choosing waiting or eating. The future is discounted by β and the cake will grow or deteriorate at rate ρ.

However, if $\rho \leq 1$, then the cake does not grow and the you will always consume when you get a realization of ϵ_h. Thus, $V(W, \epsilon_h) = V^E(W, \epsilon_h) = \epsilon_h u(W), \forall W$.

In contrast, in the low state, ϵ_l, then if β and ρ are sufficiently close to 1, then there is not much cost

incurred from delaying the decision to consumer. If π_{lh} is also close to 1, then it is likely the next period will have a draw of ϵ_h. Then, it is not optimal to eat the cake in the state (W, ϵ_l) and you would choose to delay the choice to the next state.

11.4 General Formulation

Now that we have built intuition from the cake-eating problem, we will generalize the dynamic programming approach.

11.4.1 Non-stochastic Case

We first consider an infinite horizon problem where the agent has a payoff function in period t that is denoted as $\tilde{\sigma}(s_t, c_t)$. Here s_t is the state vector and c_t is the control vector. These vectors are just the multivariate parallel to their single dimensional variables in the cake-eating example. The state vector next period is given by the control and state vectors this period and the transition equation

$$s_{t+1} = \tau(s_t, c_t).$$

Exercise 14. Since the next period's state vector depends on the current period state vector, does this rule out dependence of the past (i.e. variables from two periods ago)? If not, how can we include them in our general framework?

The state vector is pinned down by preferences and the transition equation, and the researcher can choose different representations of the control variables. We let $c \in C$ and $s \in S$, and sometimes the allowable control variables are dependent on the state or $C(s)$. We further assume the payoff $\tilde{\sigma}(s, c)$ is bounded for $(s, c) \in S \times C$. Note that neither the payoff nor the transition equation explicitly depend on time t. While the problem is dynamic, for a given state, the optimal choice of the agent will be same regardless of when she decides to optimize. The means the optimal choice is not related to a specific time t. Stationarity thus lets us solve the infinite horizon problem by removing time t subscripts from the problem.

Exercise 15. Can we use utility function that explicitly depend on time, like $u(c_t) = c_t - t$, in this dynamic programming setup?

Assuming a discount rate $0 < \beta < 1$, the agent's payoffs over the infinite horizon are

$$\sum_{t=0}^{\infty} \beta^t \tilde{\sigma}(s_t, c_t). \tag{10}$$

The dynamic programming approach would set up the value function

$$V(s) = \max_{c \in C(s)} \tilde{\sigma}(s, c) + \beta V(s'), \ \forall s \in S \tag{11}$$

where $s' = \tau(s, c)$. Next period variables are once again denoted by a prime. Following Stokey and Lucas (1989), we can reformulate the problem more compactly as

$$V(s) = \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta V(s'), \ \forall s \in S \tag{12}$$

and we assume S in is a convex subset in $\mathbb{R}^k$.

The policy function is denoted as $s' = \phi(s)$, and since we only have data on people's action and not their utility, we need the policy function to estimate our model. However, to get the policy function, we need to first solve Equation 12 for the value function. Note that payoffs and transition equations are specified by the researcher a priori as *primitive* objects and the value function is derived as a solution to Equation 12.

We focus on a set of sufficient conditions for our generalized problem. See Stokey and Lucas (1989) and Bertsekas (1976) for additional theorems under different assumptions on the payoff and transition functions.

Theorem 17. *Assume $\sigma(s, s')$ is real-valued, continuous, bounded, $0 < \beta < 1$, and the constraint set $\Gamma(s)$ is non-empty, compacted-valued, and continuous. Then there exists an unique value function $V(s)$ that is a solution to Equation 12.*

Proof. See Theorem 4.6 in Stokey and Lucas (1989). □

In our sketch of the proof, we first denote **operator** T as

$$T(\mathcal{V})(s) = \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta \mathcal{V}(s'), \ \forall s \in S.$$

The mapping takes a guess of the value function and produces *another* value function $T(\mathcal{V})(s)$. Thus, for any $V(s)$ such that $V(s) = T(V)(s)$ will be a solution to Equation 12. We then can just find the **fixed points** of $T(\mathcal{V})$ to determine the solution to our problem.

The fixed point arguments need that $T(\mathcal{V})$ satisfies the monotonicity and discounting conditions from Blackwell (1965). **Monotonicity** means that for $\mathcal{V}(s) \geq Q(s), \forall s \in S$, then $T(\mathcal{V})(s) \geq T(Q)(s), \forall s \in S$.

Exercise 16. Show how monotonicity is implied by the maximization problem.

Discounting implies that adding a constant to $\mathcal{V}$ will lead $T(\mathcal{V})$ to increase by less than that constant. Then for some constant k, we have that $T(\mathcal{V} + k)(s) \leq T(\mathcal{V})(s) + \beta k, \forall s \in S$ and for $\beta \in [0, 1)$. Since we assume the discount factor is less than 1 in the dynamic programming set up, this property holds by construction.

Since $T(\mathcal{V})$ is a contraction, we can use the **contraction mapping theorem**. The theorem tells us there is (1) a unique fixed point and (2) that the fixed point can be reached by an iterative process with an arbitrary starting position in the domain of the problem. We have already seen the first property in the theorem above.

The second property is used to find a solution to Equation 12. Let $V_0(s), \forall s \in S$ be an initial guess to the solution. Then, we can construct $V_1 = T(V_0)$. If $V_1 = V_0, \forall s \in S$, then we are at a solution. If not, we can keep iterating ($V_2 = T(V_1)$ and so on), until $T(V) = V$. Since $T(V)$ is a contraction, we will eventually converge, and this iterative process is called **value function iteration**.

We will now see that the value function that is a solution to Equation 12 can inherit some properties of the problem's primitives.

Theorem 18. *Assume that $\sigma(s, s')$ is real-valued, continuous, concave, bounded, $0 < \beta < 1$, S is a convex subset of $\mathbb{R}^k$, and the constraint set $\Gamma(s)$ is non-empty, compacted-valued, convex, and continuous. Then a*

unique solution to Equation 12 is strictly concave. Additionally, the policy function $\phi(s)$ is a continuous, single-valued function.

Proof. See Theorem 4.8 in Stokey and Lucas (1989). □

The proof relies on showing $T(V)$ preserves strict concavity or if $V(s)$ is strictly concave, then so is $T(V)(s)$. Lastly, note that the theorem gives us a *stationary policy function* that depends only on the state vector. This fact will be useful in econometric applications when deriving the property of various estimators.

11.4.2 Stochastic Case

Naturally, we can add stochasticity to the dynamic program as we saw before. Let ϵ be the current value of a vector of shocks and $\epsilon \in \mathcal{E}$ be a finite set. Then, the functional equation becomes

$$V(s,\epsilon) = \max_{s' \in \Gamma(s,\epsilon)} \sigma(s,s',\epsilon) + \beta E_{\epsilon'|\epsilon}[V(s',\epsilon')],\ \forall (s,\epsilon). \tag{13}$$

The stochastic process here is purely exogenous because the distribution of ϵ' depends on ϵ but is invariant to the current state and control variables. Further the distribution of $\epsilon'|\epsilon$ is time invariant. These are direct analogs to the stationary properties of the payoff and transition equations from our cake-eating example. Once again, we let $\pi_{ij} = Pr(\epsilon' = \epsilon_j | \epsilon = \epsilon_i)$ with $\pi_{ij} \in (0,1)$ and $\sum_{j=1} \pi_{ij} = 1$. Further, Π is the transition matrix.

Theorem 19. *Assume that $\sigma(s,s',\epsilon)$ is real-valued, continuous, concave, bounded, $0 < \beta < 1$, and the constraint set $\Gamma(s,\epsilon)$ is compact and convex. Then, we have that (1) there is a unique value function $V(s,\epsilon)$ that solves Equation 13 and (2) there exists a stationary policy function $\phi(s,\epsilon)$.*

The above theorem follows in consequence to Blackwell's Theorem. Taking the FOC, we attain

$$\sigma_{s'}(s,s',\epsilon) + \beta E_{\epsilon'|\epsilon}[V_{s'}(s',\epsilon')] = 0. \tag{14}$$

Using Equation 13 to get a value for $V_{s'}(s',\epsilon')$, we attain the Euler equation

$$\sigma_{s'}(s,s',\epsilon) + \beta E_{\epsilon'|\epsilon}[\sigma_{s'}(s',s'',\epsilon')] = 0 \tag{15}$$

which tells us the expected sum of the marginal variations in the control variable in the current period must be zero. In other words, a marginal gain in this period must be offset by a marginal loss in the next period. Lastly, note that this is different from the ex-post Euler equation (after realization of ϵ')

$$\sigma_{s'}(s,s',\epsilon) + \beta \sigma_{s'}(s',s'',\epsilon') = 0 \tag{16}$$

and Equation 16 will generally not hold for all realizations of ϵ'. For the ex-ante optimization problem, ex-post errors are not fully predictable given the information set available to the agent. Estimation of stochastic dynamic programming models will leverage the Euler equation in Equation 15.

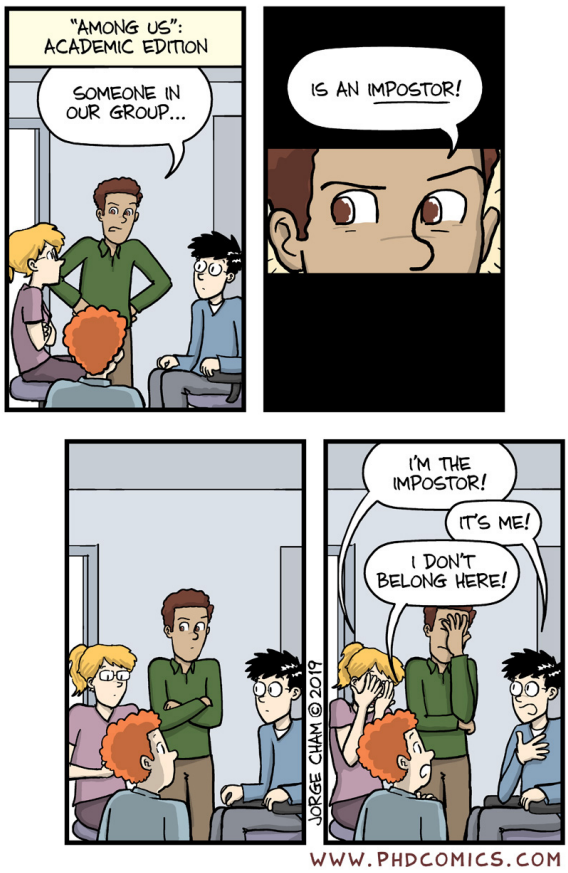
12 Example

The example notebook provides an estimation of the Rust (1987) paper, which solves an dynamic discrete choice problem with nested fixed point calculation (NFXP). This model falls under the class of optimal renewal problems and is also a Markov decision process (MDP). Please see that RMarkdown notebook for the example.

12.1 Curse of Dimensionality

In our discussion in the last section, we mainly focused on the introducing the theory of dynamic programming. However, once we consider estimating these models, we need to keep in mind the **curse of dimensionality** of dynamic programming problems. When the state vector is high dimensional (or equivalently there are many state variables in the state vector), the computational burden of evaluating the value function becomes exponentially more burdensome.

In our computational example, we discretized our one dimensional state variable into 39 different points. In the estimation step, we need to evaluate the expected value function on each of these 39 points. Now suppose our state variable is d-dimensional. Then, we would need to evaluate the value function on each of the $(39)^d$ points on the d-dimensional grid, which blows up when d becomes large. When the state space gets large, researchers often use interpolation methods (i.e. Chebyshev Polynomial Interpolation) to approximate the value function.



Dynamic Programming Example (Rust '87)

Booth Math Camp (Autumn 2021)

Walter W. Zhang

18 August 2021

In this example, we will examine a version of the Rust (1987) bus problem. This is a classic dynamic discrete choice problem and is under the class of optimal renewal problems. We will focus on the estimation using nested fixed point (NFXP), and for a more detailed description of the paper and the estimation procedure see the DSE 2019 Summer School Slides. The NFXP slides that work through the Rust problem are found here.

Contents


```
# Load packages
require(knitr)
require(kableExtra)
require(data.table)
require(ggplot2)
require(plot.matrix)
require(SQUAREM)
require(R.matlab)
require(latex2exp)
```

```
# Read in the data
data_location <- paste0("https://github.com/dseconf/DSE2019/",
                        "blob/master/02_DDC_SchjerningIskhakov/",
                        "code/zurcher/busdata1234.mat?raw=true")
DT_0 <- data.table(readMat(data_location)$data)
```

```
# Process the data
DT <- data.table(bus_id   = DT_0$V1,                # Bus id
                 bus_type = DT_0$V2,                # Bus type
                 i        = c(tail(DT_0$V5, -1), 0),  # Replacement dummy (i_t)
                 x        = DT_0$V7)                 # Odometer (x_t)
```

```r
# Set scale of x to be thousands of miles
DT[, x := x/1000]

# Primitives
time_periods <- nrow(DT)
beta         <- 0.999  # discount factor
step_size    <- 10
K_val        <- ceiling(max(DT$x)/step_size)
state_grid   <- seq(step_size, K_val * step_size, step_size) - step_size/2
```

Setting

We assume that Harold Zurcher has one bus and at the beginning of each time period he decides whether to replace the bus engine or not. When the bus engine is replaced ($i_t = 1$), the mileage becomes zero at the beginning of the period. Then the bus runs for a period and the mileage x_{t+1} is recorded. Replacing the bus engine incurs a fixed cost RC. There is a per-period dis-utility since the mileage accumulates (i.e. an increasing per-period maintenance cost). Further, there are mean zero iid T1EV shocks ϵ All together, the per-period utility has form

$$u(x_t, i_t, \epsilon_t; \theta) = \begin{cases} -c(x_t; \theta) + \epsilon_{0t} & \text{if } i_t = 0 \\ -RC - c(0; \theta) + \epsilon_{1,t} & \text{if } i_t = 1 \end{cases}$$

where we set a linear cost, $c(x_t; \theta) = \theta_1 x_t$, and the end-of-period mileage never decreases unless the engine is replaced at the beginning of the period. We assume $(x_t, \epsilon_{0,t}, i_t, \theta_2)$ follows a Markov transition probability $p(x_{t+1}, \epsilon_{0,t+1}, \epsilon_{1,t+1} | x_t, \epsilon_{0,t}\epsilon_{1,t}, i_t, \theta_2)$. The researcher observes x_t. We set $\theta = (\theta_1, \theta_2, RC)$ to be the parameters of interest.

Rust (1987)'s conditional independence (CI) assumption is that conditional on the state variable and action of the current period, the error terms do not affect the state variable next period. In symbols, we see that

$$p(x_{t+1}, \epsilon_{0,t+1}, \epsilon_{1,t+1} | x_t, \epsilon_{0,t}\epsilon_{1,t}, i_t, \theta_2) = p(x_{t+1} | x_t, i_t, \theta_2) g(\epsilon_{0,t+1}) g(\epsilon_{1,t+1})$$

where $g(\cdot)$ is the TIEV distribution function for the shocks (or error terms).

From the iid assumption of the TIEV error terms, we have $g(\epsilon_{0,t+1})$ and $g(\epsilon_{1,t+1})$ on the RHS. The CI assumption is fleshed out in that $p(x_{t+1} | x_t, i_t, \theta_2)$ shows the transition to the next state only depends on the current state and the choice take at the current time period. The CI assumption will impose a conditional first-order Markov structure on the decision problem. This will help simplify the dynamic discrete choice problem.

State space discretization

We discretize the state space by dividing the mileage from the data by 10. This will yield us an upper bound at 390 since the maximum mileage (in the thousands) observed is 387.282. Then our discretized state will have the state s in buckets $1, 2, 3, \ldots, K$ where in our set up $K = 39$. s here is the mileage buckets that discretize the state space. Further, note that $p(x_{t+1} | x_t, i_t = 1, \theta_2)$ will be a $1 \times K$ vector since when replacing the bus engine ($i_t = 1$), the new engine will have mileage set to zero regardless of the value of the previous state variable.

We can estimate the transition probabilities, $p(x_{t+1} | x_t, i_t = 1, \theta_2)$, in our discretized state space from the simple non-parametric frequency estimator (θ_2 is "estimated" non-parametrically):

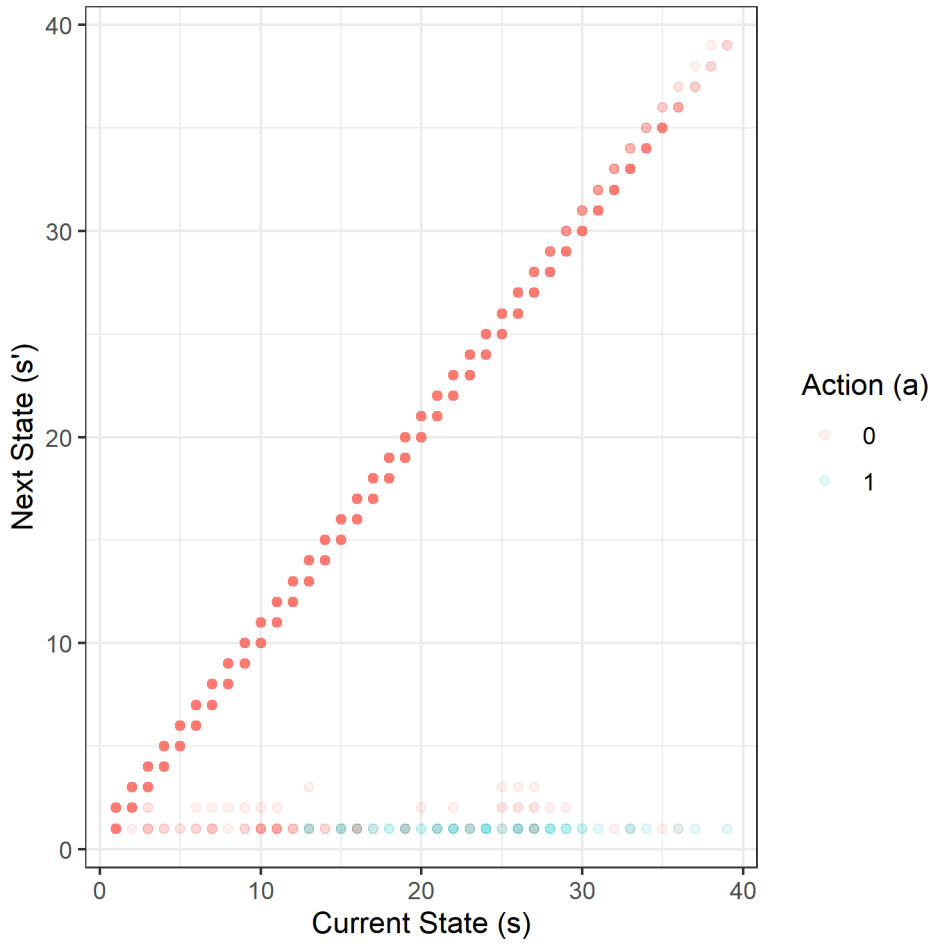
$$p(s'|s,a,\theta_2) = \frac{\sum_{t=1}^{T} \mathbf{1}\{x_{t+1} = s', x_t = s, i_t = a\}}{\sum_{k \in \mathcal{S}} \sum_{t=1}^{T} \mathbf{1}\{x_{t+1} = l, x_t = s, i_t = a\}}$$

where s' is the new state, s is the past state, and a is the action chosen.

We can visualize the state transition matrix in the following plot.

```
# Discretize the state space
DT[, s   := floor(x/step_size) + 1]
DT[, s1  := shift(s, -1)]

# Drop the last observation
DT <- DT[complete.cases(DT)]
```

```
# Graph the state transition
graph_DT <- copy(DT)[, c("i", "s", "s1")]
graph_DT[, i := factor(as.integer(i))]
ggplot(graph_DT, aes(x = s, y = s1, color = i)) +
  geom_point(alpha = 0.1) +
  labs(color = "Action (a)") +
  xlab("Current State (s)") +
  ylab("Next State (s')") +
  theme_bw()
```

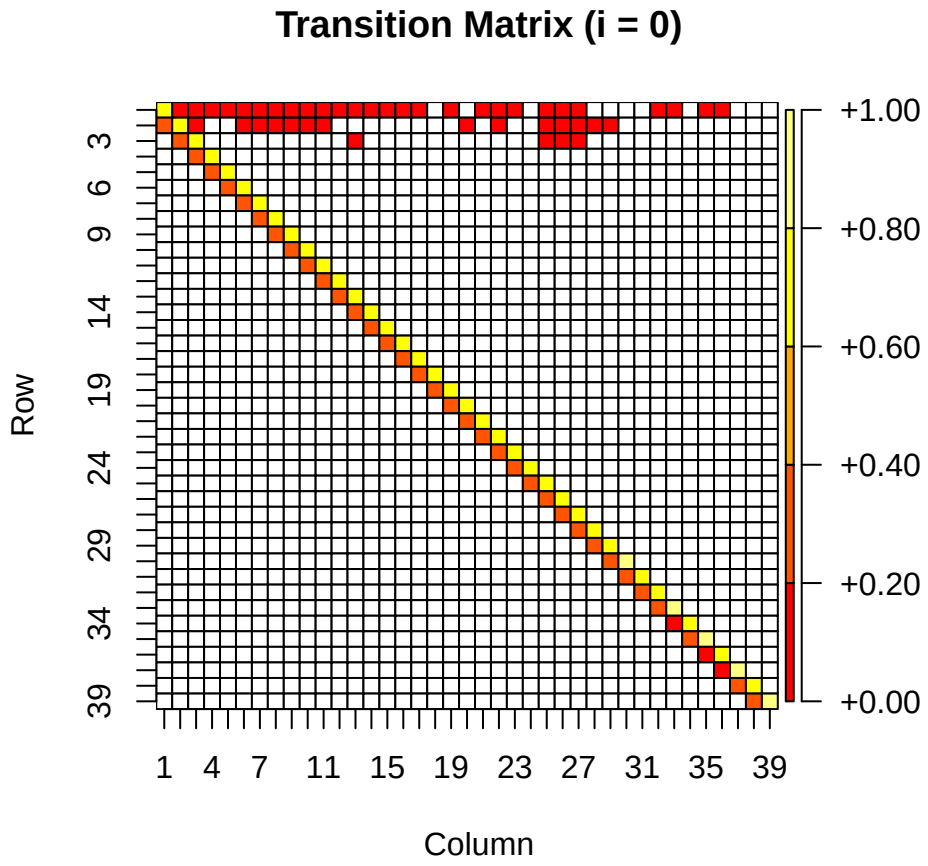


From the plot, we see that generally Harold Zurcher replaces his bus engine when the current state is high.

```
# Construct the transition matrix
## Rows are next state, columns are current state

## i = 0 Case
t0 <- data.table(s1 = 1:K_val)
for (s_val in 1:(K_val))
{
  freq_s_val <- table(DT[i == 0 & s == s_val, s1])
  if (length(freq_s_val) == 0)
  {
    # Zero frequency case
    freq_DT <- data.table(V1 = 1:K_val, N = NA)
  } else
  {
    # Frequency in non-zero case
    freq_DT <- data.table(freq_s_val)
    freq_DT[, V1 := as.integer(V1)]
  }
  t0 <- merge(t0, freq_DT, by.x = "s1", by.y = "V1", all.x = TRUE)
  setnames(t0, "N", as.character(s_val))
}
t0[is.na(t0)] <- 0
trans_mat_0 <- as.matrix(unname(t0[, -1]))
trans_mat_0 <- sweep(trans_mat_0, 2, colSums(trans_mat_0), "/")
trans_mat_0[is.nan(trans_mat_0)] <- 0


## i = 1 Case (Just a vector)
t1 <- data.table(s1 = 1:K_val)
freq_DT <- data.table(table(DT[i == 1, s1]))
freq_DT[, V1 := as.integer(V1)]
t1 <- merge(t1, freq_DT, by.x = "s1", by.y = "V1", all.x = TRUE)
t1[is.na(t1)] <- 0
trans_mat_1 <- as.matrix(unname(t1[,-1]))
trans_mat_1 <- trans_mat_1/colSums(trans_mat_1)
## Stack into a matrix
trans_mat_1 <- matrix(trans_mat_1, nrow = K_val, ncol = K_val)
```

We can also visualize the $i = 0$ transition matrix.

```
# i = 0 matrix
par(mar=c(5.1, 4.1, 4.1, 4.1))
t0_mat <- copy(trans_mat_0)
t0_mat[t0_mat == 0] <- NA
plot(t0_mat, main = "Transition Matrix (i = 0)")
par(mar=c(5.1, 4.1, 4.1, 2.1))
```



Likelihood derivation

We now derive $Pr(i_t = 0|x_t, \theta)$ and $Pr(i_t = 1|x_t, \theta)$ as a function of the expected value function $EV_\theta(x; i) = \int V_\theta(y, \epsilon)p(d\epsilon)p(dy|x, i, \theta_2)$ and the period specific utility $\bar{u}(x_t, i_t; \theta)$ net of ϵ.

From the additive separability of x_t and $\epsilon_{1,t}, \epsilon_{0,t}$ in the utility function ($u(x_t, i_t; \theta)$), we write the per-period utility function (with ϵ) as

$$u(x_t, i_t; \theta) = \bar{u}(x_t, i_t, \epsilon_t; \theta) + \epsilon_{i,t}, \forall i_t \in 0, 1$$

We also define $v_i(x)$ as the choice-specific or alternative value function.

From Rust (1987), we can use the standard Blackwell Theorems that tell us the optimal value function exists and is the unique solution to the Bellman Equation,

$$\begin{aligned} V_\theta(x, \epsilon) &= \max_{i \in \{0,1\}} \left\{ \bar{u}(x, i; \theta) + \beta EV_\theta(x, i) \right\} \\ &= \max_{i \in \{0,1\}} \left\{ v_i(x) + \epsilon_i \right\} \end{aligned}$$

Then, the conditional choice probabilities ($Pr(i_t = 0|x_t, \theta)$ and $Pr(i_t = 1|x_t, \theta)$), will have the standard logit form.

$$\begin{aligned} Pr(i_t = i|x_t, \theta) &= Pr\{v_i(x_t) + \epsilon_i \geq v_j(x_t) + \epsilon_j\} \\ &= \frac{\exp(v_i(x_t))}{\sum_{k \in \{0,1\}} \exp(v_k(x_t))} \end{aligned}$$

where $i, j \in \{0, 1\}$ and $j = 1 - i$.

Fixed point equation

We can use the log-sum property of the TIEV errors to show that the fixed point equation holds. More specifically, we use the fact that ϵ_t are mean zero, iid, and TIEV distributed. Then, we first see that

$$\begin{aligned}\int_\epsilon V_\theta(y, \epsilon) p(d\epsilon) &= \int_\epsilon \max_{i \in \{0,1\}} \{v_i(y) + \epsilon_i\} p(d\epsilon) \\ &= \log\Big(\sum_{k \in \{0,1\}} \exp(v_k(y)) \Big) \\ &= \log\Big(\sum_{k \in \{0,1\}} \exp(\bar{u}(y, k; \theta) + EV_\theta(y, k)) \Big)\end{aligned}$$

Then, we can use the conditional independence assumption that we discussed before to get,

$$\begin{aligned}EV_\theta(x, i) &= \int_y \int_\epsilon V_\theta(y, \epsilon) p(d\epsilon) p(dy|x, i, \theta) \\ &= \int_i \Big[\int_\epsilon V_\theta(y, \epsilon) p(d\epsilon) \Big] p(dy|x, i, \theta)\end{aligned}$$

Using our derivation above, we plug in for the value in the brackets, and attain

$$EV_\theta(x, i) = \int_i \log\Big(\sum_{k \in \{0,1\}} \exp(\bar{u}(y, k; \theta) + EV_\theta(y, k)) \Big) p(dy|x, i, \theta)$$

which is what we wanted to show.

MLE estimation with NFXP

We implement the MLE estimator with $\beta = 0.999$. From our results from before, we can write the likelihood as

$$\mathcal{L}(x_1, \ldots, x_T, i_1, \ldots, i_T | \theta) = \sum_{t=1}^{T} \log(P(i_t | x_t, \theta))$$

The nested fixed point (NXFP) algorithm has an outer and inner loop. The optimizer choose the parameters θ_1, RC in the outer loop. In the inner loop, the expected value function fixed point iteration is run to get the value function given the chosen parameters. Thus, in each step of the outer loop (or the MLE optimizer), a fixed point iteration of the expected value function is run on across the discretized state space. We use the `SQUAREM` package to perform the inner loop evaluation. We use the `BFGS` optimization algorithm for the outer loop.

```
# nll_Rust ------------------------------------------------------------------
#' Computes the negative log-likelihood for Rust Problem
#' @param theta_vec A vector of parameters (theta_1, RC)
#' @return Log-likelihood value (numeric)

nll_Rust   <-   function(theta_vec)
{
  # Parameters
  theta_1 <- theta_vec[1]
  RC      <- theta_vec[2]
```

```
  # Set up primitives
  u_bar_vec   <- rep(0, 2 * K_val)
  ev_vec_init <- rep(0, 2 * K_val)
  cost_vec    <- theta_1 * state_grid
  u_bar_vec[1:K_val]                 <- -1 * cost_vec
  u_bar_vec[(K_val + 1):(2 * K_val)] <- -1 * RC

  # NXFP w/ SQUAREM (inner loop)
  ## NXFP uses value function iteration in the inner loop
  ## (Expected) Value Function Iteration with `squarem`
  ev_res <- squarem(par = ev_vec_init, fixptfn = ev_Rust)
  ev_res_vec <- ev_res$par

  # Compute NLL
  ## Choice specific value function
  v_0 <- u_bar_vec[1:K_val] + beta * ev_res_vec[1:K_val]
  v_1 <- u_bar_vec[(K_val + 1):(2 * K_val)] + beta * ev_res_vec[(K_val + 1):(2 * K_val)]
  v_max_val <- max(c(v_0, v_1))  # Avoid overflow
  ## CCPs (Conditional choice probabilities)
  ccp_0 <- exp(v_0 - v_max_val) / (exp(v_1 - v_max_val) + exp(v_0 - v_max_val))
  ccp_1 <- 1 - ccp_0
  ## NLL value (Negative log-likelihood)
  nll <- -1 * sum(log(c(DT[i == 0, ccp_0[s]], DT[i == 1, ccp_1[s]])))

  return(nll)
}

# ------------------------------------------------------------------------------
```

```
# ev_Rust ----------------------------------------------------------------------
#' Computes the fixed point mapping
#' Runs inside the `nll_Rust` function
#' @param ev_vec Value of the expected value function
#' @return A vector of new expected value function values (numeric vector)

ev_Rust   <-   function(ev_vec)
{
  # Avoid overflow
  inner_val <- u_bar_vec_ + beta * ev_vec
  inner_max <- max(inner_val)
  inner_val <- exp(inner_val - inner_max)
  outer_val <- log(inner_val[1:K_val] + inner_val[(K_val + 1):(2 * K_val)])
  outer_val <- outer_val + inner_max

  # EV computation
  ev_i_0 <- outer_val %*% trans_mat_0
  ev_i_1 <- outer_val %*% trans_mat_1

  return(c(ev_i_0, ev_i_1))
}

# ------------------------------------------------------------------------------
```

Parameter	Estimate	SE
θ_1	0.014	0.00053
RC	7.286	0.13035
LL	-306.522	-

```
# Run the outer loop

## Initial values
theta_init <- c(0.3, 0.1)
## Test Run with initial values
nll_Rust(theta_init)
```

```
[1] 282427.3
```

```
## BFGS Optimizer
init_time <- Sys.time()
optim_res <- optim(theta_init,
                   nll_Rust,
                   method  = "BFGS",
                   hessian = TRUE,
                   control = list(reltol = 1e-10))
end_time <- Sys.time()
print(paste0("Time Elapsed: ", format(end_time - init_time, digits = 3)))
```

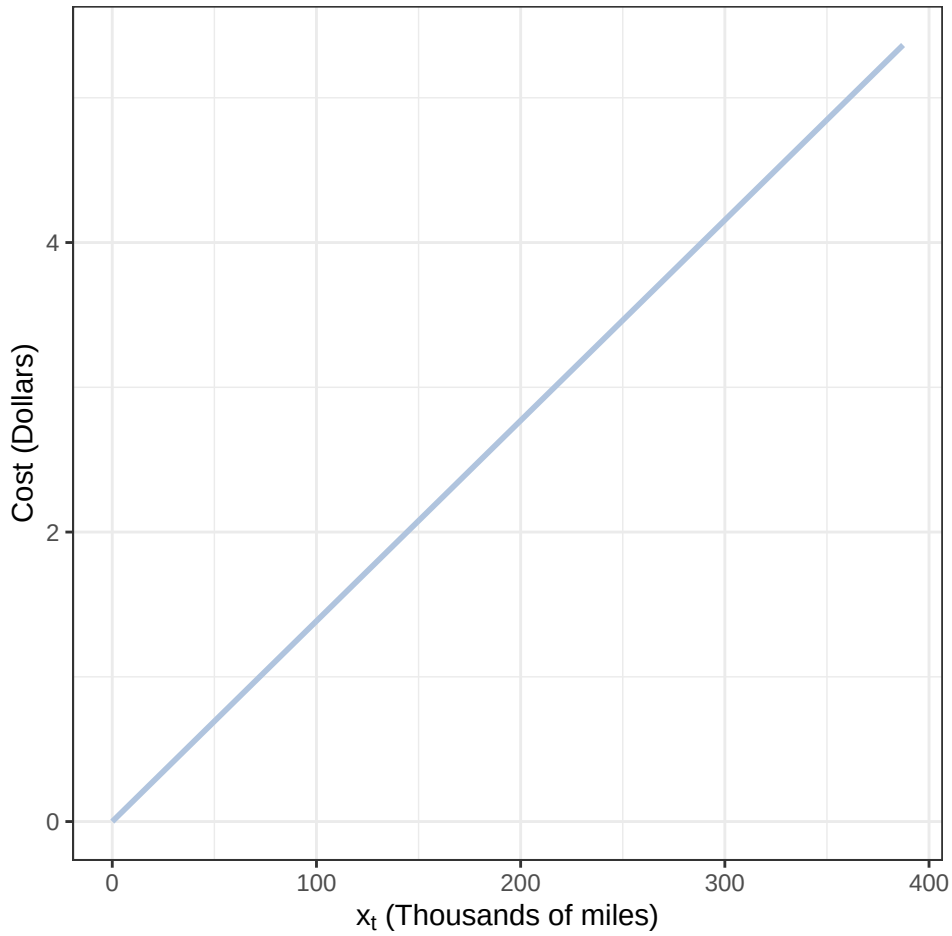
```
[1] "Time Elapsed: 0.75 secs"
```

The optimizer converged with code 0, which implies it has converged. We get the following estimates for the parameters of interest. MLE standard errors are recovered from the numerically computed Hessian.

```
# Results
res_DT <- data.table(Parameter = c("$\\theta_1$",  "$RC$", "LL"),
                     Estimate  = round(c(optim_res$par, optim_res$value * -1),
                                       digits = 3),
                     SE        = c(round(1/sqrt(diag(optim_res$hessian)),
                                         digits = 5), "-"))
kable(res_DT, booktabs = TRUE, format = 'latex', escape = FALSE) %>%
  kable_styling(latex_options = "striped", position = "center")
```

We can also plot the cost function by the state variable using our estimated coefficients.

```
cost_func <- function(x) {optim_res$par[1] * x }
cost_DT   <- data.table(x = seq(0, max(DT$x), length.out = 1000))
cost_DT[, cost := cost_func(x)]

ggplot(cost_DT, aes(x = x, y = cost)) +
  geom_line(size = 1,  color = "lightsteelblue") +
  xlab(TeX("$x_t$ (Thousands of miles)")) +
  ylab("Cost (Dollars)") +
  theme_bw()
```



Exercises

1. Estimate the Rust Problem with a quadratic cost function.

$$\tilde{c}(x_t; \theta) = \theta_{10} x_t + \theta_{11} x_t^2$$

   (*Hint*: Check for underflow/overflow issues with the estimation)
2. We used the supplied replacement indicator from the data. Assume that we were not given this indicator and instead had to construct it from the odometer data. (A replacement indicator, $\tilde{i}_t$, would be when the mileage next period is smaller than the current period.) How would the results differ? What would we miss? Will this bias our estimates?

$$\tilde{i}_t = \begin{cases} 1 & \text{if } x_{t+1} < x_t \\ 0 & \text{if } x_{t+1} \geq x_t \end{cases}$$

3. Instead of using the `SQUAREM` package for expected value function iteration, write out your own value function iteration algorithm. Is it faster or slower than the `squarem` function? Can you think of ways to speed up the expected value function iteration?