bEhaVioral Insights anD Effective eNergy policy acTions

Project No. 957117

Project acronym: EVIDENT

Project title:

Behavioral Insights and Effective Energy Policy Actions

Deliverable 4.4

Analytical usage handbooks for tools and datasets

Programme: H2020-LC-SC3-EE-2020-1

Start date of project: 01 December 2020

Duration: 36 months

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957117

Document Control Page

Deliverable Name	Analytical usage handbooks for tools and datasets
Deliverable Number	D4.4
Work Package	WP4
Associated Task	T4.3
Covered Period	M20 – M34
Due Date	September 30, 2023
Completion Date	September 28, 2023
Submission Date	September 29, 2023
Deliverable Lead Partner	Democritus University of Thrace (DUTH)
Deliverable Author(s)	Ioannis Pragidis (DUTH), Paris Karypidis (DUTH), Vasileios Melissianos (PPC), Paul Liston (TCD)
Version	**1.0**

Dissemination Level		
PU	Public	**X**
CO	Confidential to a group specified by the consortium (including the Commission Services)	

Document History

Version	Date	Change History	Author(s)	Organisation
0.1	May 30, 2023	Table of Contents	Paris Karypidis	DUTH
0.2	June 15, 2023	Final version of section 2	Paris Karypidis	DUTH
0.3	July 30, 2023	Final version of section 3	Paris Karypidis	DUTH
0.4	August 10, 2023	Initial version of section 7	Paris Karypidis	DUTH
0.5	August 20, 2023	Final version of section 6	Vasileios Melissianos	PPC
0.6	August 20, 2023	Final version of section 7	Paris Karypidis	DUTH
0.7	August 28, 2023	Initial version of section 8 and 9	Paul Liston	TCD
0.8	September 5, 2023	Initial version of section 4 and 5	Ioannis Pragidis, Karypidis Paris	DUTH
0.9	September 22, 2023	Final version of sections 4, 5, 8 and 9	Karypidis Paris, Paul Liston	DUTH, TCD
1.0	September 28, 2023	Final Version of the Deliverable after Review	Karypidis Paris	DUTH

Internal Review History

Name	Institution	Date
Dimitris Pliatsios	UOWM	September 28, 2023
Konstantinos Kyranou	SID	September 26, 2023

Quality Manager Revision

Name	Institution	Date
Dimosthenis Ioannidis	CERTH	September 28, 2023

Table of Contents

List of Figures

List of Code Blocks

Acronyms

Acronym	Explanation
3ie	International Initiative for Impact Evaluation
AEA	American Economic Association
ATE	Average Treatment Effect
CATE	Conditional Average Treatment Effect
DIME	Development Impact Evaluation
DIME	Development Impact Evaluation
EGAP	Evidence in Governance and Politics
GRF	Generalized random forests
HERs	Home Energy Reports
ITE	Individual Treatment Effect
ITE	Individual Treatment Effects
ML	Machine Learning
MLP	Multilayer Preceptor
OSF	Open Science Framework
PAP	Pre-Analysis Plan
PEP8	Python Enhancement Proposal 8
PV	Photovoltaic
RF	Random Forest
RMSE	Root Mean Squared Error
SHT	Social House Tariff
SPSS	Statistical Package for the Social Sciences
SVR	Support Vector Regressor

Executive Summary

This deliverable aims to produce a comprehensive user handbook that analyses the data structure and coding practices utilised in the analysis of the collected data. The primary objective of this handbook is to facilitate seamless replication by 3rd party researchers and provide a valuable collection of instructions for easy reference. To achieve this goal, the document is divided into 11 main sections. The first section introduces principles of credible, transparent, and reproducible research, while the second section presents the Development Impact Evaluation (DIME) framework, along with its main open-source resources (DIME Wiki, toolkits, and reproducibility protocols).

The subsequent sections of the deliverable present the methodological frameworks and analytical tools developed. Each section is dedicated to an analysis linked to the EVIDENT project's use cases. Starting with use cases 1 and 2, sections 4, 5, and 6 present the analytical tools developed to assess the impact of EVIDENT's natural field experiment in household energy consumption, examine the degree of attentiveness of customers when choosing an electricity contract, and estimate households' energy consumption based on consumers' characteristics. The following section, Section 7, presents the methodologies employed within the context of use case 3, encompassing two distinct analyses. In the first case, we construct a framework for forecasting residential energy consumption and production using various linear and machine learning (ML) models. The second analysis utilizes the predictive capabilities of ML models and the data from CW's natural field experiment to develop a methodological framework for estimating heterogeneity in causal effects. Sections 8, 9, and 10 present the methodological framework used in use cases 4 and 5. Specifically, these sections introduce the analytical tools developed to estimate consumers' willingness to pay for the repair of home appliances and for more energy-efficient home appliances, as well as to assess consumers' average price bias.

The preceding analyses utilized data from various sources, including energy companies, partners of the EVIDENT consortium, and the data collection processes performed by the EVIDENT consortium. For more comprehensive details about the data, the analysis, and the results of these analyses, please refer to D4.2, 'Econometric Analysis and Robustness Tests,' and D4.3, 'Updated Econometric Methodologies and Robustness Tests.

Throughout this deliverable, the authors make extensive use of appendices and code block sections, which serve as supplementary resources. These sections provide additional information about the DIME framework and present a compilation of instructions and programming snippets for quick reference.

1. Purpose and Overall Structure of the Deliverable

1.1 Purpose of the Deliverable

D4.4 “Analytical Usage Handbooks for Tools and Datasets” is the second deliverable within Task 4.3 “Optimized Econometric Methods and Usage Handbook”. The primary objective of Task 4.3 is to provide updated econometric models developed during the final stages of the EVIDENT project, along with enhancements in the collected datasets. Additionally, through D4.4, Task 4.3 aims to create a comprehensive user handbook that analyses the data structure and coding practices employed during the analysis of the collected data. This handbook is designed to facilitate seamless replication by 3rd party researchers and offers a valuable collection of instructions for easy reference.

1.2 Relation with other Deliverables and Tasks

Deliverable 4.4 builds upon the insights from D4.2, ”Econometric Analysis and Robustness Tests” and incorporates preliminary findings from D4.3, ”Updated Econometric Methodologies and Robustness Tests”. Its primary objective is to serve as a usage handbook, intended for third parties, facilitating the replication of results and fostering research advancements in the field of energy efficiency.

1.3 Structure of the Document

This deliverable is organized as follows:

- Section 2 – Introduction: This section provides an overview of the deliverable's purpose and content.
- Section 3 – The Development Impact Evaluation (DIME) framework: This section explains the adoption of the World Bank's DIME framework by the EVIDENT consortium which serves as a foundation for organizing the project's datasets and analytical tools.
- Sections 4 – 10: These sections present a comprehensive user handbook for both the data structure and coding utilized across all analyses within the EVIDENT project.
- Section 11 – Conclusion: This section summarizes the deliverable.

2. Introduction

Evidence-based policymaking is vital and should be based on sound and credible results of development research as it directly impacts the daily lives of thousands of individuals worldwide. In recent years, the open science movement has gained prominence, emphasising that research is a public service. Researchers are encouraged to provide comprehensive insights into their methodologies, data, and approaches to ensure the creation of high-quality, well-documented research.

Central to this approach are three key principles presented in Figure 1: credibility, transparency, and reproducibility. Researchers adhere to common standards and best practices for sharing their work, including data and analytical tools, with other researchers. This approach not only enhances the value of their work, but also fosters a culture of analytical rigor, skepticism, and thoroughness in research assumptions.

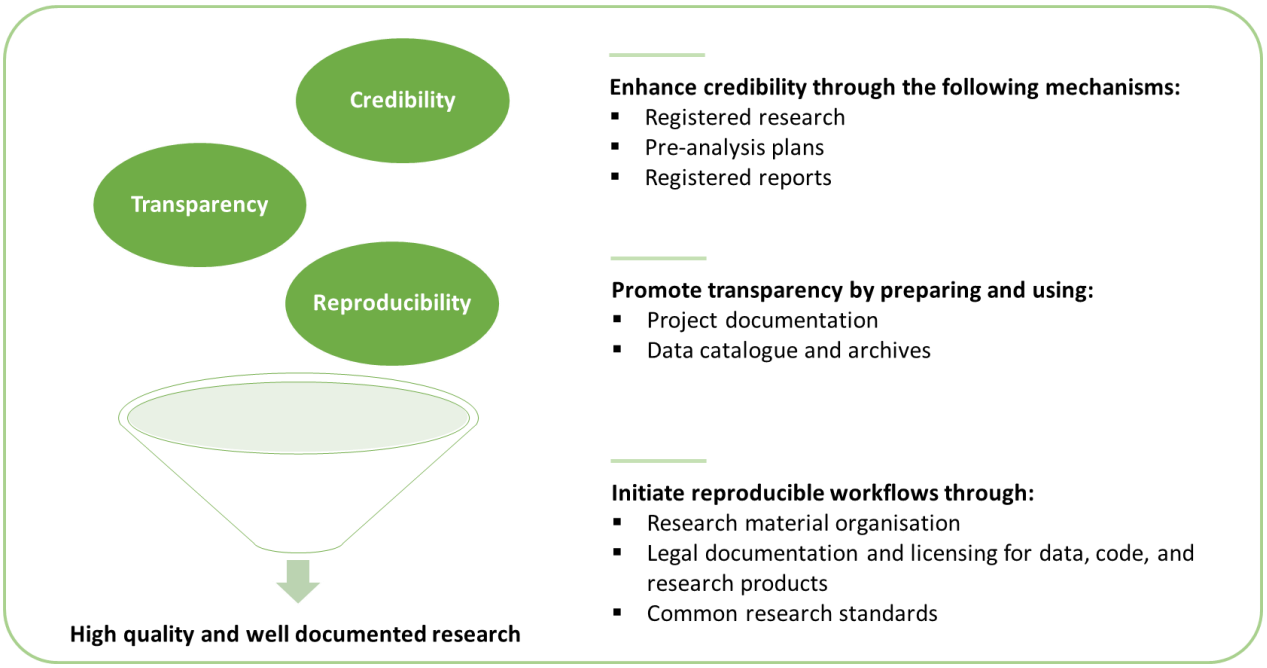

Figure 1: Principles for high quality and well documented research

Credibility

Researchers often face criticism when they choose results or outcomes after implementing projects or gathering field data. This practice can lead to potentially misleading or even 'false positive' results, which may not hold true beyond the specific experimental context. In research, especially when working with original data sources, the credibility of research design and how data is collected and utilized are crucial elements. Adhering to established, credible research practices is essential to prevent serious errors or misleading findings.

To mitigate potential criticism and ensure robust research, researchers can employ the following methodologies when designing their research, formulating research questions, and selecting their methods:

- Registered research: The primary aim of study registration is to ensure that a comprehensive record of research inquiries is readily accessible. By registering their work, researchers ensure that future scholars can easily discover what research has been conducted on a particular question, even if some or all of the work remains unpublished. Study registration can occur before, during, or after a study is completed, providing crucial information about its objectives. Researchers have the flexibility to choose a registry that aligns with the nature of their work, as each registry caters to different audiences and offers distinct features[1]. Importantly, study registration is accessible to all projects, as registries are typically free to use, and initial registration can be done with minimal information. A common practice is to gradually revise and expand the level of detail in the registry as the project progresses. Preregistration provides a straightforward and low-effort method for researchers to demonstrate that their research questions were not influenced by the data collection or analysis process, particularly when specific hypotheses are included in the preregistration. Registering research studies is becoming increasingly common, with many journals now requiring such registration for the studies they publish (Vilhuber, Turrito, & Welch, 2020).
- Pre-analysis plans: A pre-analysis plan (PAP) contains information and details about the analyses researchers intend to conduct in advance. That way, researchers can argue on HARKing, a term introduced by (Kerr, 1998), meaning that they form their hypothesis after the results are known. The primary function of a PAP is to describe one or more specific data-driven inquiries explicitly. This is crucial because specific formulations are often challenging to justify retrospectively with data, especially in projects where different approaches can be used to answer a single theoretical question. Any research aspect outside the original plan remains just as interesting and valuable as if it had never been published. However, committing to the details of a particular inquiry in advance, renders its results impervious to a wide range of criticisms related to specification searching or multiple testing. It's important to note that PAPs should not be seen as constraining researchers. Depending on the level of knowledge about the study at the time of writing, PAPs can vary in the level of detail they encompass. Various templates and checklists are available to guide the inclusion of essential information[2].
- Registered reports: A registered report takes the form of a formal publication. It combines the advantages of the initial two steps—registered research and pre-analysis planning—with a formal peer review process, ultimately securing conditional acceptance for the outlined analysis. While

[1] Common registries are the American Economic Association (AEA; https://www.socialscienceregistry.org), the International Initiative for Impact Evaluation (3ie; https://ridie.3ieimpact.org), Evidence in Governance and Politics (EGAP; https://egap.org/content/registration), and the Open Science Framework (OSF; https://osf.io/registries).
[2] The interest reader can find a checklist in (McKenzie, 2012).

registered reports are not obligatory and demand a higher level of rigor for publication, they offer several benefits to researchers. Through the peer review process, researchers or research teams receive valuable comments and expert feedback, even during the research design phase (Foster, Karlan, & Miguel, 2018). Registered reports particularly reward researchers who are willing to provide comprehensive advanced details about their research, seek publication interest irrespective of the results, or employ novel or unconventional methods.

Transparency

Research transparency is the key to allowing readers to independently assess the quality and reliability of research. It empowers them to judge whether a study is well-structured, methodologically sound, and provides valuable insights. Achieving transparency requires proactive efforts from researchers, including sharing the analytical tools they've developed and detailing the research processes they've employed.

Researchers who prioritise transparency have a built-in incentive to make informed decisions and approach their work with scepticism and thoroughness, which, in turn, saves them time by streamlining the coding process and preventing redundant discussions.

Clearly documented research is essential for enabling others to evaluate precisely which data were collected and how that information contributed to specific outcomes. Many research approaches address unique questions, often using distinctive data and innovative methodologies. While these approaches can yield fresh insights into critical academic inquiries, they must be transparently documented to facilitate future reviews or replications by others (Duvendack, Palmer-Jones, & Reed, 2017).

To promote research transparency, researchers are encouraged to prepare and leverage methodologies such as:

- Project documentation: Comprehensive project documentation is fundamental for enhancing transparency in research. Transparency necessitates the explicit recording of decisions as they are made, accompanied by thorough explanations of the underlying decision-making processes. Careful documentation offers substantial time savings for research teams by preventing the need to revisit discussions multiple times; the reasons behind specific choices are preserved in records. Several tools are available[3] to facilitate effective documentation, and this process should be ongoing rather than being an one-time requirement. As plans evolve into reality, new decisions are continuously made, and it's entirely acceptable to adapt sensibly, as long as these adaptations are recorded and disclosed. Given that each project has unique requirements for managing data, code, and documentation, it's essential to select the appropriate transparency methods before launching the project.
- Data catalogue and archives: Data and the methods used to collect it should undergo thorough cataloguing, archiving, and documentation, whether the data is generated internally by the

[3] Some well-known services are the Open Science Framework (OSF; https://osf.io) and GitHub (https://github.com).

project or obtained from external partners. This documentation should be continuously updated and kept alongside other study materials, often compiled for publication in an appendix. When data is either received from partners or collected in the field, it should be promptly stored in a secure, permanent storage system, including any field corrections. Before commencing analytical work, a "for-publication" copy of the acquired data set must be created by removing all personally identifying information. This public version of the data set should be deposited in an archival repository where it can be formally cited (Vilhuber, Turrito, & Welch, 2020). In cases where confidential data is necessary for planned analyses, these data should be stored separately, most likely in an encrypted form, to clearly distinguish which parts of the code will function with or without access to the restricted data.

Reproducibility

Reproducible research serves as a cornerstone for enabling other researchers to apply the introduced methodologies to new datasets or to implement a similar research design in different domains. In the research community, there is a growing shift towards the imposition of specific reproducibility guidelines (Christensen & Miguel, 2018).

To initiate reproducible workflow, researchers are encouraged to follow the following principles:

- Research material organisation: Major publishers and funding bodies, have taken notable strides in mandating accurate reporting, citation, and preservation of code and data as research outputs, thereby ensuring accessibility and verification by other researchers. This approach transforms research into a public good, facilitating the ease with which 3rd party researchers can utilize code and processes to enhance their own future work. For example, it is crucial to provide code in a clear and well-documented manner, making it easy for others to comprehend while at the same time the associated data should be publicly available to extent at the greatest level the potential impact of the research.
- Common research standards: Research standards often enclose both regulatory and verification policies (Stodden, Guo, & Ma, 2013). Regulatory policies necessitate authors to provide reproducibility packages before publication, subject to review by the journal for completeness. In contrast, verification policies require authors to make certain materials accessible to the public, even if their completeness is not a prerequisite for publication. Certain journals offer guidelines and checklists for reporting the implementation of various practices while producing such resources also offers the advantage of creating additional opportunities for citations.
- Legal documentation and licensing for data, code, and research products: When circumstances prevent data publication, it is essential to publish as much metadata as possible. This includes information on data acquisition methods, data field descriptions, and aggregations or descriptive statistics. Even when data publication is not feasible, code files typically contain no restricted information, making it crucial to make the code available with clear instructions on how to use it. Furthermore, when designing informed consent protocols or data license agreements for sensitive data, reproducibility requirements should be taken into account. This may involve setting acceptable conditions, such as secure data transfer or access via controlled environments, under which 3rd party researcher can independently reproduce results.

3. The Development Impact Evaluation (DIME) framework

The EVIDENT consortium adopted the Development Impact Evaluation (DIME) framework to organise better its datasets and analytical tools and provide a well-structured ready reference usage handbook. This section presents the main aims of the DIME framework and the provided open-source resources like the DIME Wiki, toolkits, and reproducibility protocols. It also describes the bests practices for handling data effectively, efficiently, and ethically. This section extensively uses the appendices section since technical guidance (e.g., information about folder structure, best coding practices, etc.) is provided.

DIME (Bjarkefur, Cardoso de Andrade, Daniels, & Jones, 2021) is an organizing principle for constituting high-quality data sets. It helps researchers structure the content and characteristics of data sets to enable policy analysis. It's introduced by World Bank and includes several steps researchers should follow to better organise their research, their tools and data. The main idea is that replication materials shall include the initial datasets, sufficient descriptions, the tools used for pre-processing and creating the analysis data, the analytical tools for the final analysis, and sufficient descriptions for the executions of the tools (README files etc.).

DIME offers a descriptive overview of the data workflow throughout every phase of an empirical research project, starting from the design phase and culminating in publication, as visualized in Figure 2.

- Research Design: The DIME framework emphasizes the importance of carefully selecting the appropriate research design for a given evaluation. This involves selecting the most appropriate intervention to evaluate, choosing the right comparison group, and selecting appropriate outcome measures.
- Data Acquisition: The DIME framework emphasises on the importance of collecting high-quality data for impact evaluations. This involves selecting the most appropriate data collection methods, designing survey instruments and questionnaires, and ensuring that the data is collected in a consistent and reliable manner.
- Data Processing and Analysis: The DIME framework emphasizes the importance of using rigorous statistical methods to analyse data collected during impact evaluations. This involves selecting the most appropriate analytical methods, controlling for confounding variables, and estimating the precision of the estimated effects.
- Reporting and Publication: The DIME framework emphasizes the importance of reporting and disseminating the results of impact evaluations in a transparent and accessible manner. This involves preparing reports, publishing papers in academic journals, and presenting findings to policymakers and other stakeholders.

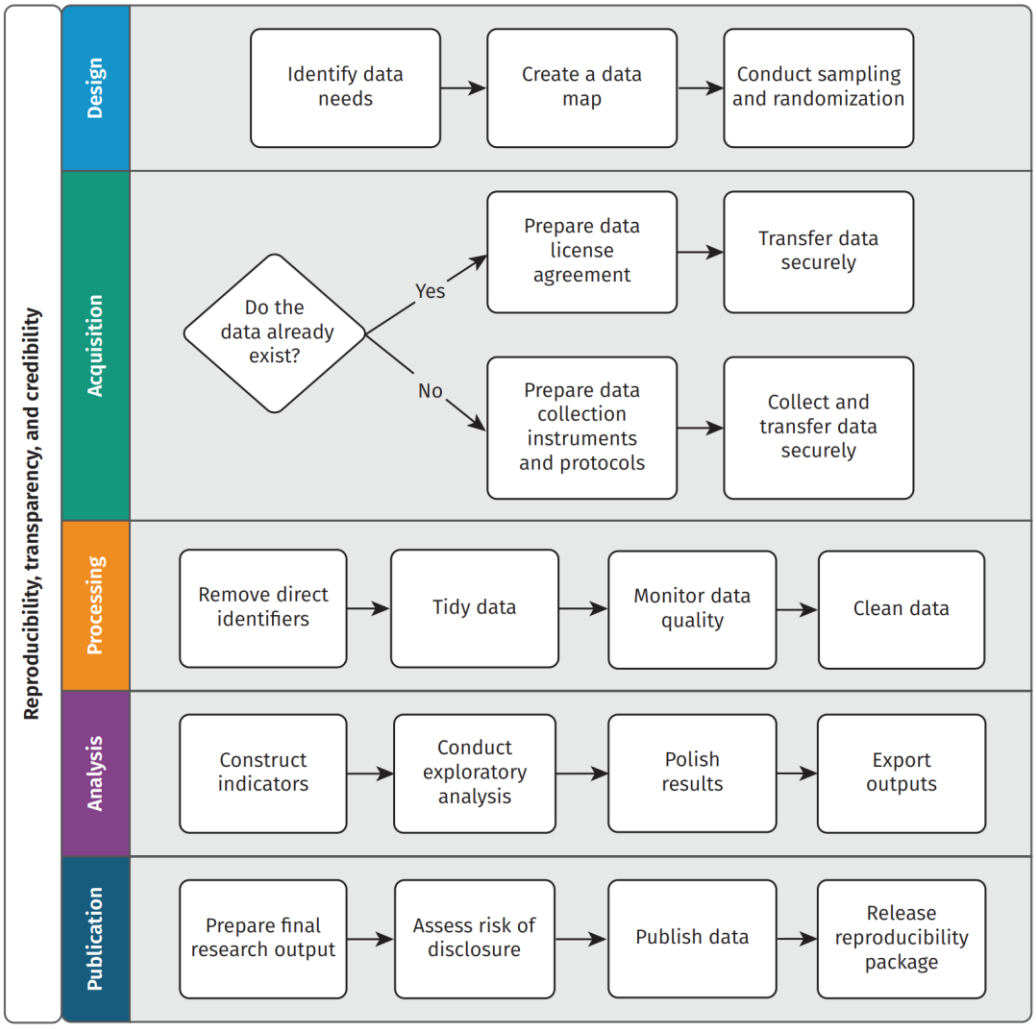

Figure 2: Overview of the tasks involved in development research data work

DIME framework also provides guidelines on how researchers can follow a common research standard even at the level of the analytical tools developed. Thus, it provides or presents well known coding principles. In the context of the EVIDENT project, several programming languages and different datasets were used. In each case, the researchers tried follow DIME's guidelines for efficiently managing data, as presented in Appendix 1: Efficient data management and stick to each programming language's official style guidelines as presented in Appendix 2: Writing "high-quality" code.

4. Econometric analysis for randomized experiments

As presented in D4.2 “Econometric analysis and robustness tests”, the main objective of this analysis is to estimate the effects of consumption feedback and peer comparison on energy consumption. Thus, in the context of use case 1 and 2, the EVIDENT consortium conducted the CW natural field experiment which started on December 2021, including 867 prosumers from Sweden. The experiment aimed to estimate the effects of non-price interventions in residential energy conservation (e.g. providing information regarding consumption feedback, peer comparison and energy efficiency tips through a HER). To achieve its goals, each customer was randomly assigned to either the treatment or control groups, with the treatment group receiving a HER every two weeks.

The data used in this experiment are collected by the energy company CW for 867 customers from December 2020 till today (October 2023) including data regarding customers’ energy usage and demographics. A complete analysis of CW’s experiment is presented in deliverable D4.2 “Econometric analysis and robustness tests” and continues in D4.3 “Updated Econometric Methodologies and Robustness Tests”.

For estimating the impact of social norms on energy consumption, we follow a series of analytical steps for ensuring statistically robust results. These steps are presented in Figure 3:

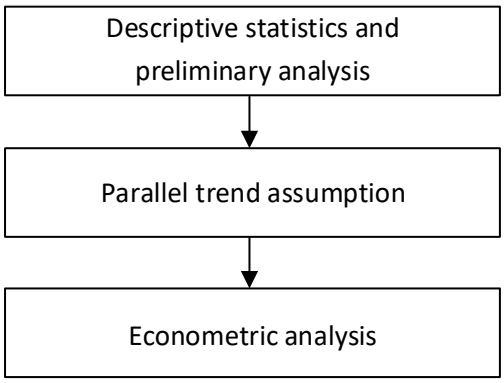

Figure 3: Methodological framework for estimating the impact of social norms on energy consumption

Descriptive statistics and preliminary analysis: This set of actions examine whether there are any imbalances in demographics between the treatment and the control group. We consider the pre-treatment period and estimate whether treatment and control groups are equal regarding any observable variables. For example, in the Swedish case we examine whether treatment and control groups are similar in terms of the average house size, electricity consumption and production. Furthermore, whether these two groups have similar average number of household members and amount of electricity purchased from the grid and sold back to the grid.

We also provide analytics regarding any seasonal and intraday patterns for the amount of electricity consumed, produced, sold to the grid and purchased from the grid. By doing so we get a better understating regarding the determinants of the impact of the HER on prosumers behavior. Examples of the previous actions are presented in code blocks Code block 1 to Code block 5. More analytics regarding the results will be provided in Deliverable D4.3.

Parallel trend assumption: In this step we examine whether the parallel trend assumption holds. This assumption is crucial for estimating any causal effect stemming from the treatment. It requires that if no treatment was to take place, then the difference between the two groups would be constant over time. Thus, any observed difference should be accounted to the treatment. There is no formal statistical test for the parallel trend assumption, and we mostly rely on creating and observing graphs. In Code block 5 we provide code for an informal statistical test.

Econometric analysis: The final step is conducting the econometric analysis. We provide detailed coding for estimating heterogenous treatment effects across groups and time following recent advances in econometric literature. The code refers to different econometric methods for the researcher to get robust results. The code also creates graphical representation of the results for the reader to get a better understanding. Code blocks Code block 7, Code block 8 and Code block 9 present three estimators: Rios Avilla estimator, Wooldridge DD estimator, and Callaway and Sant'Anna DD estimator. These estimators are used to estimate heterogeneous treatment effects across groups and time.

5. Examining the degree of attentiveness of customers when choosing an electricity contract

This section provides the coding used for examining the degree of attentiveness of customers when choosing an electricity contract based on data given by a large utility in Greece. A suboptimal contract choice is costly for consumers while it drives electricity consumption away from an optimal level. It should be noted that the fixed consumption and the social house tariff (SHT) contracts induce customers to consume within their contract limits since there is an overcharge for overconsumption. This feature will be used later in the analysis to estimate whether customers systematically underestimate their future electricity consumption across different groups.

There are three consumption tiers, the first is for an annual consumption up to 2,500kWh, the second is up to 4,000kWh and the third is up to 6,500kWh. If a customer stays as close as possible to her consumption limits in the selected tier, then she gains a discount relative to the regular tariff contract.

Some of the research questions we are trying to answer in this analysis are related to whether consumers choose the optimal contract, what is the effect of suboptimal choice on customers' loyalty and how many days does it take for a customer to switch contract. A detailed analysis will be presented in D4.3 since in this document we are presenting the code used to develop the methodological framework for this analysis.

As depicted in Figure 4, the coding is divided in 3 steps: a) Descriptive statistics, b) graphical representations, and c) econometric analysis.

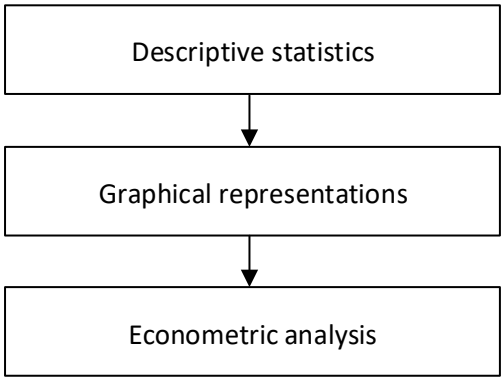

Figure 4: Methodological framework for examining the degree of attentiveness of customers when choosing an electricity contract

Descriptive statistics: This code snippet includes functions relative to the descriptive statistics of the given data. Descriptive statistics is an important step to start analysing the data since it provides useful insights about the data and the basic variables. For example, in Code block 10 we are calculating the sum of the aver_total_period_consumption (average total period consumption) variable given several constrains.

Graphical representations: Graphical representations, such as charts, graphs, and plots, are highly valuable in data analysis since they provide a clear and intuitive way to display data patterns, trends, and relationships that might not be immediately apparent from raw data. For example, in Code block 11 the provided STATA code is used to create a histogram of a variable called aver_total_period_consumption, with certain conditions, and customizing the appearance of the histogram.

Econometric analysis: This code snippet is used to examine customers' attentiveness when choosing one of the provided contract types. Thus, several steps are included. The analysis starts by first creating variables simulating the cost for each Genius contract ("Genius 1", "Genius 2", "Genius 3") and a regular pricing scheme. Then, Code block 12 conducts multiple regression analyses and calculates marginal effects to examine the relationship between contract perception, excess cost of Genius programs (contract options), and other variables such as contract duration.

6. Estimation of household energy consumption based on consumer's characteristics

The primary objective of this analysis is to assess the impact of subtle prompts in energy conservation, enhance consumer awareness about energy consumption, and provide practical tips and suggestions for more efficient electricity use. Thus, a set of approaches was established to achieve the project's goals, each focusing on both the categorical and the quantitative information provided by the PPC's 'myEnergyCoach' dataset. The most effective and reliable approach among the proposed strategies was the plan that performed only on quantitative data, providing various sub-services apart from the main objectives of the use cases. The following section offers a concise overview of the methodology's technical aspects of each sub-service, while an in-depth analysis will be presented in D4.3, "Updated econometric methodologies and robustness tests".

6.1 The concept

The analysis focused on the consumption information of the dataset that included any device's disaggregated consumption and the overall consumption report for each billing period. Observing the disaggregated consumption of each device for a specific period of time, enabled calculating the exact consumption of a consumer and also provided a closer examination of the consumer behavior through their individual devices.

This led to the main scope of providing:

- Consumption review for known and unknown consumers
- Exact consumption computation for known and unknown consumers
- Energy suggestion tips for known and unknown consumers

To provide all these three requirements, a set of operations was established as shown below:

- Efficiency rating
- Device frequency usage tips
- Device replacement tips
- Device frequency and replacement tips combination
- Device depreciation
- Consumer contract consumption and calculation

6.2 Efficiency rating

To achieve the stated objectives, the first step is to rely on the European Energy Labels, which include energy efficiency rating profiles. By processing the consumption period, its usage frequency and disaggregated consumption, the resulting numeric value acts as a consumption energy value and is matched with the available European Energy class labels to specify the class in which the device belongs, as shown in Code block 13.

6.3 Device Frequency Usage tips

The scope of this process was to verify and present the potential increase in energy efficiency achievable by a known consumer. This was achieved by reducing the frequency usage of each device by a fixed 20% and then calculating the impact in the efficiency rating of each device respectively, as shown also in Code block 14.

6.4 Device replacement tips

Another energy suggestion tip for the consumer would be a device replacement suggestion, which suggests that the user replaces their device with a more efficient one. A brief view of the implemented methodology can be observed in Code block 15.

6.5 Device Frequency and replacement tips combination

Additionally, a combination of the dual proposed operations (reducing device frequency or upgrading to a more efficient device) was applied. A brief summary of the technical part of the methodology can be observed in Code block 16.

6.6 Device Depreciation

Benefiting from the device replacement tips, depreciation insights were also available for the consumer to understand the true impact of replacing a device with a device of better class based on the European Energy Class labels. This information presents the consumer with the exact time period when the device would be entirely depreciated. The methodology is presented in Code block 17.

6.7 Consumer contract consumption and calculation

The dataset further provides the contract type of each consumer. The primary contract types are Daily and Nightly, which vary in terms of pricing for each kW consumed. For the Daily contract, the bill is calculated by multiplying the energy consumed by the current energy price. In contrast, the dataset records the amount of kW consumed during the day and night separately for the Nightly contract since the energy price varies. The methodology is presented in Code block 18.

7. Assessing the impact of behavioural insights in energy consumption using big data

As presented in D4.2 “Econometric analysis and robustness tests”, the main objective of use case 3 is to leverage the power of big data and machine learning to assess the impact of behavioural insights on energy consumption patterns. Thus, in the context of use case 3, the EVIDENT consortium has conducted two different analyses. The first analysis, an energy forecasting analysis, utilises data from CW's energy meters in customers' premises to develop a data-driven energy forecasting framework. The goal is to empower customers with insights for efficient energy cost planning. The second analysis investigates how household characteristics influence the impact of CW's interventions (from use Cases 1 and 2). It accomplishes this by building machine learning models for causal inference and assessing heterogeneity in causal effects in experimental studies.

The rest of the section provides a brief summary regarding the methodologies followed and is mainly focused on the technical aspects of the two analyses.

7.1 Building a residential energy consumption forecasting framework

In this analysis, we explore various forecasting scenarios to assess the effectiveness of ML models in predicting consumer energy consumption and production one day in advance. Additionally, we will compare the performance of ML models against linear forecasting models. This exercise will be conducted separately for energy consumption and production, since different factors may affect these variables[4].

The data utilised in this analysis spans from January 2017 to November 2022 and is sourced from CW's customers with photovoltaic (PV) installations on their premises. This dataset comprises daily records of energy consumption, energy purchased from the grid, energy sold, and energy generated by the PVs. To facilitate the execution of the forecasting scenarios, we also incorporate demographic information of CW's customers and, where available, weather conditions data (e.g., temperature, humidity, etc.) for the respective cities. By combining these datasets, we craft and execute various forecasting scenarios.

[4] The scenarios built, the features used, the methodology used and the results, are presented in detail in D4.2 “- Econometric analysis and robustness tests”. D4.3 will present insights based on additional forecasting models, models’ parameters and forecasting horizons for energy consumption and production forecasting.

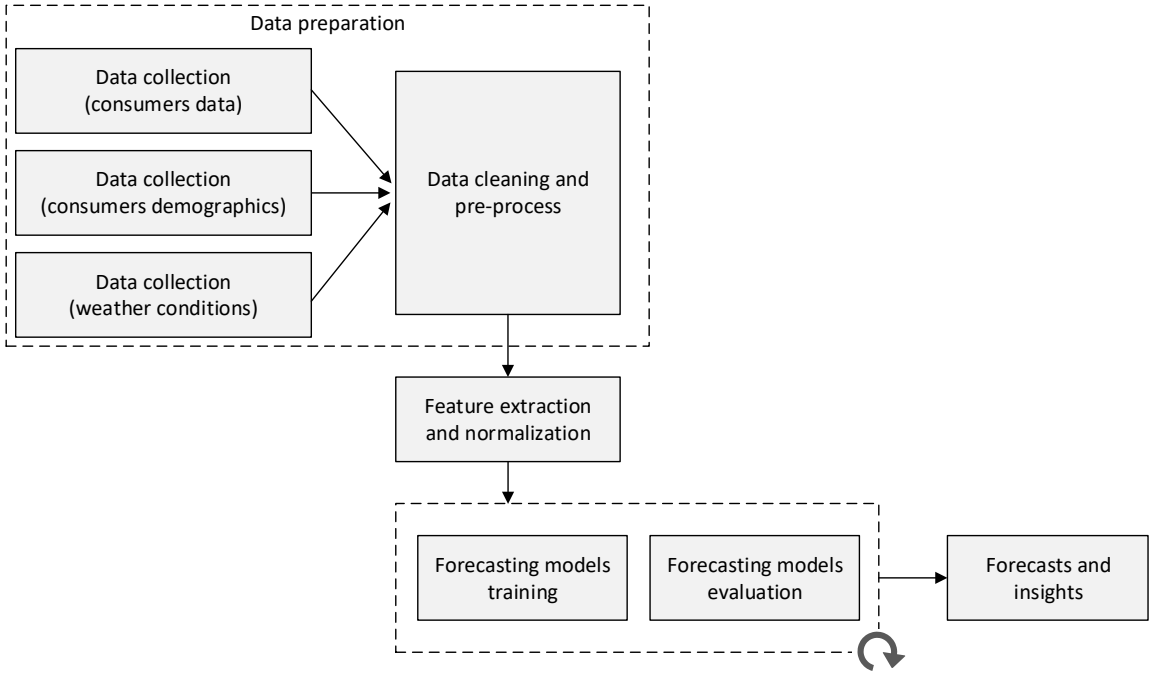

Figure 5: Methodological framework used in big data analytics for household energy consumption and production forecasting

The methodological framework used for developing the energy consumption forecasting framework is depicted in Figure 5. For each one of the discrete steps (data preparation, feature extraction, models training and model evaluation) a distinct snippet of code has been developed.

Data collection and pre-process snippet: This snippet contains actions relative to data collection, cleaning and preparation. Code block 19 is used to connect with a MySQL database, where the data are initially located, and executes a SELECT query to fetch them. In addition, Code block 20 snippet removes the outliers from the datasets by using an upper and lower threshold.

Feature extraction snippet: To create and normalise the features that will be used in the forecasting models, we are using a MinMax scalers using the sklearn library for the dependent (Y) and independent (Xs) variables, as shown in Code block 21. Additionally, we are using Code block 22 to create a correlation matrix between the model's features and plot the correlation matrix using a heatmap. After this step, the data are ready to be ingested to the machine learning models.

Model training and evaluation snippets: Finally, Code block 23 to Code block 29 are used to train and evaluate (in-sample/out-sample) the following forecasting models:

- Linear regressor (Abdi 2006)
- Lasso regressor (R. Tibshirani 1996)
- Elasticnet regressor (Zou and Hastie 2005)
- Random forest regressor (Boulesteix, et al. 2012)
- XGBOOST regressor (Mason, et al. 2000)
- Support vector regressor (Noble 2006)

- Multilayer preceptor regressor (Bishop 1996)

To do so, the python sklearn library is used. The parameters X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set.

7.2 Using machine learning to identify heterogenous treatment effect in experimental studies

The objective of this analysis is to use data-driven ML methodologies to estimate causal inference and heterogeneity in causal effects in experimental data. ML appears to be an attractive choice for this task, as it offers structured approaches for nonparametrically exploring heterogeneity and becomes particularly useful when a rich collection of baseline covariates is available. Thus, in this analysis we are trying to shed the characteristics of the participants (prosumers) that influence the conditional average treatment effect in CW's natural field experiment. Based on that, we will predict the individual treatment effects (ITE) and finally we will scale up the experiment new sample.

To achieve this, we utilise data collected from use cases 1 and 2 and leverage the honest causal forest model from the generalized random forests (GRF) R package[5] (Wager and Athey 2018). The focal point of our study is the natural field experiment conducted by the energy company CW in Sweden, which is part of the EVIDENT project. CW, at the time of delivering this report, has already provided 39 home energy reports to their clients, containing information about energy consumption, comparisons with similar households, and energy-saving tips. These HERs were distributed biweekly, commencing in December 2021.

The methodological framework used for developing a methodological framework for estimating heterogeneity in causal effects is depicted in Figure 6. For each one of the discrete steps (pre-process, models training and evaluation, prediction of ITE) a different snippet of code has been developed[6].

[5] More information about GRF package can be found here: https://grf-labs.github.io/grf

[6] The scenarios built, the features used, the methodology used and the results, are presented in detail in D4.2 "- Econometric analysis and robustness tests". D4.3 will present the results after the inclusion of additional 150 customers in the treatment group as the results of the analysis that took place.

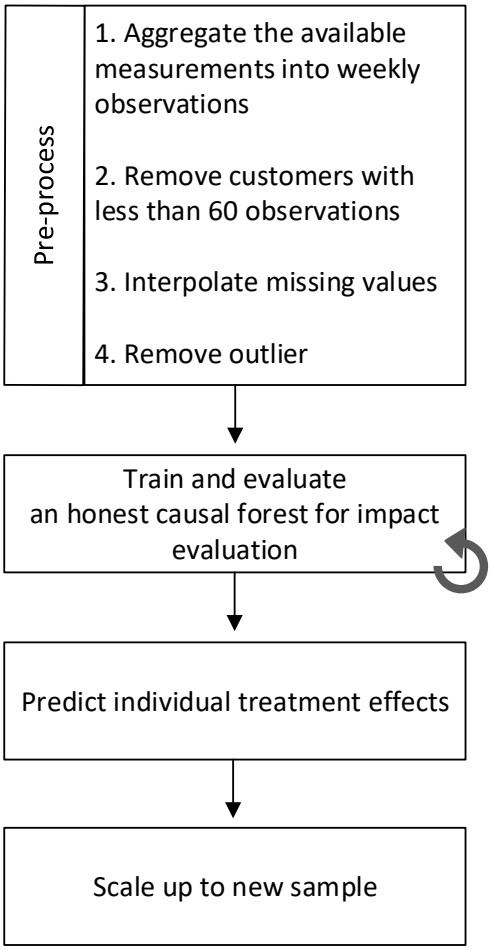

Figure 6: Methodological framework for big data analytics for optimized RCT

Pre-process snippet: This snippet contains actions relative to data aggregation (creating analysis scenarios), data cleaning and interpolation. Code block 30 is used to prepare the data for the GRF package. Through this snippet, we are creating a unified dataset for each customer where we declare Y (the dependent variable) as the difference between average weekly electricity consumed/bought for the two periods, pre and post experiment. We are also creating the W parameter (the treatment assignment) and finally we form the independent variables (X) where we use 4 statistical moments (average, standard deviation, skewness and kurtosis) for the pre-experiment period based on the dependent variable.

Model training and evaluation snippet: This snippet consists of several code block (Code block 31 to Code block 36) and it's responsible for training and evaluating a causal forest. Starting from Code block 31, it's used to train a causal forest using the GRF package by using different input parameters for the depend variable Y, the independent variables X and the treatment assignment W. Next, the Code block 32 visualises a random causal tree from the forest by using the runif function. The next two code blocks, Code block 33 and Code block 34, evaluates the causal forest by plotting all propensity scores in a and by using the 'mean forest prediction' and the 'differential forest prediction', respectively. In the context of propensity scores used to assess the randomisation of treatment assignment, we should not be able to

deterministically decide the treatment status of an individual based on its covariates, meaning none of the estimated propensity scores should be close to one or zero. When assessing the training of a causal forest using 'mean forest prediction' and 'differential forest prediction', a coefficient of 1 for the 'mean forest prediction' metric indicates that the mean forest prediction is accurate. Similarly, a coefficient of 1 for the 'differential forest prediction' suggests that the forest has successfully captured heterogeneity in the underlying signal.

The next code block, Code block 35 is used to calculate the conditional average treatment effect for a given causal forest. Function 'average_treatment_effect' used for this scope, outputs the conditional average treatment effect (CATE) value along with the standard error. Finally, Code block 36 examines the variable importance and outputs a simple weighted sum of how many times feature i was split on at each depth in the forest.

Prediction of ITE: This snippet consists of 5 code blocks (Code block 37 to Code block 41). The first block, Code block 37 is used to predict individual treatment effects given a trained causal forest. In this case, the 'predict' function takes as input the trained causal forest and a data frame containing features, allowing for the prediction of the dependent variable. By changing the features given, one can predict the ITE on the treatment group, the control group, or new data. Code block 38 uses the 'rank_average_treatment_effect' which can be used to check for heterogeneity in predictions, while Code block 39 is used to plot the relationships between a variable (in that case the four statistical moments used as the independent variables) and the predicted treatment effects. Code block 40 plots the predicted treatment effects sorted by their rank including the 95% confidence interval for the predicted treatment effects. For example, the researcher can predict the ITE for 100 unobserved customers and used this code block to plot the predicted ITE starting from the client with the greatest ITE to the one with the worst ITE[7]. Finally, Code block 41 plots the distribution of predicted treatment effects in an area chart while also drawing a red vertical line at the zero point.

[7] Depending on the use case, the definition of the 'best' and 'worst' ITE may vary. For instance, in the context of energy consumption, 'best' might refer to the client consuming the least energy. Conversely, when evaluating the amount donated to charity, the 'best' ITE could indicate the highest donation amount.

8. Estimation of consumers' willingness to pay for the repair of home appliances

The aim of this study, related to EVIDENT's use case 4, is to determine the impact of energy related financial literacy, demographic factors, and behavioural intention/attitude on decisions to repair or replace household appliances across resident types. Specifically, this study aims to answer the following research questions:

- What impact do financial, energy and environmental literacy have on the decision to repair or replace?
- What is the impact of the salience of financial information on the decision to repair or replace?
- What type of information impacts willingness to pay for a repair or replacement of an appliance? (financial, anticipated lifecycle or environmental)
- Does providing tips related to financial literacy enhance consumers ability to make better choices?
- What barriers and facilitators do individuals encounter when deciding to repair or replace an appliance (Qualitative analysis)
- What is the impact of the serious game on real life opportunities to make repair/replace decisions?

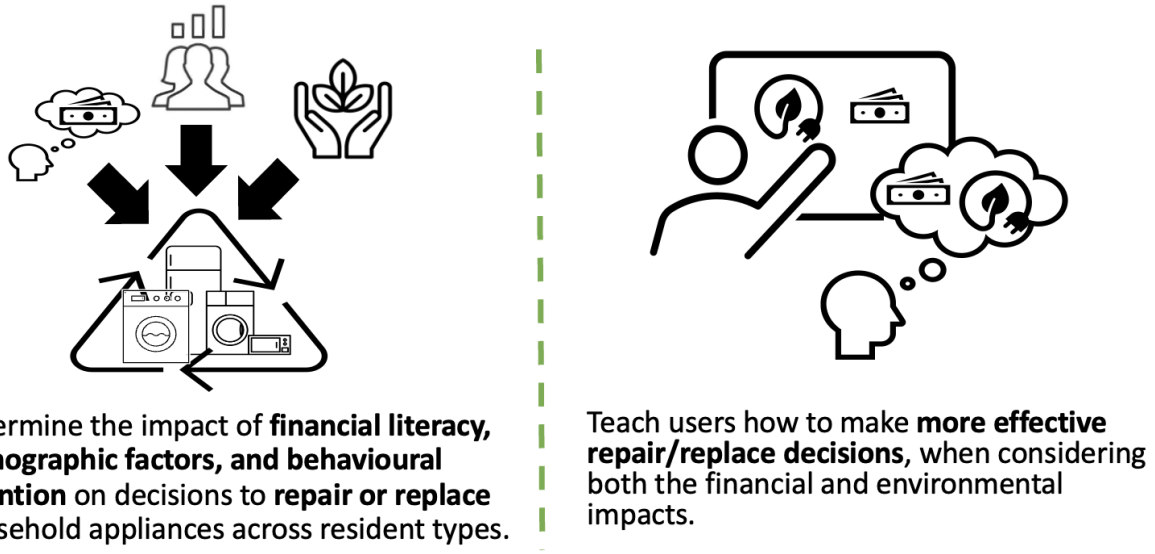

Figure 7: EVIDENT Serious Game dual focus

The research questions are investigated using a serious game whereby participants engage in a series of exercises using the online serious game, receiving valuable energy efficiency feedback on their behaviour within the game, and as part of this they complete a series of research tasks. The serious game itself is a life simulation type game within which, players are tasked with maintaining a home over the passing of time. Participants are assigned a role in keeping with their residential status (i.e. landlord, tenant or homeowner) and will be given an avatar to represent themselves within the game. The participants then move this avatar around their virtual home and complete a series of actions, all with the aim of

maintaining their avatars comfort, while also making sure their environmental impact doesn't get too high. The participants' actions in the game will be guided by indicative gauges, which will show their comfort, environmental impact (based upon Kw/h of energy use within the game) and finances. Comfort ratings will reduce should an avatars basic needs not be met (i.e. food, heat etc.) and is included as a means to motivate users to engage in actions in their home environment. At the end of the game, participants will be given a final score based on their environmental impact, comfort and finances and can see where they fall on a leader board.

Within the game, an appliance will break and the user must decide whether they would like to repair or replace the appliance. Users will be prompted to call a repairperson and will enter into a discussion about whether they would like to repair the broken appliance or can purchase a new appliance. For new appliances, differing levels of energy efficiency and cost will be available. Depending on the option selected, participants will then enter into a negotiation with the repairperson to determine their willingness to pay for a repair. For those who are landlords or tenants, additional discussions will occur, with tenants given the option to pay more rent or a small fee in exchange for a better appliance, and landlords given the option to accept more rent from tenants in exchange for a better rated appliance. Once a final choice is made users will be given some feedback on their decision and some advice on how to more easily determine whether to repair or replace a broken appliance. Users will then continue in the game, navigating more appliances that break. At the end of the game the points gained will be given as a total score and the user informed of their place on a leader board. Users will be advised also of where their score falls relative to the average.

Data collection: As outlined in D3.2 the EVIDENT serious game was hosted on the EVIDENT platform. Recruitment of participants was done using Prolific[8], due to the requirement to achieve a high response rate and the time required to play the game and gather the date (30 mins minimum). Data from the EVIDENT platform is exported in an Excel format, ready for data analysis.

Data analysis: Data analysis was performed using the statistical software R. R is a versatile statistical programming language and software environment used for data analysis and statistical modelling. It provides a wide range of tools and packages for manipulating, exploring, visualizing, and modelling data. Data analysis using R typically consists of the following sub-steps as presented in Figure 8:

[8] https://www.prolific.co

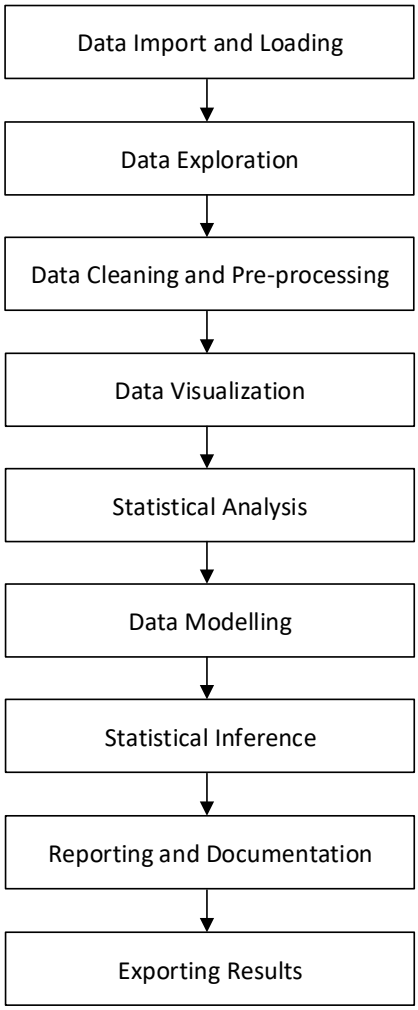

Figure 8: Methodological framework for estimating consumers' willingness to pay for the repair of home appliances

Data Import and Loading, Data Exploration, Data Cleaning and Pre-processing: Common functions for data import include **'read.csv()'**, **'read.table()'**, **'read.xlsx()'**, and database-specific functions. Once data is loaded, the dataset is explored. This includes summarizing data, checking for missing values, and understanding the structure of the dataset. Common functions for data exploration include **'summary()'**, **'str()'**, **'head()'**, **and 'tail()'**. Data cleaning involves handling missing values, outliers, and transforming variables as needed. Functions like **'na.omit()'**, **'na.exclude()'**, and **'scale()'** are used for data pre-processing. Code block 42 includes the data preprocessing and transformation actions performed on the collected data regarding this analysis.

Data Visualization, Statistical Analysis, Data Modelling: Data visualization can be performed, if needed, using ggplot2, lattice, and/or base R graphics. The statistical functions of R allow for hypothesis testing, regression analysis, ANOVA, clustering, etc. Functions like **'lm()'** for linear regression and **'t.test()'** for hypothesis testing are commonly used. Linear regression, logistic regression, decision trees, and random forests are performed using packages like **'glm()'**,

'randomForest()', and **'caret'**. Code block 43 is utilized for conducting data exploration and predictive modeling actions on the collected data.

Statistical Inference, Reporting and Documentation, Exporting Results: The estimation of confidence intervals and p-values for hypothesis testing is performed. R Markdown creates reproducible reports and documents that combine code, results, and narrative explanations. Results can be exported in various formats, such as CSV, Excel, PDF, or graphics files, for sharing or further analysis. Code block 44 conducts a statistical inference in R on the collected data.

9. Estimation of consumers' willingness to pay for more efficient energy home appliances

This study, as part of EVIDENT's use case 5, seeks to examine the impact of energy related financial literacy, demographic factors, environmental literacy and behavioural intention/attitude on discount rate and willingness to pay for more efficient household appliances. More specifically:

1) What impact do financial literacy, energy literacy, environmental concern have on implicit discount rates?
2) What impact do factors such as financial information (purchase price, operating cost), risk reduction, energy discounts and loans have on implicit discount rates for home appliances?
3) Does providing more salient financial information impact implicit discount rates and willingness to purchase more efficient home appliances?
4) For consumers who choose to purchase more efficient appliances, what impact do direct rebound rates have when choosing an appliance?

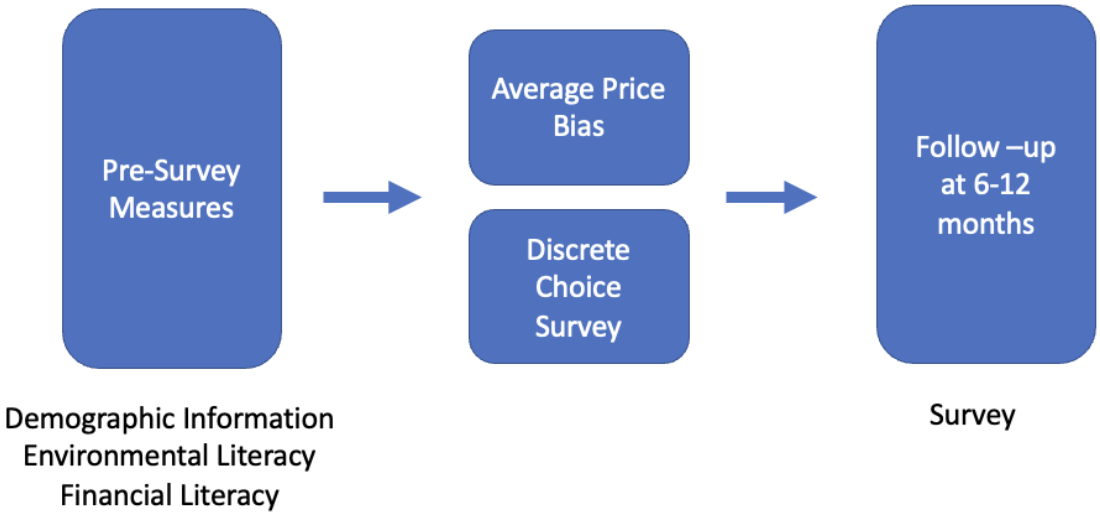

Figure 6: Methodological protocol for Use Case 5

Use Case 5 has two research components, the first is detailed herein focusing on the discrete choice survey, and the second is detailed in Section 10 (below) focusing on the average price bias portion of the use case. The pre-survey measures (gathering information on demographics, and analysing participants' environmental and financial literacy) are used for both of these research components of the use case. A follow-up survey is planned to gather information on any subsequent relevant home appliance purchases/behaviour.

Data collection: As outlined in D3.2 the data was collected leveraging participation from both the EVIDENT platform and Qualtrics survey software. Data from the both the EVIDENT platform and Qualtrics can be exported in a variety of formats including CSV, TSV, Excel, and XML.

Data analysis: Data analysis was performed using the statistical software R. R is a versatile statistical programming language and software environment used for data analysis and statistical modelling. It provides a wide range of tools and packages for manipulating, exploring, visualizing, and modelling data. Data analysis using R typically has already been presented in Section 8.

10. Assessing consumers' average price bias

This analysis aims to understand the factors influencing environmental literacy, environmental knowledge, and environmental behavior scores while considering demographics and financial factors. Understanding these relationships is essential for crafting targeted interventions and policies to promote environmental literacy and sustainable behavior.

For the data preparation 8 kind of scores were computed:

- **Financial Knowledge Score:** Computed from 6 questions. Correct responses to all 6 questions result in a perfect score of 6.
- **Financial Behavior Score:** Computed from 5 questions. Correct responses to all 5 questions result in a perfect score of 5.
- **Financial Attitude Score:** Computed from 5 questions that were transformed in Likert type scale (ranging from 1 to 5). To compute the financial attitude score, the average score is calculated based on participants' responses to these questions (with maximum of 5points).
- **Financial Literacy score**: Computed from the total of the Knowledge, Behaviour score and the average score across the attitude questions, giving a maximum of 16 points. The acceptable level of financial literacy is a minimum of 2/3 of the total.
- **Environmental Knowledge Score:** Computed from 8 questions. Correct responses to all 8 questions result in a perfect score of 8.
- **Environmental Behavior Score:** Computed from 7 questions. Correct responses to all 7 questions result in a perfect score of 7.
- **Environmental Attitude Score:** Computed from 7 questions that were transformed in Likert type scale (ranging from 1 to 5). To compute the environmental attitude score, the average score is calculated based on participants' responses to these questions. This score represents individuals' attitudes toward environmental issues, with higher values indicating more positive attitudes (with maximum of 5points).
- **Environmental Literacy score:** Computed from the total of the Knowledge, Behaviour score and the average score across the attitude questions, giving a maximum of 20 points. The acceptable level of environmental literacy is a minimum of 2/3 of the total.

The calculation of the financial and environmental knowledge, behaviour and attitude scores are presented in Code block 45, while the assessment of financial and environmental literacy levels is presented in Code block 46.

Visualisations though diagrams were created to explore the relationships between scores and demographics. Diagrams include mainly bar charts providing insights into the distribution and patterns within the data.

Multivariate Regression model was employed to assess how environmental literacy, knowledge, and behavior scores are influenced by various variables, including demographics (age demo, number of adults in home adults, number of children in home, income, employment status, home status) and financial literacy. The Multivariate Regression model is presented in Code block 47.

11. Conclusion

In the context of D4.4 “Analytical Usage Handbooks for Tools and Datasets”, the EVIDENT project consortium aimed to create a valuable handbook, offering ready to reference methodologies and code blocks for third-party researchers. This deliverable introduces the analytical tools developed throughout the project's lifecycle, specifically designed to analyse data in all five EVIDENT use cases.

Additionally, the report includes information about DIME, a framework developed by the World Bank to support high-quality research and provide best practices for researchers in organizing their research tools and data effectively.

D4.4 comprises seven technical sections, each corresponding to a specific analysis conducted during the project's lifecycle. Within each technical section, a brief presentation of the analysis's primary objectives, the methodology employed, and the tools developed are presented. The results of each analysis will be reported in D4.3 “Updated Econometric Methodologies and Robustness Tests”.

12. References

Abdi, H. (2006). The method of least squares.

Bishop, C. M. (1996). Neural Networks for Pattern Recognition. New York, NY, USA: Oxford Univ. Press.

Bjarkefur, K., Cardoso de Andrade, L., Daniels, B., & Jones, M. R. (2021). *Development Research in Practice: The DIME Analytics Data Handbook.* Washington, DC: World Bank. Retrieved from https://worldbank.github.io/dime-data-handbook/

Boulesteix, A.-L., Janitza, S., Kruppa, J., & König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2*(6), 493–507.

Christensen, G., & Miguel, E. (2018). Transparency, Reproducibility, and the Credibility of Economics Research. *Journal of Economic Literature, 56*(3).

Duvendack, M., Palmer-Jones, R., & Reed, W. R. (2017). What Is Meant by 'Replication' and Why Does It Encounter Resistance in Economics? *American Economic Review, 107*(5), 46–51.

Foster, A., Karlan, D., & Miguel, T. (2018, March 9). *Registered Reports:*. Retrieved July 2023, from Development Impact: https://blogs

Kerr, N. L. (1998). HARKing: Hypothesizing after the Results Are Known. *Personality and Social Psychology Review 2, 3*, 196–217.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. *in Advances in Neural Information Processing Systems 12*, 512–518.

McKenzie, D. (2012, October 28). *A Pre-analysis Plan Checklist*. Retrieved July 2023, from Development Impact: https://blogs.worldbank.org

Noble, W. S. (2006). What is a support vector machine? Springer Science and Business Media LLC.

Rossum, G. v., Warsaw, B., & Coghlan, N. (2001, July 5). *Style Guide for Python Code*. Retrieved from https://peps.python.org/pep-0008/

Stodden, V., Guo, P., & Ma, Z. (2013). Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. *PLoS One, 8*(6).

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288.

Vilhuber, L., Turrito, J., & Welch, K. (2020). *Report by the AEA Data.* AEA Papers and Proceedings 110.

Wager, S., & Athey, S. (2018). Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association, 113*(523), 1228-1242. doi:10.1080/01621459.2017.1319839

Wickham, H. (n.d.). *Tidyverse Style Guide*. Retrieved from https://style.tidyverse.org/

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301–320.

13. Appendices

13.1 Appendix 1: Efficient data management

To conduct effective data work in a team environment and avoid data losses and frustration between team members, good planning is required regarding the data owned and the tools developed. A team needs to decide on clear techniques for organizing the data folder, the structure of the data sets in the folder, and identifying the observations in the data sets.

Based on the DIME framework, the first step for productive workflows is adopting a data map, a template that will accompany your team throughout the project's lifetime, a valuable tool that helps in accurately and reproducibly linking all datasets associated with a project. The data map aims to organize three main aspects of your data work:

- **data analysis** which is the process of exploring your data to identify trends and results
- **data cleaning**, aiming to make a dataset easily understandable for the research team, external users and your analytical tools to ensure valid analysis
- **data management** which encompasses a range of actions and activities aimed at ensuring the quality, integrity, security, and accessibility of data throughout its lifecycle.

The data map consists of three discrete components:

1. A data linkage table,
2. A master dataset,
3. Data flow charts,

three tools used to continuously communicate the project's data requirements across the team members.

Data linkage table

The data linkage table aims to accurately link all datasets associated with the project in a reproducible manner. It's a list that holds all the datasets used in a project since, in complex projects that last several years, there might be multiple data sources where the data are collected or numerous partners that provide data. Over the years, the research team might change, and additional data may be available; thus, a data linkage table can be leveraged to help resolve errors in linking datasets[9].

Master dataset

Master datasets are data files that keep information about individual units for each level of observation (e.g., households, cities, companies, etc.). While being the second step of a data map, master data sets simplifies the combination of multiple datasets and reduce data management errors. As a must-follow

[9] A downloadable data linkage table template along with a brief explanation of the purpose and contents of each column can be found in the Data Linkage Table page here.

best practice, each observation should be accompanied by a unique identifier since the same unit might be present in multiple master datasets and merge actions are required. For example, in the EVIDENT project, the unique consumers' ID is used to combine demographic data such as household location, house size, etc., with their consumption measurements that are constantly updating. These two datasets may refer to the same unit ("the consumer") but differ in how often the data are updated or produced. The choice to keep the data in different master datasets was made to organize the provided information better and efficiently handle the exponential data size[10].

Data flow charts

A data flow chart is the last step of a data map that aims to provide useful information and visualise how individual datasets can be combined to create the datasets that will finally be used for analysis while providing a useful tool for communication between the research team's members. The data flow charts are closely related to both data linkage tables and master datasets since they describe how one can create the final analysis datasets using various intermediate datasets[11].

The DIME framework proposes a specific data folder structure (named DataWork folder) one can follow to achieve an efficient data management process, create productive workflows and achieve proper communication between the team members. The DataWork folder will eventually host every file related to the project's data, from files related to the project analysis (e.g. master files, questionnaire, documentation, programming code and analytical tools) and secondary data such as monitoring and administrative files. As a best practice, the DataWork folder is recommended to be adopted from the beginning of the project.

Figure 9 presents DIME's proposed DataWork folder template. DataWork consists of four main directories and on master Dofile[12]. It may also include README files and other documentation.

[10] Useful information about the master datasets, how and when to create can be found in the Master Dataset here.
[11] A data flow chart template, along with an explanation of the thinking of designing a data flow chart, is presented in the Data Flow Charts here.
[12] To support consistent folder organization, DIME Analytics created iefolder as a part of the STATA ietoolkit package. DIME provides a STATA package for data management actions; however, the same folder structure can be easily replicated with different programming languages or manually. The adoption of this structured folder organization is proposed in every research work.

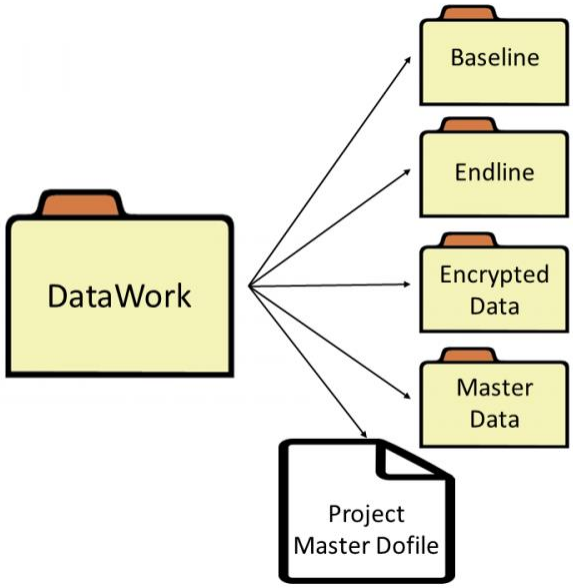

Figure 9: Example of DIME's DataWork folder template

Baseline folder: The scope of the baseline folder is to collect all baseline data, analytical scripts (e.g. do-files or others) and the outputs for this stage of the data collection process. Baseline data collection is a benchmark for the data collection before any intervention or treatment is implemented. The baseline folder may contain subfolders for the Datasets, the Dofiles (or other analytical scripts), the Outputs, Documentation and the Questionnaire.

Endline folder: The endline folder aims to store the data collected during the final phase of the data collection process. This includes the implementation of the designed intervention of the experiment. The data collected will be compared with the data collected during the baseline phase to evaluate the impact and effectiveness of the planned intervention. The folder structure is identical to the baseline folder structure. Between the baseline and the endline data collection stages, there might be additional stages (midline, interim or follow-up stages) that should be recorded using the same structure.

Encrypted data folder: The encrypted data folder should contain all personally identified data for each round of data collection. Therefore, this folder should contain as many subfolders as the data collection rounds (e.g. two in case of a baseline and an endline data collection stages). The encrypted data folder should also include a folder (or file) with all units' unique IDs that should be used during the data collection. For example, for the research unit named “John Doe”, a unique ID is given

"2F00924CE2E10CC8" and used to link John Doe's demographic data with his collected observations[13]. As its name declares, this folder should be encrypted to ensure data confidentiality and units' privacy.

Master data folder: The master data folder will store the master data sets for each unit of the research. A master dataset should contain information about individual units for each level of observation as described in the previous paragraphs.

Project Master Script (Dofile[14]): The scope of the project's master script is to provide a single code script to execute the data work for a project. The master script will execute all the other scripts of a project and acts as a map for the corresponding analytical steps. The master script will also establish an identical workspace between users by specifying settings, installing programs, and setting global variables. More or less is the starting point for every script, dataset and output every research team member can use to execute the data work for the whole project.

Finally, the DataWork might include README.md files used extensively in collaborative version control, such as GitHub, GitLab, Bitbucket, etc. The extension md stands for "Markdown", a lightweight markup language used to format text in a simple and readable way. The primary purpose of the README.md file is to provide a single point of information, including essential documentation, instructions and guidance to the research team members.

13.2 Appendix 2: Writing "high-quality" code

This section includes good-to-follow instructions about how researchers can write "high-quality" code. Some minimum requirements should be adhered to for a code to be considered "high-quality". These requirements refer to code readability and clarity, modularity and reusability, efficiency and performance and, of course, code documentation[15].

Some basic naming conventions should be employed for a code to be easy to read, understand, and maintain. This includes variable and function names and consistent indentation and formatting conventions. In addition, comments should be presented to explain complex functions or provide content for a function of a variable. Modularity and reusability are two core concepts when working with a team, and some blocks of code are repeated during the analysis. In simple terms, both concepts mean that the code should be organized into functions (or components) with a specific purpose and interface. For example, when performing the same action multiple times, it's better to create a helper function to

[13] To efficiently create a unique ID for each unit, the uuid module in Python, the runiform() function in STATA, and the uuid package in R can be utilized.
[14] Do-files are standard text files used by STATA. Instead, one can use R or Python as master scripts; this is related to the scope of the analysis or the personal preference of the research team. Of course, multiple files leveraging different programming languages can be used at the same time.
[15] There are other principles a code should contain to be considered "high-quality"; however, these principles (e.g. error handling and resilience, security, testability and reliability) are better suited for production code, software development and the industry. Of course, someone may argue that error handling and resilience may also be a core topic of research code or that researchers should consider the potential integration of their models into a production workflow (e.g. integration of forecasting models into a production system). The definition of "high-quality" code can vary depending on the area of interest, scope, industry standards, and so forth.

enhance reusability. This function should consider all proper parameters so code modularity to be achieved. That way, code reusability is promoted, duplication is reduced, and code maintainability is achieved. Efficiency and performance should be followed to minimize unnecessary computations and repetitions and avoid performance bottlenecks.

In addition, efficient coding has three major elements (a) structure, (b) syntax, and (c) style. The structure refers to the principles/logic followed to organize the code and code files, the syntax is the actual code used, which should be simple and readable, and style refers to the nonfunctional elements of code (e.g. name convention, etc.). Even if elements such as structure, syntax and style are subjective and each researcher can draw their lines, each language is associated with its style guides which are suggested to be followed. For example, in Python programming language, it's universally agreed to use the Python Enhancement Proposal 8 (PEP8) style guide[16] (Rossum, Warsaw, & Coghlan, 2001), while in R programming language, a well-recognized but unofficial style guide is Hadley Wickham's Tidyverse Style Guide[17] (Wickham, n.d.). In addition, while there is no official styling guide for the STATA programming language from StataCorp, DIME proposed several suggested guidelines through the DIME Analytics Data Handbook[18] (Bjarkefur, Cardoso de Andrade, Daniels, & Jones, 2021). Finally, the Statistical Package for the Social Sciences[19] (SPSS) has established certain conventions and recommendations to improve code readability and maintainability through the official community and the IBM corporation[20].

[16] More information about the Python programming language PEP8 style guide can be found here.
[17] More information about the R programming language style guide by Hadley Wickham can be found here.
[18] More information about the STATA programming language style guide by DIME can be found here.
[19] As its name reveals, SPSS is a statistical package developed by IBM, oriented toward social sciences.
[20] More information about the SPSS programming language style guide can be found here.

14. Code blocks

14.1 Code blocks for building the econometric analysis for randomized experiments

Programming language	STATA
Description of code block	The provided STATA code is using the codebook command to generate a summary or overview of the variable clientid based on specific conditions for other variables in the dataset. It restricts the analysis to a subset of data that meets certain criteria using the if condition. Variable groups indicates the intervention group each customers belongs to (the customers belongs to the treatment group if the group variable is 1 or to the control group otherwise). By changing the variable to be calculated and the conditions, one can generate different summaries for the given data.

```
1. codebook clientid if (consumption<10942 & consumption>-170 &      ///
     produced<=8454 & produced>=0 & bought<=12390         ///
     & bought>=0 & sold<=8000 & sold>=0 )              ///
     & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
     | numberofhersreceived == 32 | numberofhersreceived == 33   ///
     | numberofhersreceived == 34 | numberofhersreceived == 35   ///
     | numberofhersreceived == 36) & (climatezone==3 | climatezone==4) ///
      & groups == 1
```

Code block 1: Descriptive statistics for the clientid variable

Programming language	STATA
Description of code block	The provided STATA code uses the bysort and sum commands to calculate the sum of the variable consumption within specific subgroups defined by the variable groups. It applies a set of conditions to filter the data for the calculation. Variable numberofhersreceived indicates the number of the total HERs a customer received so far. By changing the variable to be calculated and the constrains, one can calculate different amounts for the given data.

```
1. bysort groups:sum consumption if ///
     (consumption<10942 & consumption>-170   ///
     & produced<=8454 & produced>=0 & bought<=12390         ///
     & bought>=0 & sold<=8000 & sold>=0)           ///
     & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
     | numberofhersreceived == 32 | numberofhersreceived == 33   ///
     | numberofhersreceived == 34 | numberofhersreceived == 35   ///
     | numberofhersreceived == 36)  ///
     & date <= 1954368000000, detail
```

Code block 2: Calculate consumption sum for the treatment and the control groups

Programming language	STATA
Description of code block	The provided STATA code uses the ttest command to conduct a t-test for the variable produced with specific conditions. The t-tests are conducted separately for different levels of the variable groups. By changing the variable to be calculated and the conditions, one can calculate different amounts for the given data.

```
1.  ttest produced    ///
      if (consumption<=7374 & consumption>=4 ///
      & produced_1<=7753 & produced_1>=0 & bought<=6610 ///
      & bought>=0 & sold<=6200 & sold>=0                    ///
      & date <= 1954368000000) & (numberofhersreceived == 0      ///
      | numberofhersreceived == 30 | numberofhersreceived == 31     ///
      | numberofhersreceived == 32 | numberofhersreceived == 33     ///
      | numberofhersreceived == 34 | numberofhersreceived == 35),  ///
      by(groups)
```

Code block 3: Conduct a t-test for the variable produced with specific conditions

Programming language	STATA
Description of code block	The provided STATA code creates a two-way histogram plot to visualize the distribution of the variable consumption under different conditions and within different groups.

```
1.  twoway (hist consumption if groups==1 &  ///
      (numberofhersreceived == 31 | numberofhersreceived == 32 | numberofhersreceived == 33
    ///
      | numberofhersreceived == 34 | numberofhersreceived == 35 | numberofhersreceived == 36),
    ///
      frac lcolor(navy%30) fcolor(navy%30)) ///
      (hist consumption if groups==0 & numberofhersreceived==0, ///
      frac lcolor(red%70) fcolor(none)) ///
      if (consumption<10942 & consumption>-170   ///
      & produced<=8454 & produced>=0 & bought<=12390         ///
      & bought>=0 & sold<=8000 & sold>=0) & date <= 1954368000000, ///
      legend(off) xtitle("Treatment (red: Control)") xscale(titlegap(*10)) ///
      yscale(titlegap(*6)) xlabel(, format(%13.0fc))
```

Code block 4: Create a two-way histogram plot to visualize the distribution of the variable consumption

Programming language	STATA
Description of code block	The code first generates a binary variable indicating post-treatment based on a specific date. It then calculates the mean consumption for specific conditions, collapses the data, reshapes it to a wide format, and finally creates a two-way line plot to visualize consumption over hours, distinguishing between pre-treatment and post-treatment periods. The value of the variable groups (value 1) indicates that the following process is performed for the treatment group. One can change the value to 0 to perform the process for the control group.

```
1.  gen posttreatment=(date>1954368000000)
2.
3.  preserve
4.  collapse (mean) consumption if (consumption<=14400 & consumption>0) & ///
5.     (numberofhersreceived==0 | numberofhersreceived==30 | numberofhersreceived==35) & ///
6.     groups==1, by(posttreatment hours)
7.
8.  reshape wide consumption, i(hours) j(posttreatment)
9.  graph twoway connect consumption * hours,  ///
       ylabel(, format(%13.0fc) labsize(small)) ytitle("Consumption (w/h)", ///
       size(small)) yscale(titlegap(*10)) ///
       xlabel(1 3 5 7 9 11 13 15 17 19 21 23, format(%13.0fc) labsize(small)) ///
       xtitle("Time of the day", size(small)) xscale(titlegap(*10))  ///
       legend(cols(1) region(lwidth(none)) ring(0) bmargin(small) size(small) position(11) ///
       order(1 "pre-treatment period" 2 "post-treatment period")) ///
       lcolor(red navy) msize(small small) msymbol(T d) mcolor(gs10 gs10*0.8)   ///
       plotregion(margin(tiny))  plotregion(color(white)) graphregion(fcolor(white))
10.
11. restore
```

Code block 5: Examining consumption seasonal patterns between pre- and post-treatment period

Programming language	STATA
Description of code block	The code first formats the date variable, calculates the mean consumption based on specific conditions, collapses the data, reshapes it to a wide format, and then creates a two-way line plot to visualize consumption over dates for different groups. In the second part, the code estimates a fixed-effects panel data regression model to test the parallel trend assumption by analyzing the marginal effects of date on consumption for different levels of the groups variable. The marginsplot command visualizes these marginal effects. This is commonly used in difference-in-differences analyses to assess if the treatment and control groups had parallel trends before the treatment was introduced.

```
1.  preserve
```

```
format date %tcDD_Mon_CCYY

collapse (mean) consumption if (consumption<10942 & consumption>-170 &          ///
   produced<=8454 & produced>=0 & bought<=12390          ///
   & bought>=0 & sold<=8000 & sold>=0 )                  ///
   & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
   | numberofhersreceived == 32 | numberofhersreceived == 33   ///
   | numberofhersreceived == 34 | numberofhersreceived == 35   ///
   | numberofhersreceived == 36) & (climatezone==3 | climatezone==4) ///
    & (date <= 1954368000000), by(groups date)

reshape wide consumption, i(date) j(groups)
graph twoway connect consumption* date if date <= 1954368000000,  ///
   ylabel(, format(%13.0fc) labsize(small)) ytitle("Consumption (w/h)", ///
   size(small)) yscale(titlegap(*10)) ///
   xlabel(, labsize(tiny)) xtitle("Date", size(small)) xscale(titlegap(*10))        ///
   legend(cols(1) region(lwidth(none)) ring(0) bmargin(small) size(small) position(1) ///
   order(1 "Control group" 2 "Treatment group")) ///
   lcolor(red navy) msize(vtiny vtiny) msymbol(T d) mcolor(gs10*0 gs10*0)   ///
   plotregion(margin(tiny)) plotregion(color(white)) graphregion(fcolor(white))

restore

xtreg consumption i.groups#c.date if (consumption<10942 & consumption>-170 &          ///
   produced<=8454 & produced>=0 & bought<=12390          ///
   & bought>=0 & sold<=8000 & sold>=0 )                  ///
   & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
   | numberofhersreceived == 32 | numberofhersreceived == 33   ///
   | numberofhersreceived == 34 | numberofhersreceived == 35   ///
   | numberofhersreceived == 36) & (climatezone==3 | climatezone==4) ///
   & (date <= 1954368000000),      ///
   fe vce(cluster clientid)
margins groups, dydx(date)
marginsplotmarginsplot
```

Code block 6: Testing if the parallel trend assumption holds

Programming language	STATA
Description of code block	The code utilizes a Rios Avilla estimator (using the reghdfe command) with fixed effects to analyze the relationship between the dependent variable consumption and the independent variable htvar. The variable htvar is created by grouping observations based on the variables hours and monthindex. The estimation considers various conditions and uses fixed

	effects to account for individual/group-specific effects. Additionally, clustered standard errors are utilized for robust standard error estimation.

```
*************************************************************************
/*
|
|  Rios Avilla proposed estimator
|
*/
*************************************************************************

reghdfe consumption i.htvar ///
  if (consumption<10942 & consumption>-170 &   ///
  produced<=8454 & produced>=0 & bought<=12390   ///
  & bought>=0 & sold<=8000 & sold>=0 )   ///
  & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
  | numberofhersreceived == 32 | numberofhersreceived == 33   ///
  | numberofhersreceived == 34 | numberofhersreceived == 35   ///
  | numberofhersreceived == 36) ///
  & (climatezone==3 | climatezone==4), ///
  abs(clientid ftvar) cluster(clientid)
```

Code block 7: Estimating heterogenous treatment effects across groups and time using a Rios Avilla estimator

Programming language	STATA
Description of code block	The code performs treatment effect estimation using utilizes a Wooldridge DD estimator (using the jwdid command). Then it calculates and visualizes marginal effects over periods and treatment subgroups, and saves the resulting graphs.

```
*************************************************************************
/*
|
|  Wooldridge DD estimator
|
*/
*************************************************************************

***************** Amount Consumed *****************
qui: jwdid consumption ///
  if (consumption<10942 & consumption>-170 &  ///
  produced<=8454 & produced>=0 & bought<=12390        ///
  & bought>=0 & sold<=8000 & sold>=0 )              ///
  & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
  | numberofhersreceived == 32 | numberofhersreceived == 33   ///
```

```
    | numberofhersreceived == 34 | numberofhersreceived == 35   ///
    | numberofhersreceived == 36) ///
    & (climatezone==3 | climatezone==4),   ///
    i(clientid) t(weekindex) gvar(first_treat)

estat simple
estat group
estat calendar

*Estimate marginal effects over the periods (jwdid)
tempvar aux
qui:bysort `e(ivar)':egen `aux'= min(`e(tvar)') if e(sample)
qui:clonevar __calendar__ = `e(tvar)' ///
  if __etr__== 1 & `aux'<`e(gvar)'

margins, subpop(if __etr__== 1) at(__tr__=(0 1)) ///
  over(__calendar__) noestimcheck contrast(atcontrast(r)) `options'

marginsplot, yline(0, lcolor(red) lwidth(vthin))  ///
  graphregion(color(white)) title("")   ///
  plotopts(msymbol(o) mcolor(dknavy%80))   ///
  ciopts(lwidth(thin) lcolor(dknavy%40))   ///
  ytitle("Impact on consumption (watts)", height(8))    ///
  xtitle("Date", height(6))
graph save calendar_consumed.gph, replace
capture drop __calendar__

*Estimate marginal effects over treatment subgroups
tempvar aux
qui:bysort `e(ivar)':egen `aux'=min(`e(tvar)') if e(sample)
capture drop __group__
qui:clonevar __group__ =  `e(gvar)' if __etr__==1 & `aux'<`e(gvar)'
margins, subpop(if __etr__==1) at(__tr__=(0 1)) over(__group__) ///
  noestimcheck contrast(atcontrast(r)) `options'

marginsplot, yline(0, lcolor(red) lwidth(vthin))  ///
  graphregion(color(white)) title("")    ///
  plotopts(msymbol(o) mcolor(dknavy%60))   ///
  ciopts(lwidth(thin) lcolor(dknavy%60))    ///
  ytitle("Impact on consumption (watts)", height(6))   ///
  ylabel(, format(%13.0fc) labsize(small)) ///
  xlabel(22620 "Early group" 22697 "Later group", labsize(small)
  angle(vertical)) xtitle("Group") xsize(5.5in)
cap graph save group_consumed.gph, replace
capture drop __group__
```

```
37.
38. graph combine calendar_bought.gph calendar_consumed.gph,
39. iscale(0.5) ycommon
40. graph save combined.gph, replace
```

Code block 8: Estimating heterogenous treatment effects across groups and time using a Wooldridge DD estimator

Programming language	STATA
Description of code block	The code performs treatment effect estimation using utilizes a Callaway and Sant'Anna DD estimator (using the csdid command). Then displays summary statistics and visualizes the treatment effects using a plot. The plot is further customized to include specific labels, lines, and titles for clarity and interpretation.

```
1.  ********************************************************************
2.  /*
3.  |
4.  |  Callaway and Sant'Anna DD estimator
5.  |
6.  */
7.  ********************************************************************
8.
9.  csdid consumption if (consumption<10942 & consumption>-170 &  ///
      produced<=8454 & produced>=0 & bought<=12390         ///
      & bought>=0 & sold<=8000 & sold>=0 )                 ///
      & (numberofhersreceived == 0 | numberofhersreceived == 31   ///
      | numberofhersreceived == 32 | numberofhersreceived == 33   ///
      | numberofhersreceived == 34 | numberofhersreceived == 35   ///
      | numberofhersreceived == 36) & (climatezone==3 |
      climatezone==4), ///
      i(clientid) t(weekindex) gvar(first_treat) drimp
10.
11. estat simple
12. estat group
13. estat calendar
14.
15. csdid_plot,
16. addplot: , xlabel(1(12)85) yline(0, lcolor(red) lpattern(dash)
17. lwidth(medthin)) graphregion(color(white))   ///
      ytitle("ATT (watts)", height(8))       ///
      xtitle("Period",  height(6)) ylabel(, labsize(small)
18. format(%13.0fc))  ///
      xlabel(,  labsize(small)) title("")
```

Code block 9: Estimating heterogenous treatment effects across groups and time using a Callaway and Sant'Anna DD estimator

14.2 Code blocks for examining the degree of attentiveness of customers when choosing an electricity contract

Programming language	STATA
Description of code block	This code block is used to extract some basic descriptive statistics from the data provided. The code calculates the sum of the aver_total_period_consumption variable given several constrains regarding the contract type (variable contract), the amount of customer's meters (variable meters_by_customers), the time period (variable year) and the average total period consumption (variable aver_total_period_consumption). By changing the variable to be calculated and the constrains, one can calculate different amounts for the given data.

```
1. sum aver_total_period_consumption if (contract == "Proteteria Home-Genius 1" ///
   | contract == "Proteteria Home N-Genius 1") & (meters_by_customers==1 | ///
   meters_by_customers==2) & year>2015 & year<2022 ///
   & aver_total_period_consumption<24.58 & aver_total_period_consumption>0.17
2.
3. sum aver_total_period_consumption if (contract == "Proteteria Home-Genius 2" ///
   | contract == "Proteteria Home N-Genius 2") & (meters_by_customers==1 | ///
   meters_by_customers==2) & year>2015 & year<2022 ///
   & aver_total_period_consumption<28.11 & aver_total_period_consumption>0.54
4.
5. sum aver_total_period_consumption if (contract == "Proteteria Home-Genius 3" ///
   | contract == "Proteteria Home N-Genius 3") & (meters_by_customers==1 | ///
   meters_by_customers==2) & year>2015 & year<2022 & ///
   aver_total_period_consumption<44.78 & aver_total_period_consumption>3.33
```

Code block 10: Descriptive statistics for average total period consumption

Programming language	STATA
Description of code block	The provided STATA code is creating a histogram of a variable called aver_total_period_consumption, with certain conditions, and customizing the appearance of the histogram. Variable Genius_chosen refers to customer's selected contract type. The part **by(Genius_chosen, cols(1) note(""))** creates separate histograms for different levels of the variable Genius_chosen. One can create different histograms by changing the variable of interest and the constrains.

```
1. histogram aver_total_period_consumption if aver_total_period_consumption >0.17 &    ///
     aver_total_period_consumption <44.78 & Genius_chosen!=0,                          ///
     by(Genius_chosen, cols(1) note("")) xlabel(1(4)44)                                ///
     xtitle("Daily electricity consumption (kWh)")                                     ///
```

```
    plotregion(color(white))  graphregion(fcolor(white) style(histogram))                ///
    xline(6.3, lpattern(dash) lcolor(red) lwidth(thin))                                   ///
    xline(9.854, lpattern(dash) lcolor(red) lwidth(thin))                                 ///
    xline(15.86, lpattern(dash) lcolor(red) lwidth(thin))                                 ///
    bfcolor(navy) blcolor(navy*0.1)                                                       ///
    text(0.14 0.2  "Regular Tariff"                                                       ///
       "opt. area", place(e) size(.15cm))                                                 ///
    text(0.14 6.5  "Tier-1"                                                               ///
       "opt. area", place(e) size(.15cm))                                                 ///
    text(0.14 11 "Tier-2"                                                                 ///
       "opt. area" , place(e) size(.15cm))                                                ///
    text(0.14 17   "Tier-3"                                                               ///
       "opt. area" , place(e) size(.15cm))
```

Code block 11: Creating a histogram for the variable aver_total_period_consumption

Programming language	STATA
Description of code block	The code first creates an indicator variable for over- and underestimation based on the customer's consumption and the selected Genius program. Thus, the variable cons_perception will be 0 when the selected contract is consistent with the consumed energy, 1 when the energy consumed by the customer is far less than the upper limit provided by the contract and 2 in the case the energy consumed by the customer is far more than the upper limit provided by the contract. Then, the code conducts multiple regression analyses and calculates marginal effects to examine the relationship between contract perception, excess cost of Genius programs, and other variables on contract duration and meter status. The regressions and marginal effects are tailored to specific conditions and interactions within the dataset. Finally, the variables presented in this code are the following: total_day_difference - a contract duration in days (e.g. 500 days) DTM2 – square meters of the house excess_cost_genius - excess cost from signed to the suboptimal contract Genius_chosen - chosen Genius-type contract

```
1.  ********Regression on contract duration
2.  ****First create an indicator variable for over- and underestimation
3.  gen cons_perception=0 if Genius_mismatch==0 & Geniuscontract==1
4.  replace cons_perception=1 if Genius_mismatch<0 & Geniuscontract==1
5.  replace cons_perception=2 if Genius_mismatch>0 & Geniuscontract==1
6.  label var cons_perception "0 for optimal contracts, 1 indicates overestimation, ///
      2 indicates underestimation"
7.
8.  ********Regression on contract duration
```

```
*Equation 1 Table 5
xtreg total_day_difference i.cons_perception i.year i.month DTM2 ///
  if year>2018 & Geniuscontract==1, fe vce(robust)

*Equation 2 Table 5
xtreg total_day_difference i.cons_perception#c.excess_cost_genius i.year i.month DTM2 ///
  if Geniuscontract==1, fe vce(robust)

*Equation 3 Table 5
xtreg total_day_difference i.cons_perception#i.Genius_chosen#c.excess_cost_genius ///
i.year i.month DTM2 if year>2018, fe vce(robust)

margins i.cons_perception#i.Genius_chosen
marginsplot
xtprobit meterStatus_ind i.cons_perception##c.excess_cost_genius i.year i.month   ///
  if Geniuscontract==1 & contract_switch_by_customer==1

margins i.cons_perception, atmeans
```

Code block 12: Indicate the optimal contract and the degree of attentiveness of customers

14.3 Code blocks for the estimation of household energy consumption based on consumer's characteristics

Programming language	Python
Description of code block	Calculate the class of device. This is applied to every device respectively. **Total_weeks** represents the weeks of the consumption of the specific device. It is computed by dividing the days with the sum of the days a week has. Next, the actual usage frequency (**total_cycles**) of the device used per unit is calculated by multiplying the total weeks with the frequency submitted in the dataset. Next, the consumption table (**specific_device_consumption**) is calculated that multiplies all energy classes with the actual usage of the device. Afterwards, the disaggregated value of the device is matched with the consumption levels and is archived in a specific row where is the next smaller consumption in the list. Finally, the efficiency (**efficiency_counter**) is calculated by measuring the distance from the minimum consumption. This number represents the energy label class from 0-8 with 0 presenting class (-) and 8 presenting class (A+++).

```
total_weeks = days_of_consumption[0] // 7
total_cycles = total_weeks * frequency_value
specific_device_consumption_levels.iloc[0,:] =
  specific_device_consumption_levels.iloc[0,:]*total_cycles
index = np.where(specific_device_consumption_levels  > dissagregated_value)[0]
```

```
5. efficiency_counter = len(index)
```

Code block 13: The main function of the methodology to calculate the consumption efficiency of a device and archive it into a specific class

Programming language	Python
Description of code block	Calculate the Consumption of a device with a new frequency device usage. This is applied to every device respectively. **Total_weeks** represents the weeks of the consumption of the specific device. It is computed by dividing the days with the sum of the days a week has. Next, the **suggested_frequency** is computed by reducing the default frequency from the dataset by 20%. Next, the consumption per week (**efficient_per_week**) is found by dividing the disaggregated value of the device with the multiplication of the total weeks and the default frequency of the device. As such, the new disaggregated consumption (**efficient_con_per_period**) is computed by multiplying the consumption per week, the total weeks, and the suggested frequency.

```
1. total_weeks = days_of_consumption[0] // 7
2. suggested_frequency = frequency_value[0] - frequency_value[0] * 20 / 100
3. efficient_con_per_week = dissagregated_value / (total_weeks * frequency_value[0])
4. efficient_con_per_period = efficient_con_per_week * (total_weeks) * suggested_frequency
```

Code block 14: The main function of the methodology to calculate the consumption of a device by reducing the frequency usage by 20%

Programming language	Python
Description of code block	Calculate the Consumption of a device with a new device class label. This is applied to every device respectively. **efficient_label** represents the European label that the device was with a class increased to the right better class (e.g. B to A class). Immediately, this label is matched with the consumption table that it has been computed before for the specific device, resulting to a more efficient consumption profile (**upgrade_con**). The new disaggregated consumption (**upgrade_dissagregated_value**) is computed by multiplying the new consumption profile (**upgrade_**con) with the days of consumption and the default device frequency usage. Finally, the efficiency (**new_grade_counter**) is calculated by measuring the distance from the minimum consumption. This number represents the energy label class from 0-8 with 0 presenting class (-) and 8 presenting class (A+++).

```
1. efficient_label = grade_consumptions[len(index[0]) + 1]
2. upgrade_con = grade_specific_device_consumption_levels[len(index[0]) + 1]
3. upgrade_dissagregated_value = upgrade_con * frequency_value[0] * days_of_consumption[0]
4. index = np.where(specific_device_consumption_levels > upgrade_dissagregated_value)
```

```
5. new_grade_counter = len(index[0]) + new_grade_counter
```

Code block 15: The main function of the methodology to calculate the consumption of a device by upgrading its energy class to its next greater class

Programming language	Python
Description of code block	Calculate the Consumption of a device with a new device class label and with a reduced frequency. This is applied to every device respectively. This is applied to every device respectively. **Efficient_upgrade_value** represents the new disaggregated consumption which is computed by multiplying the new consumption profile (**upgrade_con**) with the days of consumption and the suggested device frequency usage. Finally, the efficiency (**efficiency_new_grade_counter**) is calculated by measuring the distance from the minimum consumption. This number represents the energy label class from 0-8 with 0 presenting class (-) and 8 presenting class (A+++).

```
1. efficient_upgrade_value = upgrade_con * days_of_consumption[0] * suggested_frequency
2. index = np.where(specific_device_consumption_levels > efficient_upgrade_value)
3. efficiency_new_grade_counter = len(index[0]) + efficiency_new_grade_counter
```

Code block 16: The main function of the methodology to calculate the consumption of a device by upgrading its energy class to its next greater class and by reducing the frequency usage by 20%

Programming language	Python
Description of code block	Calculate the consumption contract based to the suggestions tips. This is applied to every consumer respectively. Each consumer provides a set of consumption periods. Thus, a percentage dor each consumption in the day is applied by dividing the total days with the total consumption in the day. Thus, a **avg_day** is provided that represents the average consumption in a single day. The average consumption in the night is also applied by subtracting the mean daily consumption with 1 (**avg_night**). As such, for the consumers that have nightly consumption, the bill will be split into multiplying the average daily consumption with the total consumption in order to get the respective portion and then is multiplied by the nightly energy price (**Nightly_Consumption_overall**). Additionally, the rest average consumption is added by calculating the bill in the daily usage with the same concept. The **Daily_Consumption_overall** is simply multiplied by the whole desagregated consumption value and its daily energy price.

```
1. day_percentage = (day_cons_list / total_cons_list)
2. avg_day = day_percentage.mean()
3. avg_night = 1 - avg_day
4.
5. Nightly_Consumtpion = [ avg_night, 0.114, avg_day, 0.155]
```

```
6. Nightly_Consumption_overall = total_con * Nightly_Consumtpion[0] * Nightly_Consumtpion[1] +
     total_con * Nightly_Consumtpion[2] * Nightly_Consumtpion[3]
7.
8. Daily_Consumption = [1,0.155]
9. Daily _Consumption_overall = total_con* Daily_Consumption [0]* Daily_Consumption [1]
```

Code block 17: The main function of the methodology to calculate the energy bill of the consumer with respect to the suggestion tips that have been followed for each device

Programming language	Python
Description of code block	Calculate the depreciation by replacing a device. This is applied to every device respectively. For each device the default consumption and its greater class device consumption is calculated by multiplying them with the daily energy price (**original_consumption, replacement_consumption**). Then their difference in the price is computed (**difference_price**) which is the input as a divider in the next calculation that finds the number of consumption periods needed to cover the actual price of the obtained device (**number_of_con_periods**). Afterwards the days needed to depreciate the device is applied by multiplying the initial consumption period with the number of consumption periods needed to cover this price (**Depreciation_occurance**). Finally, this values that represents days, it is converted into Years-Months-Days to make the depreciation suggestion more consumer-friendly (**output_period**).

```
1. original_consumption = round(device_consumption, 2) * 0.155
2. replacement_consumption = round(new_device_consumption, 2) * 0.155
3. difference_price = original_consumption - replacement_consumption
4. number_of_con_periods = device_price_value // difference_price + 1
5. Depreciation_occurance = int(consumption_period) * number_of_con_periods
6. output_period = days_to_years_months_days(int(Depreciation_occurance))
```

Code block 18: The main function of the methodology to calculate the exact period on when depreciation will be completed

14.4 Code blocks for building a residential energy consumption forecasting framework

Programming language	Python
Description of code block	This code block connects to a MySQL database and executes a data extraction query. It returns the headers of the data and all data matched the SQL query.

```
1. def fetch_table_data(table_name, lowdate_limit, upperdate_limit):
2.     cnx = mysql.connector.connect(
```

```
        host='DB_HOST', database=DB_NAME', user=' DB_USER', password='PWD'
    )

    cursor = cnx.cursor()
    query = "SELECT * FROM " + table_name + "' ORDER BY HOUR DESC"
    cursor.execute(query)

    header = [row[0] for row in cursor.description]
    rows = cursor.fetchall()

    cnx.close()

    return header, rows
```

Code block 19: Connect to a MySQL database and execute a SELECT query

Programming language	Python
Description of code block	This code block calculates the 10% and 90% percentiles of the consumption measurements and removes all records smaller than the 10% percentile or greater than the 90% percentile.

```
Q1 = df['totalconsumption'].quantile(0.10)
Q3 = df['totalconsumption'].quantile(0.90)

df = df.loc[(df['totalconsumption'] >= Q1) &(df['totalconsumption'] <= Q3)]
```

Code block 20: Remove outliers from consumption

Programming language	Python
Description of code block	This code block creates two MinMax scalers using the sklearn library for the dependent (Y) and independent (Xs) variables. Train and test variables are two pandas dataframes for the train and the test sets respectively. The code replaces the initial data with the scaled.

```
from sklearn.preprocessing import MinMaxScaler

Y = "totalconsumed"
numeric_features = ["TC1", "TC2", "B1","B2", "PR1", "PR2"]

scaler_y = MinMaxScaler()
train[Y] = scaler_y.fit_transform(train[[Y]])
test[Y] = scaler_y.transform(test[[Y]])

scaler_x = MinMaxScaler()
train[numeric_features] =
```

```
12. scaler_x.fit_transform(train[numeric_features])
13. test[numeric_features] = scaler_x.transform(test[numeric_features])
```

Code block 21: Scale dependent and independent variables using sklearn and pandas libraries

Programming language	Python
Description of code block	Create a correlation matrix using matplotlib, seaborn and corr function to check for the correlation between features. Plot the correlation matrix using a heatmap.

```
1. import matplotlib.pyplot as plt
2. import seaborn as sn
3.
4. corrMatrix = train.corr()
5. sn.heatmap(corrMatrix, annot=True)
6. plt.show()
```

Code block 22: Create a correlation matrix using matplotlib, seaborn and corr function to check for the correlation between features

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a linear regression using sklearn library. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set.

```
1. from sklearn import linear_model
2. from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
3.
4. linear_regr = linear_model.LinearRegression()
5. linear_regr.fit(X_train, y_train)
6.
7. # INSAMPLE
8. insample_prediction = linear_regr.predict(X_train)
9. rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]), squared=False)
10.
11. mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]))
12.
13. # OUTSAMPLE
14. outsample_prediction = linear_regr.predict(X_test)
15. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]), squared=False)
16.
```

```
17. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
        scaler_y.inverse_transform([outsample_prediction]))
```

Code block 23: Train and evaluate (in-sample/out-sample) a linear regression using sklearn library

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a lasso regressor using sklearn library and grid search. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named "parameters" holds all possible parameter values to be used in the grid search.

```
1.  from sklearn import linear_model
2.  from sklearn.model_selection import GridSearchCV
3.  from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
4.
5.  parameters = {
        'alpha': np.arange(0.0001, 2.00, 0.001, dtype=float),
    }
6.
7.  lasso_gs = GridSearchCV(linear_model.Lasso(), parameters, cv = 10, scoring =
        'neg_root_mean_squared_error')
8.  lasso_gs.fit(X_train, y_train)
9.
10. # in-sample
11. insample_prediction = lasso_gs.predict(X_train)
12. rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
        scaler_y.inverse_transform([insample_prediction]), squared=False)
13. mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
        scaler_y.inverse_transform([insample_prediction]))
14.
15. # out-sample
16. outsample_prediction = lasso_gs.predict(X_test)
17. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
        scaler_y.inverse_transform([outsample_prediction]), squared=False)
18. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
        scaler_y.inverse_transform([outsample_prediction]))
```

Code block 24: Train and evaluate (in-sample/out-sample) a lasso regressor using sklearn library and grid search

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) an elasticnet regressor using sklearn library and grid search. X_train and y_train refers to the features

	and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named "parameters" holds all possible parameter values to be used in the grid search.

```
1. from sklearn.linear_model import ElasticNet
2. from sklearn.model_selection import GridSearchCV
3. from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
4.
5. parameters = {
     'l1_ratio': [0.001, 0.01, 0.1, 0.5, 0.7, 1],
     'alpha': [0.001, 0.01, 0.1, 1, 10, 100]
   }
6.
7. enet_gs = GridSearchCV(linear_model. ElasticNet (), parameters, cv = 10, scoring =
     'neg_root_mean_squared_error')
8. enet_gs.fit(X_train, y_train)
9.
10. # in-sample
11. insample_prediction = enet_gs.predict(X_train)
12. rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]), squared=False)
13. mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]))
14.
15. # out-sample
16. outsample_prediction = enet_gs.predict(X_test)
17. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]), squared=False)
18. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]))
```

Code block 25: Train and evaluate (in-sample/out-sample) an elasticnet regressor using sklearn library and grid search

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a random forest regressor using sklearn library and grid search. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named "parameters" holds all possible parameter values to be used in the grid search.

```
1. from sklearn.ensemble import RandomForestRegressor
2. from sklearn.model_selection import GridSearchCV
3. from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
```

```
4.
5.  parameters = {
      'n_estimators': [5, 10, 20, 30, 40, 50, 100, 150, 200, 300, 500],
      'criterion': ["squared_error"],
      'max_depth': [5, 10, 15, 20, 30, 40, 50, 100, 150, 200, 300],
      'min_samples_split': [5, 10, 15]
    }
6.
7.  rf_gs = GridSearchCV(linear_model. ElasticNet (), parameters, cv = 10, scoring =
      'neg_root_mean_squared_error')
8.  rf_gs.fit(X_train, y_train)
9.
10. # in-sample
11. insample_prediction = rf_gs.predict(X_train)
12. rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]), squared=False)
13. mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]))
14.
15. # out-sample
16. outsample_prediction = rf_gs.predict(X_test)
17. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]), squared=False)
18. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]))
```

Code block 26: Train and evaluate (in-sample/out-sample) a random forest regressor using sklearn library and grid search

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a xgboost forest regressor using sklearn library and grid search. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named "parameters" holds all possible parameter values to be used in the grid search.

```
1.  from xgboost import XGBRegressor
2.  from sklearn.model_selection import GridSearchCV
3.  from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
4.
5.  parameters = {
      'n_estimators': [5, 10, 20, 30, 40, 50, 100, 150, 200, 300, 500],
      'max_depth': [5, 10, 15, 20, 30, 40, 50, 100, 150, 200, 300],
      'min_samples_split': [5, 10, 15]
    }
```

```
6.
7.  xgb_gs = GridSearchCV(xgboost.XGBRegressor(), parameters, cv = 10, scoring =
      'neg_root_mean_squared_error')
8.  xgb_gs.fit(X_train, y_train)
9.
10. # in-sample
11. insample_prediction = xgb_gs.predict(X_train)
12. rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]), squared=False)
13. mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
      scaler_y.inverse_transform([insample_prediction]))
14.
15. # out-sample
16. outsample_prediction = xgb_gs.predict(X_test)
17. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]), squared=False)
18. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
      scaler_y.inverse_transform([outsample_prediction]))
```

Code block 27: Train and evaluate (in-sample/out-sample) a xgboost forest regressor using sklearn library and grid search

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a support vector regressor using sklearn library and grid search. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named “parameters” holds all possible parameter values to be used in the grid search.

```
1.  from xgboost import XGBRegressor
2.  from sklearn.model_selection import GridSearchCV
3.  from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
4.
5.  parameters = {
      'kernel': ['linear', 'rbf', 'poly'],
      'C': [0.001, 0.01, 0.01, 0.05, 0.1, 0.5, 1, 3, 5, 10],
      'epsilon': [0.001, 0.01, 0.01, 0.05, 0.1, 0.5, 1, 3],
      'gamma': [0.001, 0.01, 0.01, 0.05, 0.1, 0.5, 1, 3]
    }
6.
7.  svr_gs = GridSearchCV(SVR(), parameters, cv = 10, scoring =
      'neg_root_mean_squared_error')
8.  svr_gs.fit(X_train, y_train)
9.
10. # in-sample
```

```
insample_prediction = svr_gs.predict(X_train)
rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
    scaler_y.inverse_transform([insample_prediction]), squared=False)
mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
    scaler_y.inverse_transform([insample_prediction]))

# out-sample
outsample_prediction = svr_gs.predict(X_test)
rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
    scaler_y.inverse_transform([outsample_prediction]), squared=False)
mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
    scaler_y.inverse_transform([outsample_prediction]))
```

Code block 28: Train and evaluate (in-sample/out-sample) a support vector regressor using sklearn library and grid search

Programming language	Python
Description of code block	Train and evaluate (in-sample/out-sample) a multilayer perceptron regressor using sklearn library and grid search. X_train and y_train refers to the features and the true values for the train set respectively. X_test and y_test refers to the features and the true values for the test set. The variable named “parameters” holds all possible parameter values to be used in the grid search.

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error

parameters = {
    'solver': ['adam', 'lbfgs', 'sgd'],
    'max_iter': [100, 200, 300, 500, 700],
    'alpha': [0.001, 0.01, 0.1, 0.2, 0.3, 0.5, 0.7],
    'hidden_layer_sizes': [4, 8, 16, 24, 32],
    'learning_rate': ["invscaling"]
}

mlp_gs = GridSearchCV(MLPRegressor(), parameters, cv = 10, scoring =
    'neg_root_mean_squared_error')
mlp_gs.fit(X_train, y_train)

# in-sample
insample_prediction = mlp_gs.predict(X_train)
rmse_in = mean_squared_error(scaler_y.inverse_transform([y_train]),
    scaler_y.inverse_transform([insample_prediction]), squared=False)
mape_in = mean_absolute_percentage_error(scaler_y.inverse_transform([y_train]),
    scaler_y.inverse_transform([insample_prediction]))
```

```
14.
15. #  out-sample
16. outsample_prediction = mlp_gs.predict(X_test)
17. rmse_out = mean_squared_error(scaler_y.inverse_transform([y_test]),
        scaler_y.inverse_transform([outsample_prediction]), squared=False)
18. mape_out = mean_absolute_percentage_error(scaler_y.inverse_transform([y_test]),
        scaler_y.inverse_transform([outsample_prediction]))
```

Code block 29: Train and evaluate (in-sample/out-sample) a multilayer perceptron regressor using sklearn library and grid search

14.5 Code blocks for using machine learning to identify heterogenous treatment effect in experimental studies

Programming language	Python
Description of code block	This code block shows how to create a pandas dataframe including all useful information needed to train a causal forest using the GRF R package. The snippet is written in python programming language as a preprocess step. We define our dependent variable (Y) as the difference between average weekly electricity consumed/bought for the two periods, pre and post experiment. We set the treatment assignment (W) as a binary indicator for the treated and controlled customers, 1 and 0, respectively. Finally, to form the independent variables (X) we use 4 statistical moments (average, standard deviation, skewness and kurtosis) for the pre-experiment period based on the dependent variable. The variables df_pre and df_post indicates all available measurements of each customer, 1 year before and all available data after the experiment started. The following process is repeated for all customers.

```
1.  df_customer = df_customer.append({
        'clientid': customer,
        'W': 1 if group == "treatment" else 0, # W: get intervention information
        'Y': df_post['consumption'].mean() - df_pre['consumption'].mean(),
            ignore_index = True)
    }
2.
3.  # merge W and Y with Xs (all expect 'kurtosis')
4.  featuresdf = df_pre.groupby('clientid')[ 'consumption'].agg(['mean', 'std', 'skew', 'kurtosis'])
5.  df_customer = df_customer.merge(featuresdf, on='clientid')
6.
7.  kurtosis = df_pre[['consumption', 'clientid']].groupby('clientid').apply(pd.DataFrame.kurt)
8.  df_customer = df_customer.merge(kurtosis, on='clientid')
```

Code block 30: Prepare customers' data for causal forest

Programming language	R
Description of code block	Receives as an excel file with Y (dependent variable), W (treatment assignment) and X (features or independent variables) and trains a causal forest from GRF package. The input files looks like this: group (W) | Y (dependent variable) | mean | std | skew | kurtosis

```
dat <- read_excel("Data/Data_tidy/input_file.xlsx")

# Split data ------------------------------------------------------------
X <- model.matrix(~ ., data = dat[, 3:6])  # mean, std, skew, kurtosis are used as features
Y <- dat$Y
W <- as.numeric(dat$group) # group variable has two possible variables: 1-treatment, 0-control

# Train Causal Forest Model ---------------------------------------------------------
cf <- causal_forest(
  X = X,
  Y = Y,
  W = W,
  num.trees = 8000,  #  number of trees used in the forest
  seed = 1839,  #  a seed so the results will be reproducible every time the script is executed
)
```

Useful links:

1. https://grf-labs.github.io/grf/reference/causal_forest.html

Code block 31: Train a causal forest using the GRF package

Programming language	R
Description of code block	Plot a randomly selected causal tree from a given causal forest.

```
plot(tree <- get_tree(cf, runif(n=1, min = 0, max = 7000))) #  cf variable refers to the causal forest
```

Code block 32: Plot a causal tree from a causal forest

Programming language	R
Description of code block	Evaluate model fit by assessing overlap in propensity scores. We should not be able to deterministically decide the treatment status of an individual based on its covariates, meaning none of the estimated propensity scores should be close to one or zero.

```
# Propensity score histogram ----------------------------------------------------------
hist(e.hat <- cf$W.hat)

#  Covariance balance plot-------------------------------------------------------
```

```
5. ggplot(data.frame(W.hat = cf$W.hat, W = factor(cf$W.orig))) +
     geom_histogram(aes(x = W.hat, y = stat(density), fill = W), alpha=0.3, position = "identity") +
     geom_density(aes(x = W.hat, color = W)) + xlim(0,1) +
     labs(title = "Causal forest propensity scores",
     caption = "The propensity scores are learned via GRF's regression forest")
```

Useful links:

1. https://grf-labs.github.io/grf/articles/diagnostics.html

Code block 33: Evaluate causal forest fit by assessing overlap in propensity scores

Programming language	R
Description of code block	Evaluate model fit by computing the best linear fit of the target estimand using the forest prediction (on held-out data) as well as the mean forest prediction as the sole two regressors.

```
1. test_calibration(cf)  #  cf refers to causal forest variable
```

Useful links:

1. https://grf-labs.github.io/grf/reference/test_calibration.html

Code block 34: Evaluate causal forest fit using test_calibration function

Programming language	R
Description of code block	Calculate the conditional average treatment effect for a given causal forest

```
1. # Calculate the CATE -----------------------------------------------------------
2. average_treatment_effect <- average_treatment_effect(
     cf,  # cf variable refer to the causal forest
     target.sample = c("all"),
     method = c("AIPW"),  # augmented inverse-propensity weighting (AIPW)
     subset = NULL,
     debiasing.weights = NULL,
     compliance.score = NULL,
     num.trees.for.weights = 8000
   )
3. average_treatment_effect
```

Useful links:

1. https://grf-labs.github.io/grf/reference/average_treatment_effect.html

Code block 35: Calculate the conditional average treatment effect

Programming language	R

Description of code block	Examine variable importance. A simple weighted sum of how many times feature i was split on at each depth in the forest.

```
1. cf %>%
     variable_importance() %>%
     as.data.frame() %>%
     mutate(variable = colnames(cf$X.orig)) %>%
     arrange(desc(V1))
```

Code block 36: Examine variable importance

Programming language	R
Description of code block	Predict individual treatment effects given a causal forest. Based on the data used at this stage, you can calculate either the ITE for the treatment or the control group. The prediction also provide the estimates variances.

```
1. preds <- predict(
     object = cf,
     newdata = model.matrix(~ ., data = dat[, 4:7]),  # variable data refer to all data
     estimate.variance = TRUE,
     type = "cate"
   )
```

Useful links:

1. https://rdrr.io/r/stats/predict.html

Code block 37: Predict individual treatment effects

Programming language	R
Description of code block	Check for heterogeneity in predictions by using rank_average_treatment_effect function.

```
1. rate <- rank_average_treatment_effect(
     cf,
     dat$preds,
     target = c("QINI"),
     q = seq(0.1, 1, by = 0.1),
     R = 200,
     subset = NULL,
     debiasing.weights = NULL,
     compliance.score = NULL,
     num.trees.for.weights = 8000
   )
2. rate
3. plot(rate)
```

Useful links:

1. https://grf-labs.github.io/grf/reference/rank_average_treatment_effect.html

Code block 38: Check for heterogeneity

Programming language	R
Description of code block	Plot the relationships between a variable and the predicted treatment effects. You can select the variable by replacing the value in x variable.

```
p1 <- ggplot(dat, aes(x = mean, y = preds)) + geom_point() + geom_smooth(method = "loess",
  span = 1) + theme_light()

p2 <- ggplot(dat, aes(x = std, y = preds)) + geom_point() + geom_smooth(method = "loess",
  span = 1) + theme_light()

cowplot::plot_grid(p1, p2)
```

Code block 39: Plot the relationships between a variable and the predicted treatment effects

Programming language	R
Description of code block	Plot the predicted treatment effects by their rank. The following function will plot also the 95% confidence interval for the predicted treatment effects.

```
plot_htes <- function(cf_preds, ci = FALSE, z = 1.96) {
 if (is.null(cf_preds$predictions)|| nrow(cf_preds$predictions) == 0)
  stop("cf_preds must include a matrix called 'predictions'")

 out <- ggplot(
  mapping = aes(
   x = rank(cf_preds$predictions),
   y = cf_preds$predictions
  )
 ) +
 geom_point() +
  labs(x = "Rank", y = "Estimated Treatment Effect") + theme_light()

 if (ci && nrow(cf_preds$variance.estimates) > 0) {
  out <- out +
   geom_errorbar(
    mapping = aes(
     ymin = cf_preds$predictions + z * sqrt(cf_preds$variance.estimates),
     ymax = cf_preds$predictions - z * sqrt(cf_preds$variance.estimates)
    )
```

```
    )
  }
  return(out)
}

plot_htes(preds, ci = TRUE)
```

Code block 40: Plot the predicted treatment effects by their rank

Programming language	R
Description of code block	Plot the distribution of predicted treatment effects.

```
d <- density(preds$predictions)
plot(d, main="Distribution of Predicted Treatment Effects")
polygon(d, col="blue", border="black")
abline(v=0, col="red")
```

Code block 41: Plot the distribution of predicted treatment effects

14.6 Code blocks for the estimation of consumers' willingness to pay for the repair of home appliances and for more efficient energy home appliances

Programming language	R
Description of code block	This R code performs various data preprocessing and transformation tasks on a dataset loaded from a CSV file such as importing, cleaning and pre-processing.

```
# Load required packages
library(readr)

# Import data from a CSV file
data <- read_csv("your_data.csv")
# Remove rows with missing values
data <- na.omit(data)

# Fill missing values with a specific value (e.g., 0)
data$column_name[is.na(data$column_name)] <- 0
# Remove duplicate rows
data <- unique(data)
# Calculate z-scores for a specific column
z_scores <- scale(data$column_name)

# Define a threshold for outlier detection
threshold <- 2  # Adjust as needed
```

```
18.
19. # Remove rows with z-scores above the threshold
20. data <- data[abs(z_scores) <= threshold, ]
21. # Standardize numeric columns
22. data[, numeric_columns] <- scale(data[, numeric_columns])
23. # Normalize numeric columns to [0, 1]
24. data[, numeric_columns] <- scale(data[, numeric_columns], center
25.     = FALSE)
26. # Convert a categorical column to dummy variables
27. data <- model.matrix(~ column_name - 1, data = data)
28. # Scale numeric columns to a specific range (e.g., [0, 1])
29. data[, numeric_columns] <- scale(data[, numeric_columns])
```

Code block 42: Data preprocessing and transformation tasks on a dataset loaded from a CSV

Programming language	R
Description of code block	This code performs a series of actions such as creating visualizations, performing statistical analysis and data modelling R. More specifically, the code starts by loading the necessary libraries for data manipulation, visualization, and machine learning. Then the code imports the dataset, presents summary statistics, and visualizes data distributions using box plots, histograms, and scatterplot matrices. It explores the correlation structure within the dataset through a correlation matrix plot. It preprocesses the data by splitting it into training and testing sets, scaling numeric features, and combining them with categorical features. Then, it trains a random forest regression model on the training data, predicts on the test set, and evaluates model performance using the Root Mean Squared Error (RMSE).

```
1.  # Load libraries
2.  library(ggplot2)
3.  library(dplyr)
4.  library(corrplot)
5.  library(caTools)
6.  library(caret)
7.  library(randomForest)
8.  # Load the dataset (replace 'energy_efficiency.csv' with your dataset file)
9.  data <- read.csv("energy_efficiency.csv")
10. # Summary of the dataset
11. summary(data)
12.
13. # Box plot for the target variable
14. ggplot(data, aes(x = X6, y = Y1)) + geom_boxplot()
15.
16. # Correlation matrix visualization
```

```r
cor_matrix <- cor(data)
corrplot(cor_matrix, method = "color")

# Scatterplot matrix
pairs(data[, c("X1", "X2", "X3", "X4", "X5", "X6", "Y1")])

# Histogram of a numeric variable
ggplot(data, aes(x = X1)) + geom_histogram(binwidth = 0.1, fill = "blue", color = "black")

# Bar plot of a categorical variable
ggplot(data, aes(x = X2)) + geom_bar(fill = "green")
# Split the data into training and testing sets
set.seed(123)
split <- sample.split(data$Y1, SplitRatio = 0.7)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)

# Scale the numeric features
numeric_features <- train_data[, c("X1", "X2", "X3", "X4", "X5", "X6")]
scaled_features <- scale(numeric_features)

# Combine scaled numeric features with categorical features
X_train <- cbind(scaled_features, train_data[, c("X7", "X8")])
X_test <- cbind(scale(test_data[, c("X1", "X2", "X3", "X4", "X5", "X6")]), test_data[, c("X7", "X8")])

# Define the target variable
y_train <- train_data$Y1
y_test <- test_data$Y1
# Train a random forest regression model
rf_model <- randomForest(y_train ~ ., data = data.frame(X_train, Y1 = y_train), ntree = 100)

# Predict on the test set
rf_pred <- predict(rf_model, data.frame(X_test))

# Evaluate the model
rmse <- sqrt(mean((rf_pred - y_test)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

Code block 43: Data exploration and predictive modeling script for a dataset

Programming language	R
Description of code block	This R code is used to conduct statistical inference on the collected data, report results and export the results in R. It begins by loading necessary libraries, then generates a sample dataset with a specified structure and

	random values. Next, it performs an independent t-test to compare the means of the two groups. Additionally, the code creates a boxplot visualization of the data using ggplot2.

```
# Load necessary libraries
library(dplyr)
library(ggplot2)

# Create a sample dataset (replace this with your own data)
set.seed(123)
data <- data.frame(
Group = rep(c("Group A", "Group B"), each = 50),
Value = c(rnorm(50, mean = 5, sd = 1), rnorm(50, mean = 6, sd = 1)))
# Perform a t-test to compare means of two groups
t_test_result <- t.test(Value ~ Group, data = data)
# Print the results
cat("T-Test Results:\n")
print(t_test_result)
# Export the results to a CSV file
write.csv(t_test_result, file = "t_test_results.csv")

# Create a plot to visualize the data
ggplot(data, aes(x = Group, y = Value)) +
geom_boxplot() + labs(title = "Boxplot of Value by Group", x = "Group", y = "Value") +
theme_minimal()

# Save the plot as an image file
ggsave("boxplot.png")
```

Code block 44: Conduct statistical inference in R on a sample dataset

14.7 Code blocks for assessing consumers' average price bias

Programming language	STATA
Description of code block	This code is used to the calculate financial knowledge score. The code is very similar for calculating the rest indicators regarding financial and environmental knowledge, behavior and attitude. Details on the questions selected in each category can be found in D4.3. In summary, the code processes a financial knowledge survey dataset and creates indicator variables to identify specific aspects of financial knowledge. It then calculates a total financial knowledge score based on these indicators and determines if the knowledge score meets a minimum threshold.

```
**Case 1.Financial Knowledge score
generate borrow_100case1=0
```

```
3.  replace borrow_100case1=1 if finance_borrow_100_c==2
4.  label variable borrow_100case1 "Identification of interest"
5.
6.  generate inflation_qualitativecase1=0
7.  replace inflation_qualitativecase1=1 if finance_inflation_qualitative_c==3
8.  label variable inflation_qualitativecase1 "Impact of inflation in spending power"
9.
10. generate inflation_quantitativecase1 =0
11. replace inflation_quantitativecase1=1 if finance_inflation_quantitative_c ==1
12. label variable inflation_quantitativecase1 "Identification of compound interest_1"
13.
14. generate interest_ratecase1=0
15. replace interest_ratecase1=1 if finance_interest_rate_c ==1
16. label variable interest_ratecase1 "Identification of compound interest_2"
17.
18. generate investmentscase1=0
19. replace investmentscase1=1 if finance_investments_c ==2
20. label variable investmentscase1 "Risk diversification"
21.
22. generate finance_blockchaincase1=0
23. replace finance_blockchaincase1=1 if finance_blockchain_c==1 |  finance_blockchain_c==2
24. label variable finance_blockchaincase1 "Product awareness"
25.
26. **generate a variable for the total score in financial knowledge
27. egen financial_knowledge_score_c1=rowtotal( borrow_100case1 inflation_qualitativecase1
28. inflation_quantitativecase1 interest_ratecase1 investmentscase1 finance_blockchaincase1)
29. label variable financial_knowledge_score_c1 "Financial Knowledge score"
30.
31. **6points maximum
32. **minimum knowledge score
33. generate minimun_knowledge_score=0
34. replace minimun_knowledge_score=1 if financial_knowledge_score_c1>=4
35. label variable minimun_knowledge_score "Knowledge score over 4"
```

Code block 45: Calculate financial knowledge score

Description of code block	This code is used to the calculate financial literacy score computed from the total of the Knowledge, Behaviour score and the average score across the attitude questions. The code is very similar for calculating the environmental literacy score. Details on how these indicators are calculated can be found in D4.3. The code calculates a financial literacy score by summing three component scores (financial knowledge score, financial behaviour score and financial attitude score) and defines an indicator to check if the literacy score meets a minimum threshold.

```
**Case1.Financial Literacy score
egen financial_literacy_score_c1=rowtotal(financial_knowledge_score_c1
financial_behaviour_score_c1 financial_attitude_score_c1 )
label variable financial_literacy_score "Financial Literacy score"

*16 points maximum
**minimum 60-70%proficiency 10??
generate minimun_literacy_score=0
replace minimun_literacy_score=1 if financial_literacy_score>=10
label variable minimun_literacy_score "Knowledge score over 3"
```

Code block 46: Calculate financial literacy score

Description of code block	The provided code performs a multivariate regression (specifically, a multivariate linear regression) with multiple dependent variables: envir_literacy_score_es1 (environmental literacy score), envir_knowledge_score_es1 (environmental knowledge score), envir_behaviour_score_es1 (environmental behaviour score), and env_attitude_score_es1 (environmental attitude score). The independent variables include various financial and demographic factors such as the financial literacy score, the number of people that lives in the house (both adults and children), household's yearly income, the employability status (Student, Full-time employment, Part-time employment, etc.) and the home status (Tenant, Homeowner and Landlord).

```
**multivariate regression

mvreg envir_literacy_score_es1 envir_knowledge_score_es1 envir_behaviour_score_es1 ///
   env_attitude_score_es1 = financial_literacy_score_c1  demo_age ///
   demo_people_in_home_adults demo_people_in_home_children ///
   i.demo_income_c i.demo_employment_status_c i.demo_home_status_c
```

Code block 47: Multivariate regression model to assess how environmental literacy, knowledge, and behavior scores are influenced by various variables, including demographics and financial literacy.