Scaling Causal Inference to Big Data with ZaliQL

Babak Salimi, Corey Cole, Dan Suciu, Dan R. K. Ports,
University of Washington
{bsalimi,coreylc,suciu,drkp}@cs.washington.edu

ABSTRACT

Causal inference from observational data is a subject of active research and development in statistics and computer science. Many toolkits have been developed for this purpose that depend on statistical software. However, these toolkits do not scale to large datasets. We propose and demonstrate ZaliQL: a SQL-based framework for drawing causal inference from observational data. ZaliQL supports the state-of-the-art methods for causal inference and runs at scale within a database engine. ZaliQL is designed as an extension for a PostgreSQL object-relational database system. In addition, we designed a visual interface to wrap around ZaliQL to demonstrate the causal effect of different weather conditions on flight delays.

1 INTRODUCTION

Most data driven decisions today are based on purely predictive analytics, which only finds associational dependencies and correlations. The danger is that naive data enthusiasts will incorrectly interpret these predictive models as causal. While the distinction between causal and predictive analysis has been recognized, the conflation between the two is common which results in false discovery claims.

Causal inference has been studied extensively in statistics and computer science [2, 4, 9, 12], and many toolkits have been developed for performing causal inference in statistical software such as R, e.g., MatchIt and CEM [3, 6]. However, these toolkits do not scale to large datasets. For example, performing causal inference on a dataset with 10M entries takes several hours using MatchIt or CEM. Additionally, causal analysis is part of a larger pipeline that includes data acquisition, data cleaning, and data integration; for large datasets, these tasks are best handled by a relational database engine, which provides most of the functionality needed for these tasks, and also scales up to large datasets.

We argue that *causal inference is a data management problem.* In this demonstration, we propose ZaliQL,[1] a SQL-based framework for drawing causal inference within DBMS. ZaliQL takes the first step towards truly scalable causal inference by modeling it as a data management problem. We demonstrate that approaching causal inference from this fresh perspective is key to scalability and robustness. ZaliQL supports state-of-the-art methods for causal inference and runs at scale within a database engine. The framework includes a series of optimization techniques that allow it to scale to billions of records.

2 SYSTEM ARCHITECTURE

The overall architecture of ZaliQL is shown in Figure 1. The API is a set of stored procedures that support a wide range of methods for performing causal inference on data stored in the PostgreSQL DBMS. The API will be packaged as a PostgreSQL extension. The functionality of the API is modeled after the MatchIt and CEM R libraries [3, 6] and includes methods for drawing causal inference from relational data. ZaliQL also includes a web GUI (see Figure 3) that can be hosted either locally or remotely .

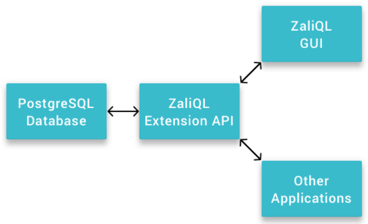


Figure 1: ZaliQL architecture

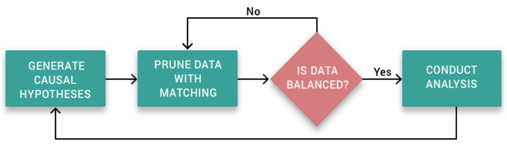


Figure 2: Causal analysis workflow

3 DEMONSTRATION DETAILS

Modern causal analysis is an iterative process, as we illustrate in Fig 2. One generates a hypothesis, pre-processes relevant data with matching (explained below), then conducts the causal analysis; most often this needs to be repeated with a new matching method, a new hypothesis or new datasets. We demonstrate ZaliQL by showing several real causal analysis performed on two, large datasets that we downloaded from the Web and integrated in Postgres.

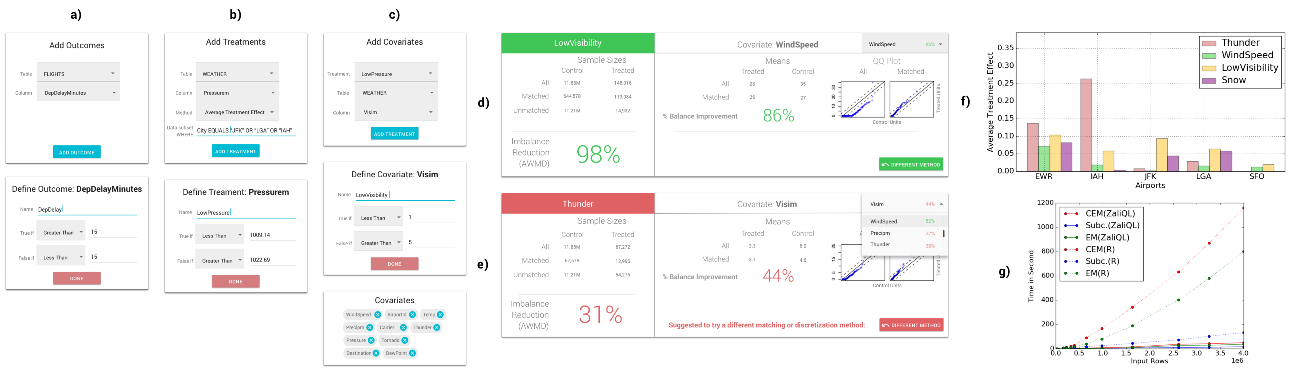
Data: The analysis will be conducted on a spatio-temporal join of the following datasets: (a) *Flight dataset (105M entries)*: collected by the US Department of Transportation.[2] It contains records of more than 90% of US domestic flights of major airlines from 1988 to the present. It includes attributes such as FlightDate, OriginAirportID, CarrierID, CRSDepTime (scheduled departure time), and DepDelayMinutes (departure delay). (b) *Weather dataset (40M entries):* historical weather data for relevant flight gathered using the Weather Underground API.[3] It includes Code (Airport ID), Date, Time, Visim (visibility in km), Tempm (Temperature in C°) Wspdm (wind speed kph), Pressurem (Pressure in mBar), Precipm (Precipitation in mm), Snow (binary), Thunder (binary).

Causal questions: Our demonstration starts by creating a set of *causal questions* regarding the effect of different weather attributes on flight delays. Q1: Does low air pressure causes flight departure delays? Q2: Which weather features have the main causal effect on departure delays? Q3: Do the findings to the previous question

[1] The prefix Zali refers to al-Ghzali (1058-1111), a medieval Persian philosopher. It is known that David Hume (1711-1776), a Scottish philosopher, who gave the first explicit definition of causation in terms of counterfactuals, was heavily influenced by al-Ghzali's conception of causality [13].

[2] http://www.transtats.bts.gov/

[3] https://www.wunderground.com



a) Add Outcomes — Table: FLIGHTS; Column: DepDelayMinutes; ADD OUTCOME

Define Outcome: **DepDelayMinutes** — Name: DepDelay; True if: Greater Than 15; False if: Less Than 15; DONE

b) Add Treatments — Table: WEATHER; Column: Pressurem; Method: Average Treatment Effect; Data subset WHERE: City EQUALS "JFK" OR "LGA" OR "IAH"; ADD TREATMENT

Define Treament: **Pressurem** — Name: LowPressure; True if: Less Than 1009.14; False if: Greater Than 1022.69; DONE

c) Add Covariates — Treatment: LowPressure; Table: WEATHER; Column: Visim; ADD TREATMENT

Define Covariate: **Visim** — Name: LowVisibility; True if: Less Than 1; False if: Greater Than 5; DONE

Covariates: WindSpeed, AirportId, Temp, Precipm, Carrier, Thunder, Pressure, Tornado, Destination, DewPoint

d) LowVisibility

Sample Sizes	Control	Treated
All	11.85M	148,016
Matched	644,578	113,084
Unmatched	11.21M	14,932

Imbalance Reduction (AWMD) 98%

Covariate: **WindSpeed** (WindSpeed 86%)

Means	Treated	Control
All	28	35
Matched	26	27

% Balance Improvement 86%

DIFFERENT METHOD

e) Thunder

Sample Sizes	Control	Treated
All	11.85M	67,272
Matched	67,579	12,996
Unmatched	11.21M	54,276

Imbalance Reduction (AWMD) 31%

Covariate: **Visim** (Visim 44%; WindSpeed 52%; Precipm 32%; Thunder 38%)

Means	Treated	Control
All	3.3	6.0
Matched	3.1	4.6

% Balance Improvement 44%

Suggested to try a different matching or discretization method: DIFFERENT METHOD

Figure 3: Demonstration screenshot described in section 3

differ between major aiports? We answer these questions using ZaliQL by

(1) specifying DepDelay as our outcome of interest (potential effect) as shown in Figure 3(a)
(2) specifying a set of binary treatments (potential causes) that might affect DepDelay, as shown in Figure 3(b), In particular the following binary treatments will be created: LowVisibility (1 if Visim$<$ 1; 0 if Visim$>$ 5); HeavySnow (1 iff Precipm$>$ 0.3 and Snow= 1); WindSpeed (1 if Wspdm$>$ 40; 0 if Wspdm$<$ 20); Thunder; LowPressure (1 if Pressurem$<$ 1009.14; 0 if Pressurem$>$ 1022.69)
(3) specifying a subset of data that is relevant to the analysis. We select five US airports with high rate of weather-related delay, namely, San Francisco (SFO) , John F. Kennedy (JFK), Newark Liberty (EWR), George Bush (IAH), and LaGuardia Airport (LGA).

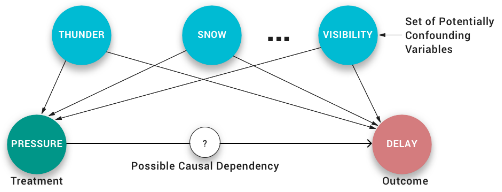
Computing ATE: In causal inference, the objective is to quantify the causal effect of a binary attribute called treatment, on an attribute called outcome. A common measure to to compute causal effect is *average treatment effect (ATE)*, which is defined as the difference in expected values of the treated and untreated units.[4] For example for Q1, ZaliQL computes ATE as

$$E[\text{Depdelay}|\text{LowPressure} = 1] - E[\text{Depdelay}|\text{LowPressure} = 0]$$

where, E[Depdelay—LowPressure=x], $x = (0, 1)$ is computed by taking the emetical average of Depdelay where LowPressure is x. It is a common measure to compare an outcome (DepDelay) in the *treated group*, those subjects (flights) that receive a treatment (LowPressure=1), with the *control group*, the opposites. We show that for LowPressure, ATE is 3.7min, which is a relatively large that suggests pressure it affects DepDelay. However, it is known that LowPressure does not have any causal impact on departure delay (it only requires longer takeoff distance) [14]. This observation raises the question that where does this difference coming from?

Confounding variables: The main issue with causal inference from observational data is the exitance of *confounding variables* that make it difficult to establish a clear causal link between treatment and outcome. We show that the observed positive effect of LowPressure was the result of the confounding influence of other

[4]In causality literature subjects of a study, e.g., patients in a clinical trial, are called units. In our case each flight is considered a unit.



THUNDER
SNOW
...
VISIBILITY
Set of Potentially Confounding Variables
PRESSURE
?
DELAY
Possible Causal Dependency
Treatment
Outcome

Figure 4: Confounding influence

variables such as visibility, snow and Thunder Specifically, we show that, LowPressure is highly associated with unsettled weather conditions such as LowVisbility (Figure 4). For instance the average of LowVisbility in the treated group is much less than the opposite group. Thus, it is unclear that the observed DepDelay difference between the groups is causes by LowPressure or LowVisbility.

Adjusting for confounding variables: Causal inference provides a framework for adjusting for the effects of confounding variables. With only one categorical confounding variable, adjustment can be done simply by partitioning data into groups with similar values of the confounding variable, then computing the ATE as weighted average of average treatment effects in each groups. However, real-world causal analysis requires a dozen or more continues confounding variables, and in this case there is not enough data in each group to enable a meaningful estimation of the ATE. For example, as exemplified in Figure 4, LowPressure has more than ten confounding variables. Most of the complexity in causal analysis tools, as well as in ZaliQL, consists of implementing sophisticated methods for enlarging the groups while still performing a meaningful inference of the ATE. ZaliQL controls for confounding influences (a task usually referred to as *covariate adjustment*) using *subclassification* and *propensity score matching* [5, 10, 11]. These are popular methods for covariate adjustment developed in social science and statistics. We refer to these as matching methods. The key goal of matching is to prune data so that the remaining data (*matched data*) have better *balance* between the treated and control groups. That is, the empirical distributions of the confounding variables in the treated and control groups are similar after matching. Once the two groups are balanced, any observed difference of the outcome between the two groups can be attributed to the treatment. In our demo,

Unit	Covariates X	Treatment assignment T	Treatment outcome $Y(1)$	Control outcome $Y(0)$	Causal Effect $Y(1) - Y(0)$
1	X_1	T_1	$Y_1(1)$	$Y_1(0)$	$Y_1(1) - Y_1(0)$
2	X_2	T_2	$Y_2(1)$	$Y_2(0)$	$Y_2(1) - Y_2(0)$
...	...	...	...	...	...
N	X_N	T_N	$Y_N(1)$	$Y_N(0)$	$Y_N(1) - Y_N(0)$

Figure 5: The Potential Outcome Framework [12]

(1) for each treatment, we select a set of covariates deemed to confound the treatment and DepDelay (Figure 3(d)).
(2) we select a matching method and adjust its tuning parameters (Figure 3(e)).

Checking Balance: In this step, we check whether the matching process successfully improved the covariates' balance. In particular, (1) we compare the distribution for each covariate between the treated and control group on matched data (this will be done for all treatments). For this task, ZaliQL provides numerical summaries such as the mean difference and visual summaries such as quantile quantile plots 3(d,e). (2) we can perform different matching methods using different parameters, showing that different methods trade off balance and size of matched data

The main objective of this step is to demonstrate the typical workflow of causal inference from observational data. As shown in Figure 2, it consists of the iterative process of matching and balance checking. This highlights the demand for tools such as ZaliQL which perform matching at scale.

Answer to the causal questions: In this step, we answer the causal questions created at the very first step of the demonstration using ZaliQL. For Q1, we show that LowPressure has no significant causal effect on DepDelay. For Q2, we identify that other treatments have significant causal effect on DepDelay (Figure 3(F)). For Q3, we report the major causes of flight delay at the airports under study are actually different. We show that the obtained results are in accordance with FAA reports. For example, the runway configuration at SFO makes the airport unusually susceptible to delays in low-visibility conditions [1].

Scalability: Finally, we demonstrate system scalability by letting users interactively run queries on this large dataset–which would take R hours. As seen in Figure 3(g), ZaliQL is able to run through 4M entries in less than a minute, whereas R takes over 20 minutes.

4 INTERNALS

4.1 Basic Formalism

The basic causal model in statistics is called the *potential outcome framework (POF)* [4, 12]. In this model, we are given a table with N rows called *units* indexed by $i = 1 \dots N$ (see Table 5). The binary attribute T denotes *treatment assignment*. X is a vector of background characteristics, (e.g., sex, race, age, ...) of each unit, called *covariates*, unaffected by treatment; and the two attributes $Y(0), Y(1)$ represent *potential outcomes*: $Y(1)$ is the outcome of the unit if it is exposed to the treatment and $Y(0)$ is the outcome when it is exposed to the control.

The *treatment effect* caused by the treatment T_i for the ith unit is defined as $Y_i(1) - Y_i(0)$. The goal of causal analysis is to compute the *average treatment effect (ATE)*: ATE = E[Y(1)-Y(0)] = E[Y(1)] - E[Y(0)]. The so-called *fundamental problem of causal inference (FPCI)* is that for each unit we only know either $Y(1)$ or $Y(0)$ but not both. . For example, for the treatment of LowPressure either Depdelay(1) or Depdelay(0) are available. Thus, further assumption is needed for estimating ATE.

The strongest is the *independence assumption*, which states that the treatment mechanism is independent of the potential outcomes, i.e., $(Y(1), Y(0)) \perp\!\!\!\perp T$. Then, it holds that $\mathbb{E}[Y(1)] = \mathbb{E}[Y(1)|T = 1]$ and similarly $\mathbb{E}[Y(0)] = \mathbb{E}[Y(0)|T = 0]$ and we have: ATE = E[Y(1)—T=1]-E[Y(0)—T=0]. . ZaliQL is designed for drawing causal inference from *observational data*, where the mechanism used to assign treatments to units is not known, and where independence fails in general.

In observational data, the mechanism used to assign treatments to units is not known, thus independence fails in general. For example, thunder occurs mostly in the summer, which is also high travel season, and therefore delays may be caused by the high traffic. In that case T and $Y(0)$ (the delay when thunder does not occur) are correlated, since they are high in the summer and low in the winter, and similarly for T and $Y(1)$. Here, statistical literature makes the following weaker assumption [10] called *Strong Ignorability*: Forall x the following hold: (1) Unconfoundedness $(Y(0), Y(1) \perp\!\!\!\perp T|X = x)$ and (2) Overlap $0 < \Pr(T = 1|X = x) < 1$.

If strong ignorability holds, one can estimate ATE by taking the average difference for each group with $X = x$ i.e., $E_x[E[Y(1) - Y(0)|X = x]]$. In practice, however, a direct application of this method is impossible, because the data is typically very sparse: for any value $X = x$ we either have no data values at all, or very few such values, which means that estimating $E[Y(T)|T = 1 \text{ or } 0, X = x]$ as an average from the database leads to large sampling error. A solution adopted in statistics is *matching* and *subclassification* [10]. The idea is to match each treated unit with one or multiple control units with "close" values of the covariate attributes X, where closeness is defined using some distance function between the covariate values of two units. The most commonly used distance functions are presented in Table 6.

Name	$\delta(x_i, x_j) =$	**Comments**
Coarsened distance	0 if $C(x_i) = C(x_j)$ ∞ if $C(x_i) \neq C(x_j)$	Where $C(x)$ is a function that coarsen a vector of continues covariate [5]
Propensity score distance (PS)	$\lvert E(x_i) - E(x_j)\rvert$	where $E(x) = \Pr(T = 1\lvert X = x)$ is the propensity score [10]
Mahalanobis Distance (MD)	$(x_i - x_j)'\Sigma^{-1}(x_i - x_j)$	where Σ = covariance matrix

Figure 6: Distance Measures used in Matching

4.2 Methods

Define the table $R(T, X_1, X_2, \dots, Y)$, the goal is to compute ATE conditioned on a set of covariates X. ATE within each group $X = \bar{x}$ cane be computed with the simple SQL statement:

```
SELECT FROM
  (SELECT avg(Y) FROM R WHERE X = x̄ AND T=1) -
  (SELECT avg(Y) FROM R  WHERE X = x̄ AND T=0)
```

However, this is impossible, as explained above. Instead, we apply two techniques used in statistics, then optimize the resulting queries before sending them to PostgresSQL The following methods are supported by ZaliQL.

Nearest neighbor matching The most common method is that of $k : 1$ nearest neighbor matching (NNM). NNM controls matches for each treated unit and can be done with or without replacement. We denote them respectively by NNMWR and NNMNR. In practice,

```
CREATE VIEW Nnmnr
AS WITH potential_matches AS
  (SELECT treated.ID AS tID, control.ID AS cID,
          $\delta(treated.X, control.X)$  AS distance
   FROM $R$ AS control, $R$ AS treated
   WHERE control.T=0 AND treated.T=1
     AND $\delta(treated.X, control.X)$ < $caliper$)),
          ordered_potential_matches AS
  (SELECT *, ROW_NUMBER() over (ORDER BY distance) AS order
   FROM potential_matches)
SELECT *
FROM ordered_potential_matches AS rp
WHERE NOT EXISTS
    (SELECT *
     FROM ordered_potential_matches AS z
     WHERE z.order < rp.order AND z.cID=rp.cID)
  AND (SELECT count(*)
     FROM ordered_potential_matches AS rp
     WHERE z.order < rp.order AND z.tID=rp.tID) $\leq$ $k$;
```

Figure 7: NNMNR

```
CREATE VIEW SubC AS
(WITH tmp0 AS
  (SELECT *. ntile($n$) over w subclass,
   FROM $R$ window w AS (ORDER BY ps))
SELECT ID, T, $\mathcal{X}$, Y, subclass,
             max(T) over w maxT, min(T) over w minT
FROM tmp0  window w AS (PARTITION BY BY subclass)
WHERE maxT!=minT)
```

Figure 8: Subclassification based on propensity score

```
CREATE VIEW Cem AS
WITH subclasses AS
  (SELECT *,
          max(ID) OVER w subclass, max(T) OVER w AS minT,
          min(T) OVER w AS maxT
   FROM $R^c$
   Group by $\mathcal{X}$
   Having minT!=maxT)
SELECT ID, T, $\mathcal{X}$, Y, subclass
FROM subclasses,$R^c$
WHERE subclasses.$\mathcal{X}$=$R^c$.$\mathcal{X}$
```

Figure 9: CEM

matching is usually performed without replacement. Notice that NNM faces the risk of bad matches if the closest neighbor is far away. This issue can be resolved by imposing a tolerance level on the maximum distance, known as the *caliper* (see e.g., [7]).

The basic SQL statement to perform NNMNR is depicted in Figure 7. In this solution, nearest control units are identified by means of an anti-join. Specifically, all potential matches and their distances are identified by joining the treated with the control units that are closer than the caliper. Then, this set is sorted into ascending order of distances. In addition, the order of each row in the sorted set is identified using the window function `ROW_NUMBER`. Finally, all units with the order of less than or equal to k are selected as the matched units. However, ZaliQL does NNM efficiently by leveraging recent developments in the context of spatial-databases (see e.g., [8]).

Subclassification It is easy to see that NNM does not necessarily use all the data, meaning that despite being in the range of a treatment, many control units are discarded. In subclassification, the aim is to form subclasses for which the distribution of covariates for the treated and control groups are as similar as possible.

Subclassification based on the propensity score in this approach data is is typically partitioned into five subclasses based on the n quintiles of propensity score. Figure 8, shows the SQL implemention of this method using window function `ntile`.

Coarsening Exact Matching (CEM) This is a particular form of subclassification in which the vector of covariates X is coarsened into according to a set of user-defined cutpoints or any automatic discretization algorithm. All units with similar coarsened covariates values are placed in unique subclasses. All subclasses with at least one treated and one control unit are retained and the rest of units are discarded. Let $\mathcal{X}$ be the coarsened version of X and R^c be a discretized version of R. As depicted in Figure 9, CEM is essentially a GROUP-BY-HAVING query, which is relatively expensive. To perform CEM efficiently, ZaliQL leverages techniques from data management including pushing aggregate down to the base relation, data cube-aggregation, pre-matching wrt. multiple treatments and bitmap indices. It also support several methods discretization methods for coarsening.ZaliQL include several discretization methods for coarsening.

5 CONCLUSIONS

In this demonstration we introduce ZaliQL, which is aimed at performing causal inference on large relational data within a DBMS. ZaliQL makes a first step towards truly scalable causal inference by modeling it as a data management problem. It leverages existing techniques in data management to efficiently evaluate several causal hypotheses from relational data. ZaliQL is optimized for PostgreSQL, and also includes a visual interface to wrap around the framework.

REFERENCES

[1] FAA. 2013. Simultaneous Close Parallel Approach Procedure. https://www.faa.gov/documentLibrary/media/Notice/N8260_73.pdf. (2013).

[2] Ronald Aylmer Fisher. 1935. *The design of experiments.* Oliver and Boyd, Oxford, England.

[3] Daniel E Ho, Kosuke Imai, Gary King, and Elizabeth A Stuart. 2007. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis* 15, 3 (2007), 199–236.

[4] Paul W Holland. 1986. Statistics and causal inference. *Journal of the American statistical Association* 81, 396 (1986), 945–960.

[5] Stefano M Iacus, Gary King, and Giuseppe Porro. 2011. Causal inference without balance checking: Coarsened exact matching. *Political analysis* (2011), mpr013.

[6] Stefano M Iacus, Gary King, Giuseppe Porro, and others. 2009. CEM: software for coarsened exact matching. *Journal of Statistical Software* 30, 9 (2009), 1–27.

[7] Mark Lunt. 2014. Selecting an appropriate caliper can be essential for achieving good balance with propensity score matching. *American journal of epidemiology* 179, 2 (2014), 226–235.

[8] Regina O Obe and Leo S Hsu. 2015. *PostGIS in action.* Manning Publications Co.

[9] Judea Pearl. 2000. *Causality: models, reasoning, and inference.* Cambridge University Press.

[10] Paul R. Rosenbaum and Donald B. Rubin. 1983. The Central Role of the Propensity Score in Observational Studies for Causal Effects. *Biometrika* 70, 1 (1983), pp. 41–55.

[11] Paul R Rosenbaum and Donald B Rubin. 1984. Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American statistical Association* 79, 387 (1984), 516–524.

[12] Donald B Rubin. 2005. Causal Inference Using Potential Outcomes. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331. DOI:http://dx.doi.org/10.1198/016214504000001880

[13] Cosma Shalizi. 2012. *Advanced data analysis from an elementary point of view.* Cambridge University Press.

[14] United States Department of Transportation, Federal Aviation Administration. 2008. FAA. Pilots Handbook of Aeronautical Knowledge. (2008).