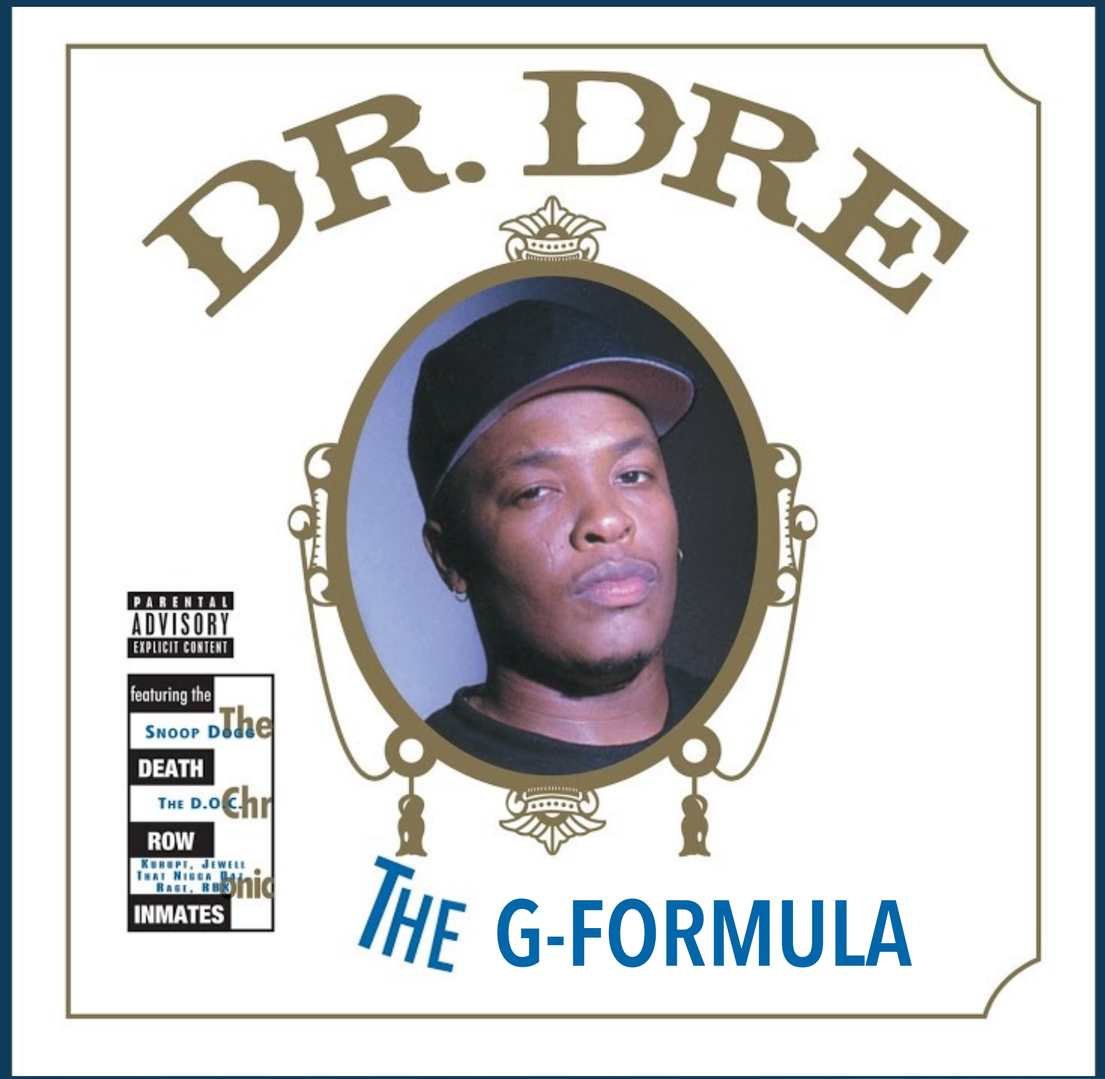
G-thangs.



March, 2024

travis@cstructure.io

Hi! + disclosures, fine print

- I'm Travis Gerke, ScD

- Past:
  - Training in biostatistics/epidemiology (Harvard)
  - Faculty at University of Florida / Moffitt Cancer Center
  - Director of Health Informatics, Moffitt Cancer Center (post-ac)

- Current:
  - Director of Data Science, The Prostate Cancer Clinical Trials Consortium (PCCTC / MSK)
  - Co-Founder / Chief Data Scientist, cStructure
  - Chief Data Officer, Toby Health

- Real Life:
  - Background in construction / HVAC
  - Lived in RV 2021-2023, currently San Diego

Suppose we run **Disney World**

8am, **Seven Dwarfs ride**

9am, **Seven Dwarfs line**

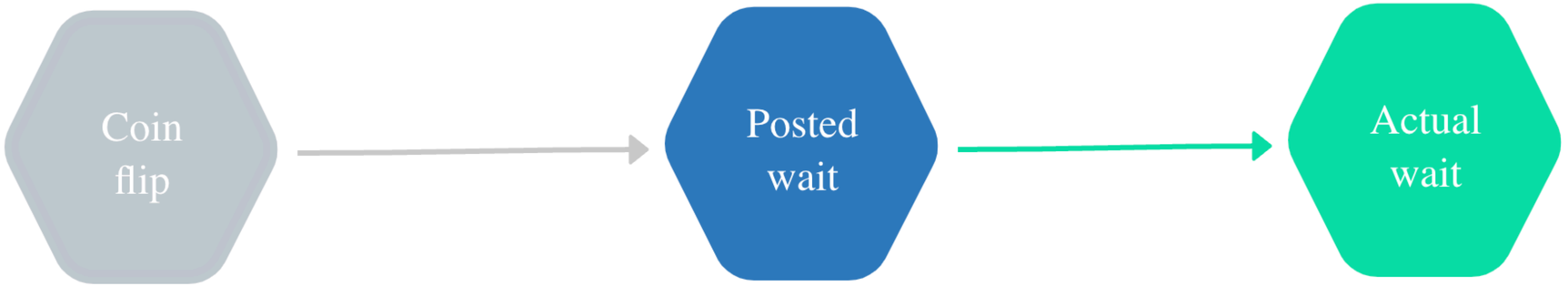
8am, **Seven Dwarfs ride**

9am, **Seven Dwarfs line**

Can we strategically **set posted wait times** at 8am such that **actual wait times** at 9am optimize attendee experience?

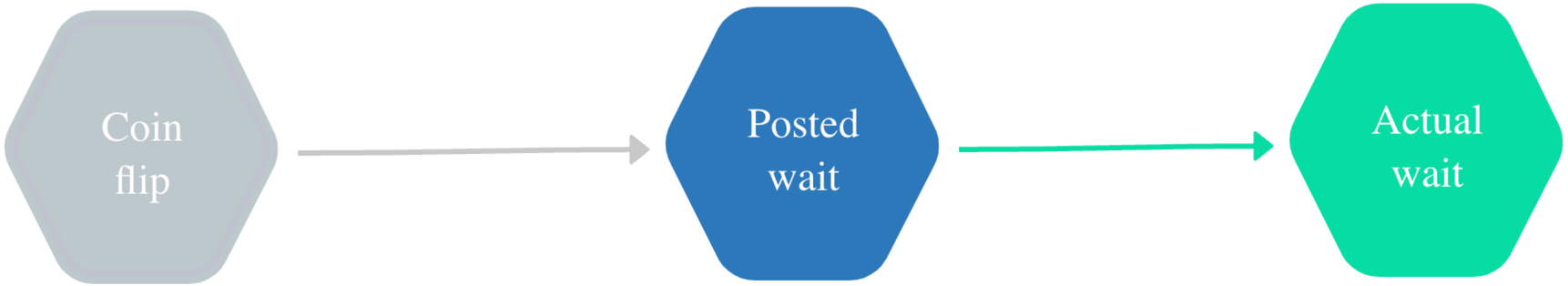
Ideal: RCT aka A/B test

- Every morning, flip a coin
  - Heads: set posted time to 60 minutes at 8am, measure actual wait time at 9am
  - Tails: set posted time to 30 minutes at 8am, measure actual wait time at 9am



Ideal: RCT aka A/B test

- Every morning, flip a coin
  - Heads: set posted time to 60 minutes at 8am, measure actual wait time at 9am
  - Tails: set posted time to 30 minutes at 8am, measure actual wait time at 9am



- Result: the **correlation** between posted and actual wait **is causation**

The data we have

- **TouringPlans** helps plan trips to Disney and Universal theme parks
  - Goal: accurately predict attraction wait times at these theme parks with data and statistical modeling

lucymcgowan.github.io

touringplans 0.0.1 Reference

touringplans

The goal of touringplans is to provide access to Disney World Ride Wait Time Datasets curated by the TouringPlans.com team.

Installation

You can install the development version of touringplans with:

```
devtools::install_github("LucyMcGowan/touringplans")
```

You can find a list of all data sets along with variable information on the touringplans package website

License

Full license

MIT + file LICENSE

Citation

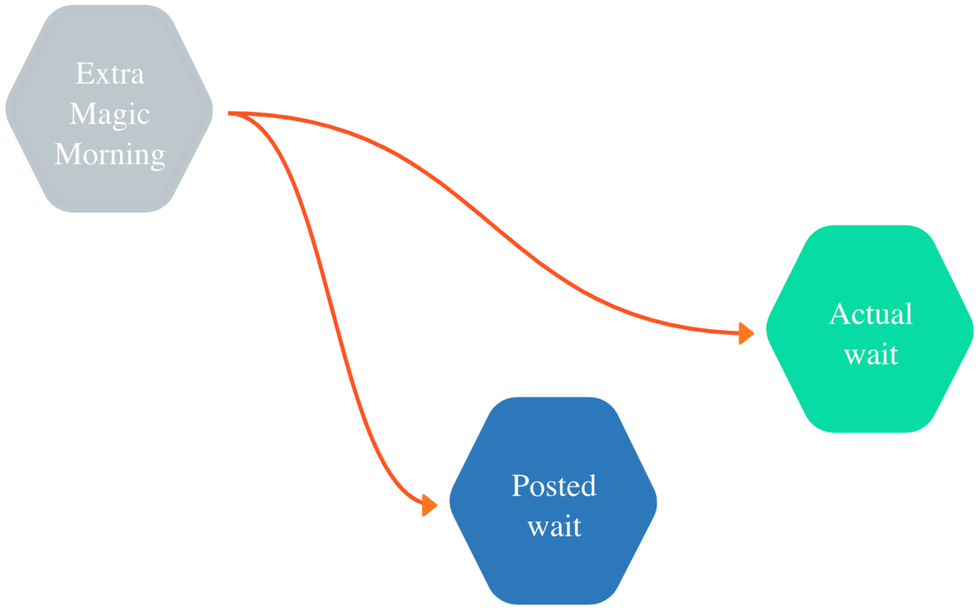
Citing touringplans

Developers

Lucy D'Agostino McGowan
Author, maintainer

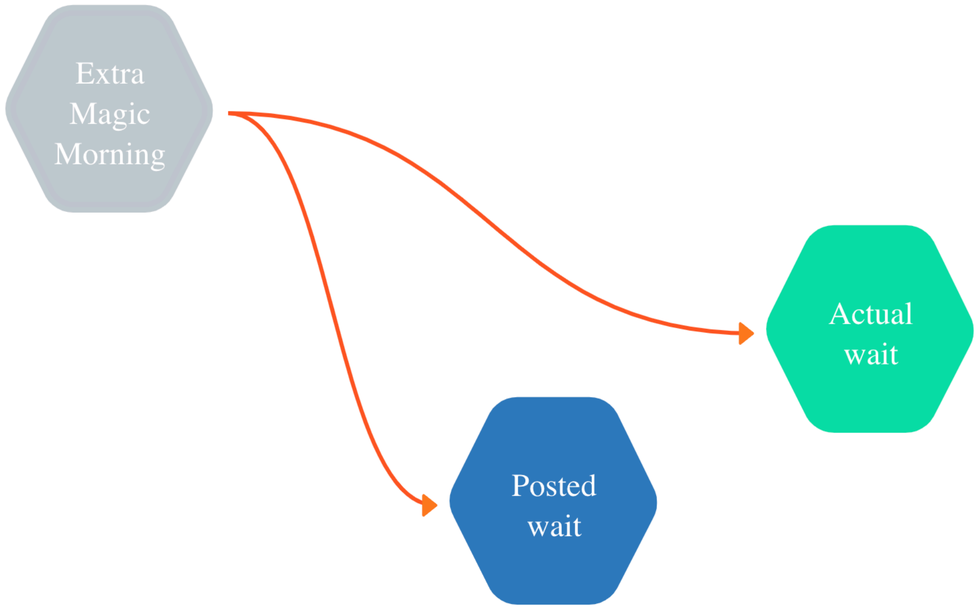
The observational study we can do

- We’re going to have to deal with some **confounders**
  - Common causes of exposure and outcome



The observational study we can do

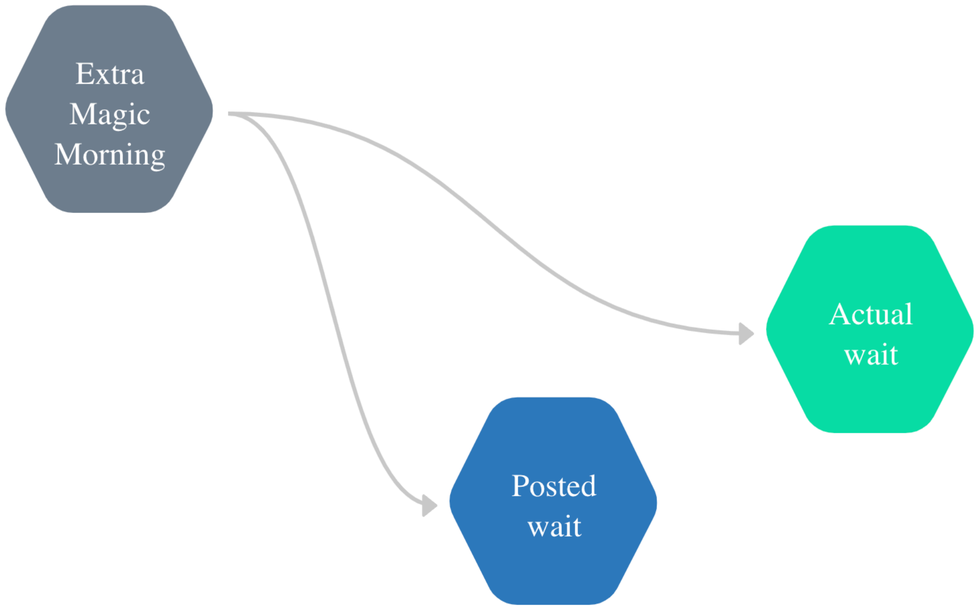
- We're going to have to deal with some **confounders**
  - Common causes of exposure and outcome
  - We need to **adjust/control** for their biasing effect




```
lm(
  wait_minutes_actual_avg ~ wait_minutes_posted_avg,
  data = touringplans::seven_dwarfs_train_2018
)
```

The observational study we can do

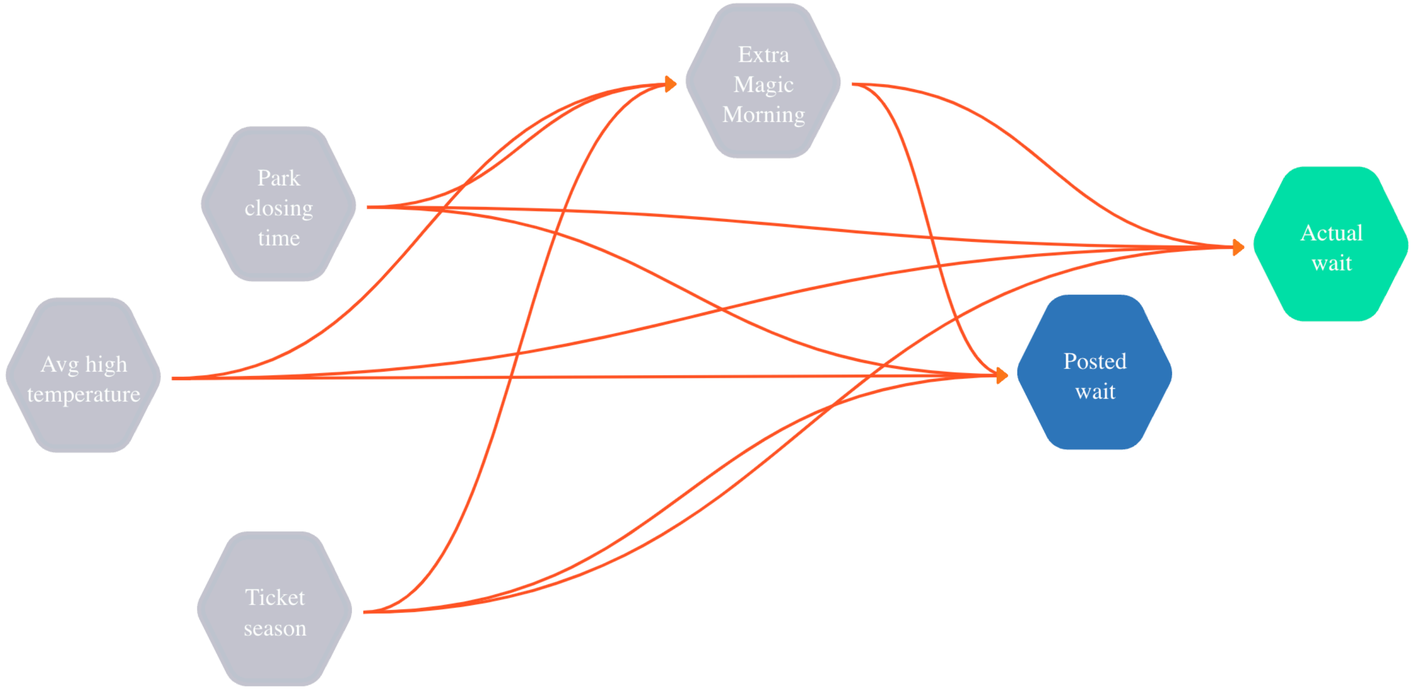
- We're going to have to deal with some **confounders**
  - Common causes of exposure and outcome
  - We need to **adjust/control** for their biasing effect




```
lm(
  wait_minutes_actual_avg ~
    wait_minutes_posted_avg + park_extra_magic_morning,
  data = touringplans::seven_dwarfs_train_2018
)
```

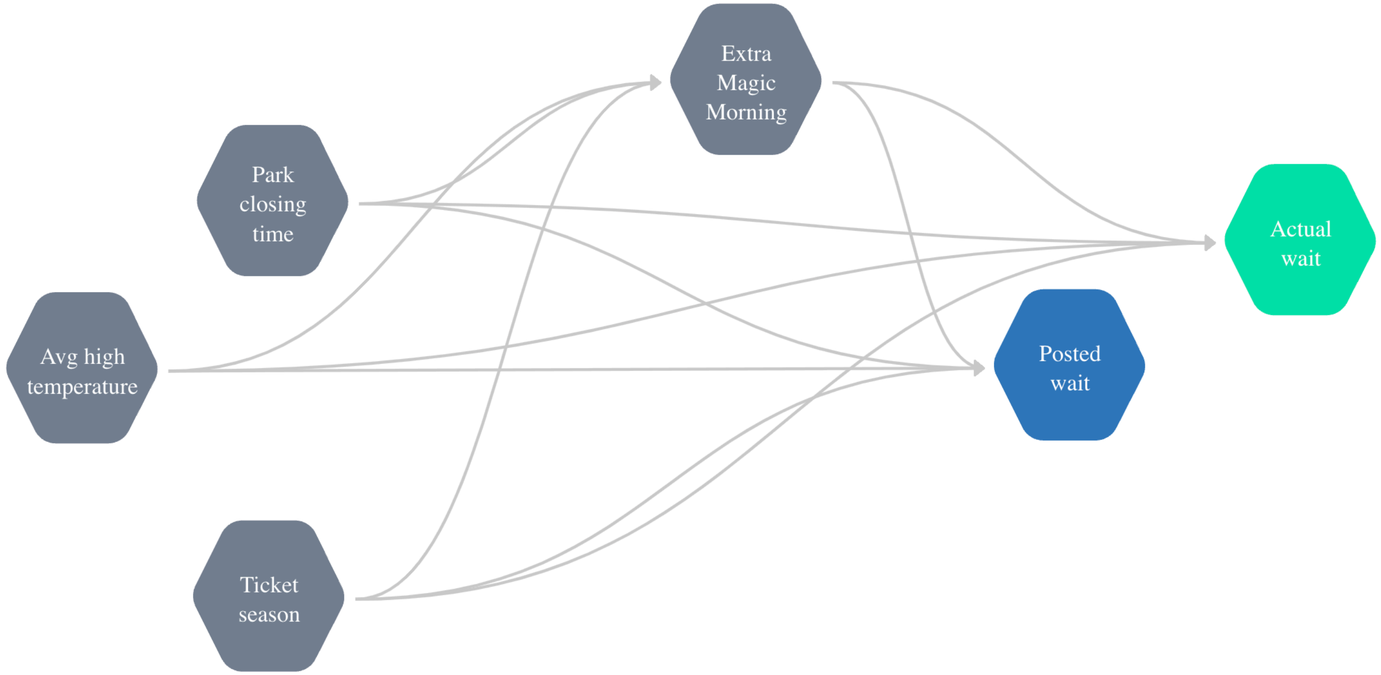
The observational study we can do

- We're going to have to deal with many **confounders**
  - Common causes of exposure and outcome
  - We need to **adjust/control** for their biasing effect



The observational study we can do

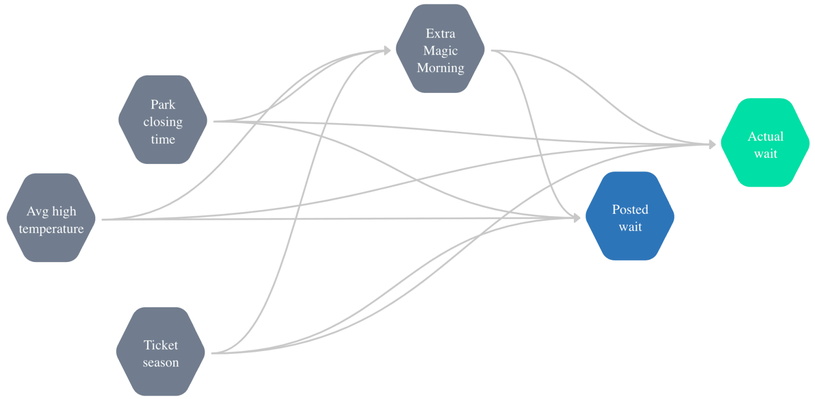
- We're going to have to deal with many **confounders**
  - Common causes of exposure and outcome
  - We need to **adjust/control** for their biasing effect



The kitchen sink linear model

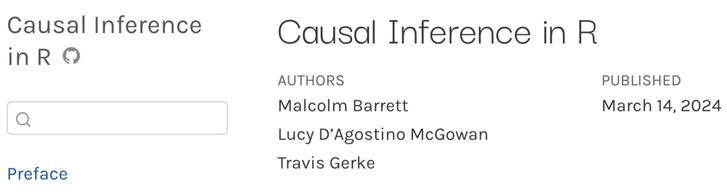
- Can we do this?
  - Yes, but many assumptions that we may not want to make (e.g. many conditional linear relationships encoded in this model)

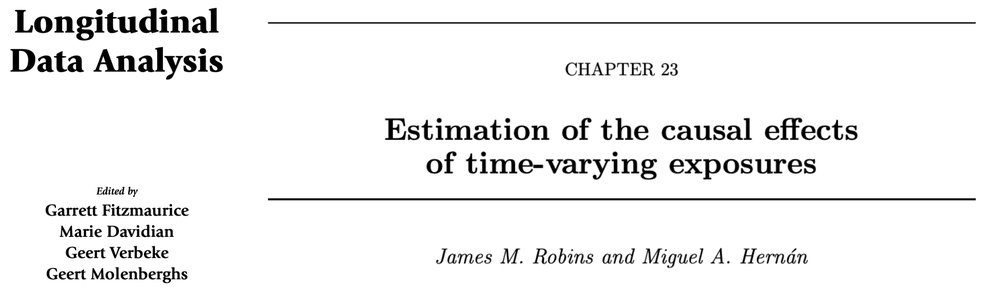
```
lm(
  wait_minutes_actual_avg ~
    wait_minutes_posted_avg + park_extra_magic_morning +
    park_close + park_ticket_season + park_temperature_high,
  data = touringplans::seven_dwarfs_train_2018
)
```



G-methods

- Inverse probability weighting
  - With weights (propensity scores), create a pseudo-population in which exposure is independent of confounders
- G estimation of a structural nested model
  - Uses conditional independence between exposure and potential outcomes to estimate nested models
- **G formula / G computation**
  - Model the joint distribution of observed data to generate potential outcomes under different exposures





The G-formula in 4 steps

1. Draw the time-ordered DAG
2. Pick a parametric model that predicts each variable's value based on previously measured variables in the DAG
3. Draw a large sample (with replacement) from the baseline variables, then simulate values for all subsequent variables based on the models from 2.
    - One key modification: for each exposure value you are interested in comparing, assign the exposure variables accordingly (that is, don't let the simulation assign values for exposure variables)
4. Compute the causal contrast of interest based on the simulated outcome in each exposure group.

Detour: **Monte Carlo** simulations

- Step 3 is a **Monte Carlo** simulation
  - **Def:** A computational approach that generates a sample of outcomes for random processes

- **Example:** What is the probability of rolling "snake eyes" (two ones) on a single roll of two six-sided dice? (Yes, we know it's 1/6 * 1/6 = 2.8%)

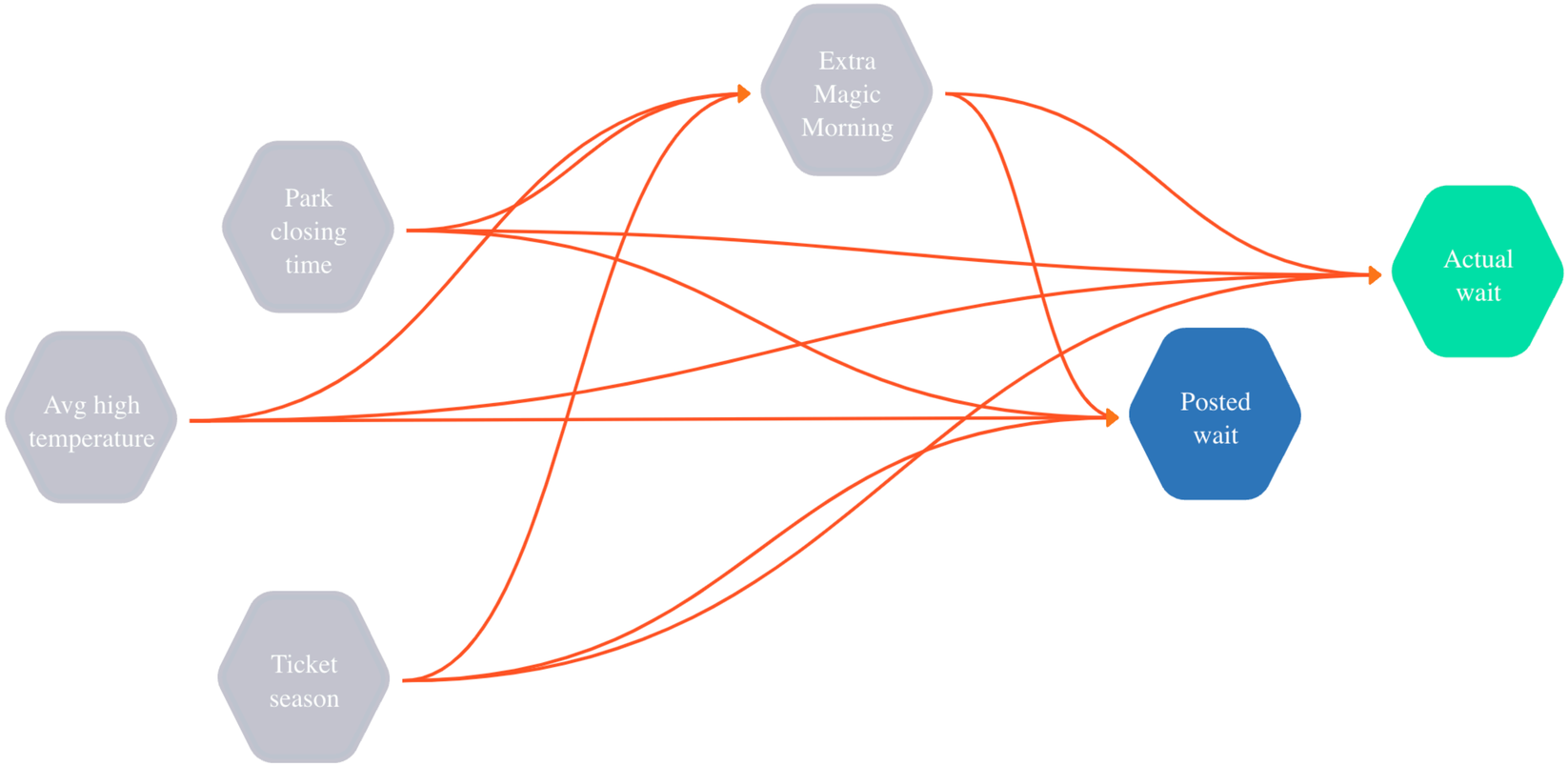
```
n <- 1000000
tibble(
  roll_1 = sample(1:6, n, replace = TRUE),
  roll_2 = sample(1:6, n, replace = TRUE),
) |>
  reframe(roll_1 + roll_2 == 2) |>
  pull() |>
  sum()/n

[1] 0.02761
```

The G-formula in 4 steps

1. Draw the time-ordered DAG
2. Pick a parametric model that predicts each variable's value based on previously measured variables in the DAG
3. Draw a large sample (with replacement) from the baseline variables, then simulate values for all subsequent variables based on the models from 2.
    - One key modification: for each exposure value you are interested in comparing, assign the exposure variables accordingly (that is, don't let the simulation assign values for exposure variables)
4. Compute the causal contrast of interest based on the simulated outcome in each exposure group.

Step 1: Draw the DAG

- Let's estimate the causal effect of setting posted wait time to 60 minutes compared to 30 minutes on actual wait time an hour later



Step 2: Pick parametric models

- We need a model for each node based on all previous nodes
  - For this example, the 3 non-baseline variables we need models for are: `park_extra_magic_morning`, `wait_minutes_posted_avg`, and `wait_minutes_actual_avg`

Step 2: Pick parametric models

```
# A logistic regression for park_extra_magic_morning
fit_extra_magic <- glm(
  park_extra_magic_morning ~
    park_ticket_season + park_close + park_temperature_high,
  data = .data,
  family = "binomial"
)

# A linear model for wait_minutes_posted_avg
fit_wait_minutes_posted <- lm(
  wait_minutes_posted_avg ~
    park_extra_magic_morning + park_ticket_season + park_close +
    park_temperature_high,
  data = .data
)

# A linear model for wait_minutes_actual_avg
fit_wait_minutes_actual <- lm(
  wait_minutes_actual_avg ~
    ns(wait_minutes_posted_avg, df = 3) +
    park_extra_magic_morning +
    park_ticket_season + park_close +
    park_temperature_high,
  data = .data
)
```

Step 3a: Draw a baseline sample

```
n_sample <- 10000
df_baseline <- touringplans::seven_dwarfs_train_2018 |>
  select(park_ticket_season, park_close, park_temperature_high) |>
  sample_n(n_sample, replace = TRUE)
```

Step 3b: Simulate the process

```
n_sample <- 10000
df_baseline <- touringplans::seven_dwarfs_train_2018 |>
  select(park_ticket_season, park_close, park_temperature_high) |>
  sample_n(n_sample, replace = TRUE)


# Simulate park_extra_magic_morning
df_sim_time_1 <- fit_extra_magic |>
  augment(newdata = df_baseline, type.predict = "response") |>
  # .fitted is the probability that park_extra_magic_morning is 1,
  # so let's use that to generate a 0/1 outcome
  mutate(
    park_extra_magic_morning = rbinom(n(), 1, .fitted)
  )

# Assign wait_minutes_posted_avg, since it's the exposure
df_sim_time_2 <- df_sim_time_1 |>
  mutate(
    wait_minutes_posted_avg = c(rep(60, n_sample/2), rep(30, n_sample/2))
  )

# Simulate the outcome
df_outcome <- fit_wait_minutes_actual |>
  augment(newdata = df_sim_time_2) |>
  rename(wait_minutes_actual_avg = .fitted)
```

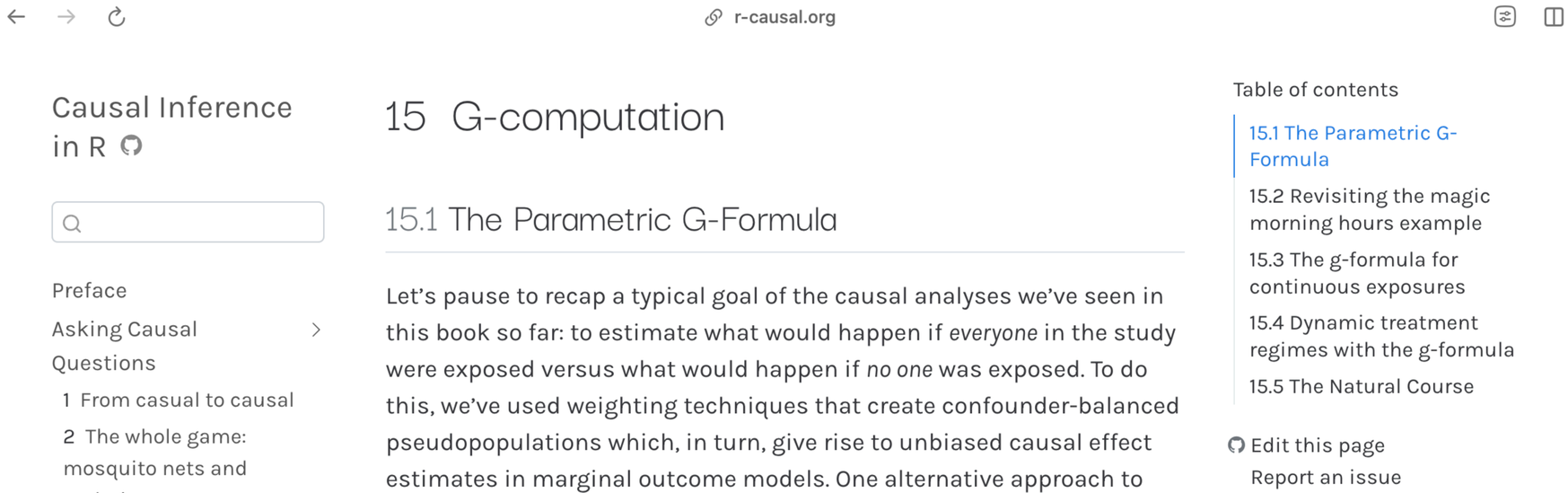
Step 4: Estimate the causal effect

```
df_outcome |>
  group_by(wait_minutes_posted_avg) |>
  summarize(avg_wait_actual = mean(wait_minutes_actual_avg)) |>
  pivot_wider(
    names_from = wait_minutes_posted_avg,
    values_from = avg_wait_actual,
    names_prefix = "x_"
  ) |>
  summarize(
    x_60, x_30, x_60 - x_30
  ) |>
  pivot_longer(
    names_to = "term",
    values_to = "estimate",
    cols = everything()
  )
```

```
# A tibble: 3 × 2
  term        estimate
  <chr>          <dbl>
1 x_60            29.9
2 x_30            40.6
3 x_60 - x_30    -10.7
```

Postscript: you'll need CIs

- Confidence intervals are estimated with the bootstrap
  - See Chapter 15 of Causal Inference in R for this example



Where the g-formula is **really** useful

- Time-varying exposures and confounding
  - Standard methods (i.e. the kitchen sink linear model we used) can't handle this

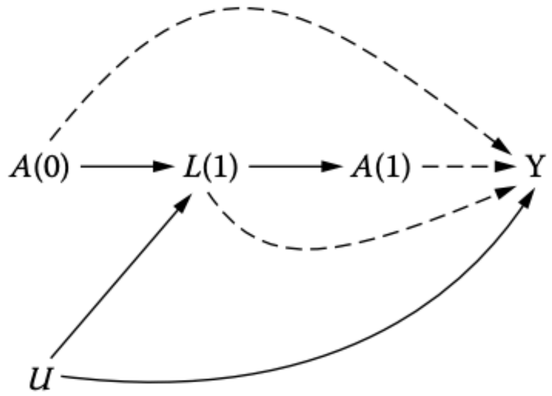


Figure 23.3 Causal DAG in the hypothetical study population.

Resources

- Robins / Hernan “Chapter 23”
  - https://www.hsph.harvard.edu/wp-content/uploads/sites/1268/2013/01/RobinsHernan_Chapter_23.pdf
- Causal Inference in R
  - https://www.r-causal.org/
- { ggdag }
  - https://r-causal.github.io/ggdag/