Online Appendix

Contents

A Visualizing Fowler et al.'s (2023) "Conversians' '

A.1 Figure 1

In this section we show how we simulated two-dimensional data for Figure 1. We use the following code:

```
draw.2d.model.data <- function() {
  n.respondents <- 2500
  n.issue.domains <- 2
  n.issue.questions.per.domain <- 50

  # generate two correlated dimensions
  data.2d <- data.frame(true.type = rep("Two-Dimensional", n.respondents))
  generate.cor <- rnorm(n.respondents)
  data.2d$latent.d1 <- .75 * rnorm(n.respondents) + .75 * generate.cor
  data.2d$latent.d2 <- .75 * rnorm(n.respondents) + .75 * generate.cor

  # generate preferences within each domain
  for (j in 1:n.issue.domains) {
    if (j == 1) {
      latent.issue.domain.views <- rnorm(n.respondents) * .5 + data.2d$latent.d1
    } else {
      latent.issue.domain.views <- rnorm(n.respondents) * .5 + data.2d$latent.d2 * .75
    }
    for (k in 1:n.issue.questions.per.domain) {
      # Rasch model translates latent issue views into binary outcome
      b <- rnorm(1) # difficulty parameter
      data.2d[, paste0("issueopinion_", j, ".issue", k)] <- as.numeric(
        exp(latent.issue.domain.views - b) / (1 + exp(latent.issue.domain.views - b)) >
          ↪ runif(length(latent.issue.domain.views))
      )
```

```
    }
  }

  mod <- data.2d %>%
    select(starts_with("issueopinion_")) %>%
    mirt(data = ., model = 2)
  scores <- fscores(mod)
  data.2d$d1 <- scores[, 1]
  data.2d$d2 <- scores[, 2]

  return(data.2d)
}
```

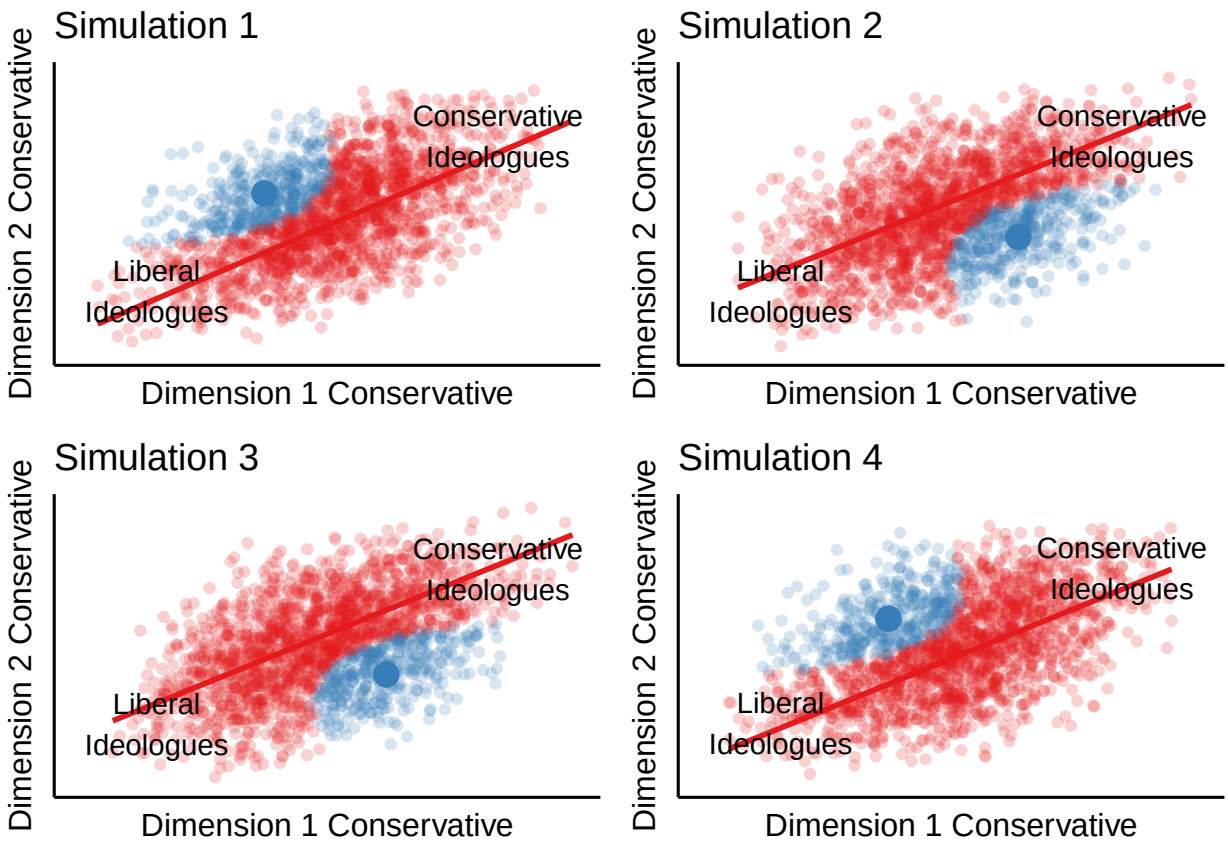
We then add the estimates from the Fowler et al. (2023) model and plot them alongside 2D IRT estimates of the two underlying dimensions from the mirt package. As shown and discussed in the main text, plotting these according to their first and second dimensions illustrates that the Moderates model categorizes respondents closer to a *single point* point it selects than the main one-dimensional line it estimates as Conversian.

```
make.fig1 <- function(sim.number) {
  data.2d.withests <- draw.2d.model.data() %>%
    add.moderates.estimates()

  data.2d.withests.downsians <- filter(data.2d.withests, modpaper_category == "Downsian")

  conversians.mean <- data.2d.withests %>%
    filter(modpaper_category == "Conversian") %>%
    summarize(d1 = mean(d1), d2 = mean(d2))

  text <- data.frame(
    text = c("Conservative\nIdeologues", "Liberal\nIdeologues"), # , 'Non-Ideologues',
     ↪  'Non-Ideologues'),
    d1 = c(2.75, -2.75), # , 2.75, -2.75),
    d2 = c(2.25, -2.25) # , -2.25, 2.25)
  )

  g <- data.2d.withests %>%
    sample_n(size = 2000) %>% # make graph readable
    mutate(modpaper_category = factor(modpaper_category,
      ordered = T, levels =
        c("Downsian", "Conversian", "Inattentive")
    )) %>%
    ggplot(aes(x = d1, y = d2)) +
    geom_point(aes(color = modpaper_category), alpha = .2) +
    geom_point(data = conversians.mean, aes(
      x = d1, y = d2, color = "Conversian",
      fill = "Mean of Conversians"
    ), size = 4) +
    geom_smooth(
      data = data.2d.withests.downsians,
      aes(
        x = d1, y = d2, color = "Downsian",
        linetype = "Line of Best Fit for Downsians"
      ),
      se = FALSE,
```

```
      formula = y ~ x,
      method = "lm"
    ) +
    scale_color_brewer('"Moderates" Model Output', palette = "Set1") +
    scale_linetype("") +
    scale_fill_brewer("") +
    ylab("Dimension 2 Conservative") +
    xlab("Dimension 1 Conservative") +
    guides(
      color = guide_legend(order = 1, override.aes = list(alpha = .3)),
      fill = guide_legend(override.aes = list(color = "#377EB8"), order = 2),
      linetype = guide_legend(override.aes = list(color = "#E41A1C"), order = 3)
    ) +
    theme_classic() +
    theme(
      axis.text.x = element_blank(), axis.ticks.x = element_blank(),
      axis.text.y = element_blank(), axis.ticks.y = element_blank()
    ) +
    geom_text(data = text, aes(label = text), size = 3.75) +
    expand_limits(x = c(-4, 4), y = c(-4, 4))
  return(g)
}

# select two examples where conversians are above the line and two examples where
 ↪  conversians are below the line
gs <- alply(1:8, 1, make.fig1, .parallel = TRUE)
above <- c()
below <- c()
for (i in 1:length(gs)) {
  if (mean(gs[[i]]$data$d2[gs[[i]]$data$conversian]) > 0) {
    above <- c(above, i)
  } else {
    below <- c(below, i)
  }
}

g <- ggpubr::ggarrange(gs[[above[1]]] + ggtitle("Simulation 1") + theme(legend.position =
 ↪  "none"),
  gs[[below[1]]] + ggtitle("Simulation 2") + theme(legend.position = "none"),
  gs[[below[2]]] + ggtitle("Simulation 3") + theme(legend.position = "none"),
  gs[[above[2]]] + ggtitle("Simulation 4") + theme(legend.position = "none"),
  ncol = 2, nrow = 2, legend = "none"
)
```

Furthermore, Figure OA1 shows that where "Conversians" are placed in this space varies across random simulations, as Conversians are not a meaningful group.

Figure OA1: Motivating Example: Multiple Simulations



A.1.1 Version of Figure 1 With Uncorrelated Dimensions

In this section we show that the results of Figure 1 look similar even if the two underlying dimensions are not correlated.

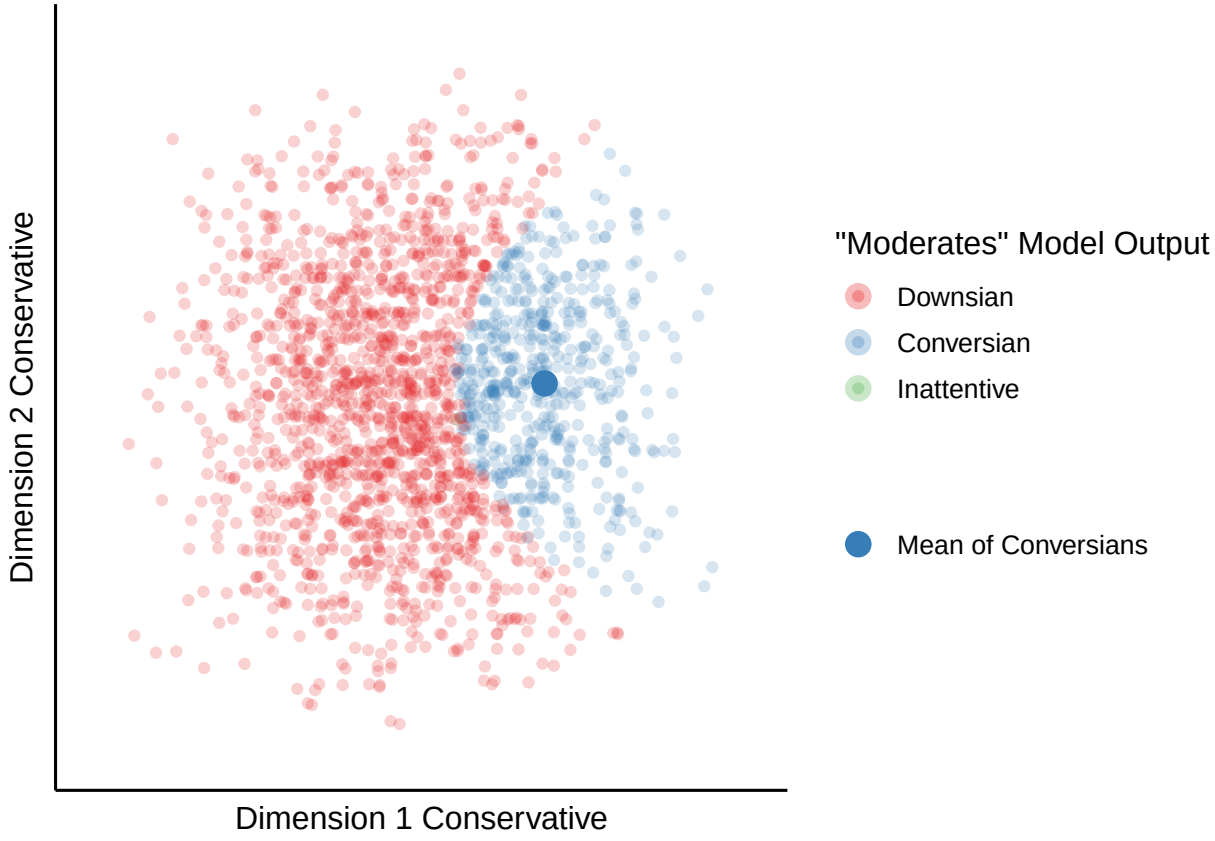
As before, we first simulate the data.

```
draw.2d.model.data <- function() {
  n.respondents <- 2500
  n.issue.domains <- 2
  n.issue.questions.per.domain <- 50

  # generate two correlated dimensions
  data.2d <- data.frame(true.type = rep("Two-Dimensional", n.respondents))
  data.2d$latent.d1 <- rnorm(n.respondents)
  data.2d$latent.d2 <- rnorm(n.respondents)

  # generate preferences within each domain
  for (j in 1:n.issue.domains) {
    if (j == 1) {
      latent.issue.domain.views <- rnorm(n.respondents) * .5 + data.2d$latent.d1
    } else {
      latent.issue.domain.views <- rnorm(n.respondents) * .5 + data.2d$latent.d2 * .75
    }
    for (k in 1:n.issue.questions.per.domain) {
      # Rasch model translates latent issue views into binary outcome
      b <- rnorm(1) # difficulty parameter
      data.2d[, paste0("issueopinion_", j, ".issue", k)] <- as.numeric(
        exp(latent.issue.domain.views - b) / (1 + exp(latent.issue.domain.views - b)) >
          ↪ runif(length(latent.issue.domain.views))
      )
    }
  }

  mod <- data.2d %>%
    select(starts_with("issueopinion_")) %>%
    mirt(data = ., model = 2)
  scores <- fscores(mod)
  data.2d$d1 <- scores[, 1]
  data.2d$d2 <- scores[, 2]

  return(data.2d)
}
```

As before, we then add the estimates from the Fowler et al. (2023) model and plot them alongside 2D IRT estimates of the two underlying dimensions from the mirt package. As shown and discussed in the main text, plotting these according to their first and second dimensions illustrates that the Moderates model categorizes respondents closer to a *single point* point it selects than the main one-dimensional line it estimates as Conversian.

```
data.2d.withests <- draw.2d.model.data() %>%
  add.moderates.estimates()
```

Figure OA2: Motivating Example With Uncorrelated Dimensions



A.2 Figure 2

In this section we show how we generated Figure 2, which made a version of the same plot but in the data used in Fowler et al. (2023). Note that the 2D IRT model from the mirt package is used strictly for the purposes of visualization and does not affect the estimates the "Moderates" model produces, which we draw directly from the article's replication data.

```
# Make plots
football.plot.moderates.data <- function(df) {
  source <- df %>%
    pull(source) %>%
    unique() %>%
    str_replace("_", " ")

  issues.mat <- df %>%
    select(cces2006_minimumwage:cces2017_buyamerican) %>%
    select_if(~ any(!is.na(.))) %>% # keep columns with any data, drop columns with all
     ↪  NA
    as.matrix()

  # 2d irt estimates from mirt package
  mod <- mirt(data = issues.mat, model = 2)
  scores <- fscores(mod)
  mirt.d1 <- scores[, 1]
  mirt.d2 <- scores[, 2]

  df <- df %>%
    mutate(
      d1 = mirt.d1,
      d2 = mirt.d2
    )

  # orient dimensions so larger values = more conservative
  if (cor(as.numeric(df$pid3), df$d1, use = "complete") < 0) df$d1 <- -1 * df$d1
  if (cor(as.numeric(df$pid3), df$d2, use = "complete") < 0) df$d2 <- -1 * df$d2

  df <- df %>%
    dplyr::rename(
      downs.prob = w1,
      conversian.prob = w2,
      inattentive.prob = w3
    ) %>%
    mutate(
      downsian = downs.prob > inattentive.prob & downs.prob > conversian.prob,
      conversian = conversian.prob > inattentive.prob & conversian.prob > downs.prob,
      inattentive = inattentive.prob > conversian.prob & inattentive.prob > downs.prob,
      modpaper_category = case_when(
        downsian ~ "Downsian",
        conversian ~ "Conversian",
        inattentive ~ "Inattentive",
        TRUE ~ NA_character_
      )
    )

  df.downsians <- filter(df, downsian)
```

```
  g <- df %>%
    sample_n(2500) %>% # make graphs readable
    mutate(modpaper_category = factor(modpaper_category,
      ordered = T, levels =
        c("Downsian", "Conversian", "Inattentive")
    )) %>%
    ggplot(aes(x = d1, y = d2)) +
    geom_point(aes(color = modpaper_category), alpha = .2, size = 0.5) +
    geom_smooth(
      data = df.downsians,
      aes(
        x = d1, y = d2, color = "Downsian",
        linetype = "Line of Best Fit"
      ),
      formula = y ~ x,
      se = FALSE, method = "lm"
    ) +
    scale_color_brewer('"Moderates" Model Output', palette = "Set1") +
    scale_linetype("") +
    scale_fill_manual("", values = "black") +
    ggtitle(source) +
    xlab("D1 Conservatism") +
    ylab("D2 Conservatism") +
    guides(
      color = guide_legend(order = 1),
      fill = FALSE, # guide_legend(override.aes = list(color = "#377EB8"), order = 2),
      linetype = FALSE
    ) + # guide_legend(override.aes = list(color = '#E41A1C'), order = 3)) +
    theme_classic() +
    theme(
      axis.text.x = element_blank(), axis.ticks.x = element_blank(),
      axis.text.y = element_blank(), axis.ticks.y = element_blank(),
      legend.position = "bottom"
    )

  return(g)
}

data <- read.dta(paste0(wd, "/Moderates Paper and Replication Archive/Moderates
 ↪  Replication Archive/bigsurveys_recoded4.dta"))
gs <- dlply(data, .(source), football.plot.moderates.data, .parallel = TRUE)
```

B Simulations

In this Appendix section we describe details of the simulations described in Table 1 (simulation 1) and the extended simulations reported in Table OA1 (simulation 2).

B.1 Global Parameters

```
set.seed(95821)
n.respondents <- 5000
n.issue.domains <- 15
n.issue.questions.per.domain <- 5
n.questions.total <- n.issue.domains * n.issue.questions.per.domain
```

Both simulations contain data from one or multiple DGPs. Each DGP has 5000 respondents. There are 15 issue domains represented on the survey. They are asked 5 questions in each issue domain, for a total of 75 issue questions.

B.2 Main Text Simulation (Simulation 1)

B.2.1 DGPs

Below we show the code used to generate the simulated survey responses in each DGP. DGPs 1-3 are used in the paper's main text. The remaining DGPs are used in a second simulation below.

The simulation code for each DGP below simulates the latent views of respondent i on issue j, denoted $\theta_{i,j}$, for each question for each respondent in the DGP. Note that these θ parameters are latent parameters representing respondent's true latent preferences specific to each respondent-issue. Further details about how these are translated into binary responses to a simulated survey instrument are given in the following subsection.

B.2.1.1 DGP 1: Non-Ideologues

Voters simulated in this DGP have genuine views in each policy domain but these are uncorrelated with their views in other policy domains.

```
data.non.ideologues <- data.frame(true.type = rep("No Ideological Constraint",
 ↪  n.respondents))
for (j in 1:n.issue.domains) {
  latent.issue.domain.views <- rnorm(n.respondents)
  for (k in 1:n.issue.questions.per.domain) {
    data.non.ideologues[, paste0("latentissueopinion_domain", j, ".issue", k)] <-
 ↪  latent.issue.domain.views
  }
}
```

B.2.1.2 DGP 2: Some Ideological Constraint

Voters simulated in this DGP's views in each policy domain are equally influenced by genuine views specific to each policy domain and a cross-domain ideology. This generates the presence of some "ideological constraint" in their beliefs.

```
# Note: parameter 1/sqrt(2) chosen because it satisfies the system of equations x^2 + x^2
 ↪  = 1^2 (so that latent issue opinions have the same SD in these DGPs as in
 ↪  data.non.ideologues from DGP 1).

data.some.constraint <- data.frame(true.type = rep("Some Ideological Constraint",
 ↪  n.respondents))
latent.ideology <- rnorm(n.respondents)
for (j in 1:n.issue.domains) {
  latent.issue.domain.views <- 1 / sqrt(2) * rnorm(n.respondents) + 1 / sqrt(2) *
 ↪  latent.ideology
```

```
  for (k in 1:n.issue.questions.per.domain) {
    data.some.constraint[, paste0("latentissueopinion_domain", j, ".issue", k)] <-
 ↪  latent.issue.domain.views
  }
}
```

B.2.1.3 DGP 3: One-Dimensional Ideologues

This simulation models one-dimensional ideologues. Ideology is perfectly predictive of their latent views in every issue domain.

```
data.ideologues <- data.frame(true.type = rep("Pure Ideologues", n.respondents))
ideology <- rnorm(n.respondents)
for (j in 1:n.issue.domains) {
  for (k in 1:n.issue.questions.per.domain) {
    data.ideologues[, paste0("latentissueopinion_domain", j, ".issue", k)] <- ideology
  }
}
```

B.2.2 Converting θs to responses (for non-inattentive types)

We next probabilistically convert the respondent i's $\theta_{i,j}$ parameters on each issue j to binary issue responses to each issue question using a 1 parameter IRT model (Rasch model). In this model the questions' difficulty parameters (b) are universal across DGPs and respondents. The formula for the probability that respondent i responds Yes to question j is thus

$$P(X = 1|\theta_{i,j}, b_j) = \frac{e^{\theta_{i,j}-b_j}}{1 + e^{\theta_{i,j}-b_j}}$$

.

To convert the θ parameters for each respondent on each question to responses, we (1) first calculate the probability the respondent would answer Yes to the question using a Rasch model and then (2) to represent measurement error in the observed responses, record the answer as Yes with that probability.

This is implemented as follows:

```
# draw question difficulty parameters
bs <- rnorm(n.questions.total)

# function to implement Rasch model
convert.column.to.binary <- function(theta.vec, b) {
  as.numeric(
    exp(theta.vec - b) / (1 + exp(theta.vec - b)) > runif(length(theta.vec))
  )
}
```

We apply this function to all the DGPs above (except the Inattentives DGP, where response probabilities are simply 0.5). To increase realism, half the issues are reverse coded as well.

```
reverse.coded.issue.cols <- sample(1:n.questions.total, n.questions.total / 2) + 1 +
 ↪  n.questions.total

implement.rasch.model <- function(df) {
  df.names <- names(df)

  # implement rasch model
  for (i in 2:ncol(df)) {
    df[, paste0("issueopinion_binary_", df.names[i])] <-
```

```
      convert.column.to.binary(df[, i], bs[i - 1]) # first column of data is the
       ↪  true.type label
  }

  # reverse code some items for realism
  for (i in reverse.coded.issue.cols) df[, i] <- 1 - df[, i]

  return(df)
}

df.list <- list(
  data.non.ideologues,
  data.some.constraint,
  data.ideologues
)
result.list <- lapply(df.list, implement.rasch.model)
```

B.2.3 Simulation Results

Finally, we generate the results shown in Table 1 in the main text.

```
# Add moderates estimates
sim1.results <- do.call(rbind.data.frame, result.list) %>%
  add.moderates.estimates()

# Helper functions to print results
is.in.middle.tercile <- function(x) {
  qtiles <- quantile(x, probs = c(1 / 3, 2 / 3), na.rm = T)
  return(x >= qtiles[1] & x <= qtiles[2])
}

calculate.moderate.issue.share <- function(df) {
  issue.dvs <- names(df)[str_starts(names(df), "latentissueopinion_domain")]
  for (idv in issue.dvs) df[, paste0(idv, "_moderate")] <- is.in.middle.tercile(df[,
   ↪  idv])
  share.issue.views.moderate <- rowMeans(df[, names(df)[str_ends(names(df),
 ↪  "_moderate")]], na.rm = T)
  return(share.issue.views.moderate)
}

print.results <- function(df, caption = "") {
  df$share.issue.views.moderate <- calculate.moderate.issue.share(df)

  df %>%
    group_by(true.type) %>%
    dplyr::summarize(
      `Share of True Latent Preferences Which Are Moderate` =
       ↪  mean(share.issue.views.moderate),
      Downsian = mean(downsian),
      `Downsian 'Moderate'` = mean(downsian.moderate, na.rm = T),
      Conversian = mean(conversian),
      Inattentive = mean(inattentive)
    ) %>%
```

```
    dplyr::rename(`True Type` = true.type) %>%
    mutate_at(vars(!`True Type`), scales::percent, accuracy = 1) %>%
    kable(caption = caption, position = "H", format = "latex", booktabs = T) %>%
    add_header_above(header = c(
      "Ground Truth in DGP" = 2,
      "Moderates Paper: Share Categorized As..." = 4
    )) %>%
    column_spec(1, bold = T) %>%
    column_spec(2, width = "7em") %>%
    column_spec(4, width = "5em")
}

table1 <- sim1.results %>%
  print.results(caption = "Simulation 1 Results")
```

See Section B.5 below for details on the robustness of this simulation's results across values of the hyperparameters.

B.3 Simulation 2

As noted in the main text, in the Online Appendix we also conduct a second simulation which adds several additional DGPs.

B.3.1 Additional DGPs

We add the following new DGPs in this simulation.

B.3.1.1 DGP 4: Left Pure Partisans Left pure partisans have a θ of 2 on every item.

```
data.leftpartisans <- data.frame(true.type = rep("Left Pure Partisan", n.respondents))
for (j in 1:n.issue.domains) {
  for (k in 1:n.issue.questions.per.domain) {
    data.leftpartisans[, paste0("latentissueopinion_domain", j, ".issue", k)] <- 2
  }
}
```

B.3.1.2 DGP 5: Right Pure Partisans Right pure partisans have a θ of -2 on every item.

```
data.rightpartisans <- data.frame(true.type = rep("Right Pure Partisan", n.respondents))
for (j in 1:n.issue.domains) {
  for (k in 1:n.issue.questions.per.domain) {
    data.rightpartisans[, paste0("latentissueopinion_domain", j, ".issue", k)] <- -2
  }
}
```

B.3.1.3 DGP 6: Libertarians This DGP models a group of "off-dimensional" voters who all share a *common* system of correlated beliefs across questions. A motivating example case would be libertarians. We model them has share a common set of views in each of the 15 issue domains which are correlated within those domains.

```
data.libertarians <- data.frame(true.type = rep("Libertarians", n.respondents))
shared.latent.issue.views <- rnorm(n.issue.domains, sd = 2.5)
for (j in 1:n.issue.domains) {
  for (k in 1:n.issue.questions.per.domain) {
    data.libertarians[, paste0("latentissueopinion_domain", j, ".issue", k)] <-
↪   shared.latent.issue.views[j] # All respondents share the same latent views in issue
↪   domain j
```

```
  }
}
```

B.3.1.4 DPG 7: Inattentives

Finally, we also create a DGP for inattentive respondents. Their response probabilities for every question are 0.5; they are unaffected by the difficulty parameter of the question.

```
data.inattentive <- data.frame(true.type = rep("Inattentive", n.respondents))
for (j in 1:n.issue.domains) {
  for (k in 1:n.issue.questions.per.domain) {
    data.inattentive[, paste0("issueopinion_binary_latentissueopinion_domain", j,
 ↪  ".issue", k)] <- rbinom(n.respondents, 1, .5)
  }
}
```

B.3.2 Results

The results are as follows:

```
df.list <- list(
  # Previously used DGPs
  data.non.ideologues,
  data.some.constraint,
  data.ideologues,
  # New DGPs (except inattentives, added below)
  data.leftpartisans,
  data.libertarians,
  data.rightpartisans
)
result.list <- lapply(df.list, implement.rasch.model)

sim2.results <- do.call(rbind.data.frame, result.list) %>%
  plyr::rbind.fill(data.inattentive) %>% # Inattentive type does not have Rasch model
 ↪  applied as probability for all questions is 0.5.
  add.moderates.estimates()

sim2.results %>%
  print.results(caption = "Simulation 2 Results")
```

Ground Truth in DGP		Moderates Paper: Share Categorized As...			
True Type	Share of True Latent Preferences Which Are Moderate	Downsian	Downsian 'Moderate'	Conversian	Inattentive
Inattentive	NA	2%	2%	0%	98%
Left Pure Partisan	13%	100%	0%	0%	0%
Libertarians	67%	0%	0%	100%	0%
No Ideological Constraint	52%	97%	81%	1%	2%
Pure Ideologues	52%	100%	35%	0%	0%
Right Pure Partisan	20%	100%	0%	0%	0%
Some Ideological Constraint	52%	99%	48%	0%	1%

Table OA1: Simulation 2 Results

B.3.3 Implications

This simulation intends to illustrate two things.

First, it attempts to illustrate that the Moderates model's Conversian mode "formalizes a single cluster of voters with a distinctive pattern of views." Libertarians are an extreme example of this idea. And as can be seen above, 100% of libertarians were categorized as Conversians in this particular simulation (although the behavior of the model is unstable). Likewise, the Figure below shows that, by plotting the estimated first and second dimenions from Simulation 2 as estimated using a 2D IRT model, it can be seen visually that Conversians form a single cluster of "those who are far from the first dimension in a similar way." (Note that the 2D IRT model from the mirt package is used strictly for the purposes of visualization and does not affect the estimates the "Moderates" model produces.)

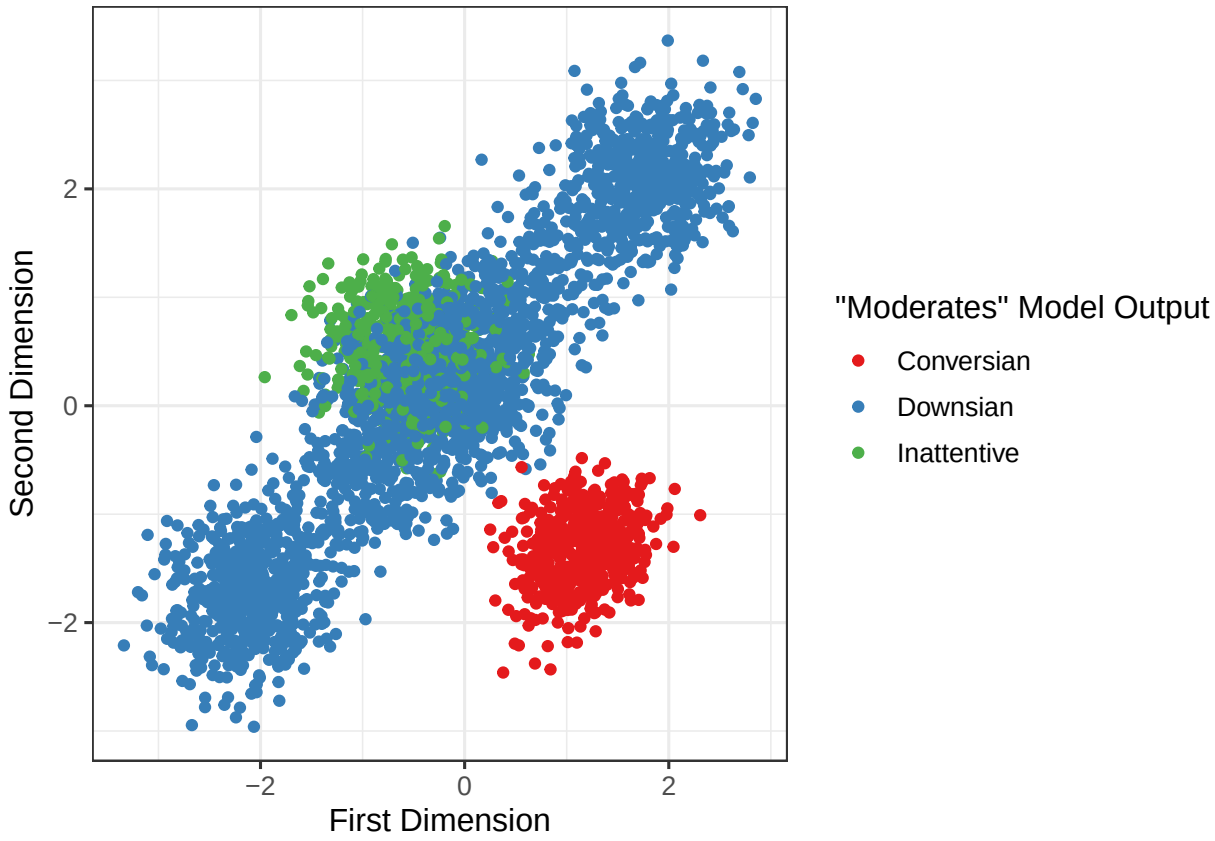
Second—and much more importantly—Simulation 2 illustrates the Fowler et al. (2023)'s model's tendency to absorb non-ideologues into the ideological type in cases when the non-ideological type has already been "occupied" with a particular type of respondents. In Simulation 2, because libertarians are categorized as Conversians, the values of λ_j are set to extreme values consistent with libertarians. However, this forces almost all remaining respondents whose responses are neither ideological nor libertarian to nevertheless be categorized as ideologues, because the Conversian category is already "occupied' ' with libertarians. This illustration is made in service of our broader point: If the restrictive nature of the non-ideological type leads many actually non-ideological respondents to often be characterized as ideological, that is indicative of the model not being reliable at telling us how many ideologues there are, and consequently how many ideological moderates there are.

```
# 2d irt estimates from mirt package
irt2d.mod <- sim2.results %>%
  select(starts_with("issueopinion_binary_")) %>%
  mirt(data = ., model = 2)
scores <- fscores(irt2d.mod)

sim2.results$d1 <- scores[, 1]
sim2.results$d2 <- scores[, 2]
```

```
sim2.results %>%
  sample_frac(.1) %>%
  ggplot(aes(x = d1, y = d2, color = modpaper_category)) +
  xlab("First Dimension") +
  ylab("Second Dimension") +
  scale_color_brewer('"Moderates" Model Output', palette = "Set1") +
```

```
  geom_point() +
  theme_bw()
```

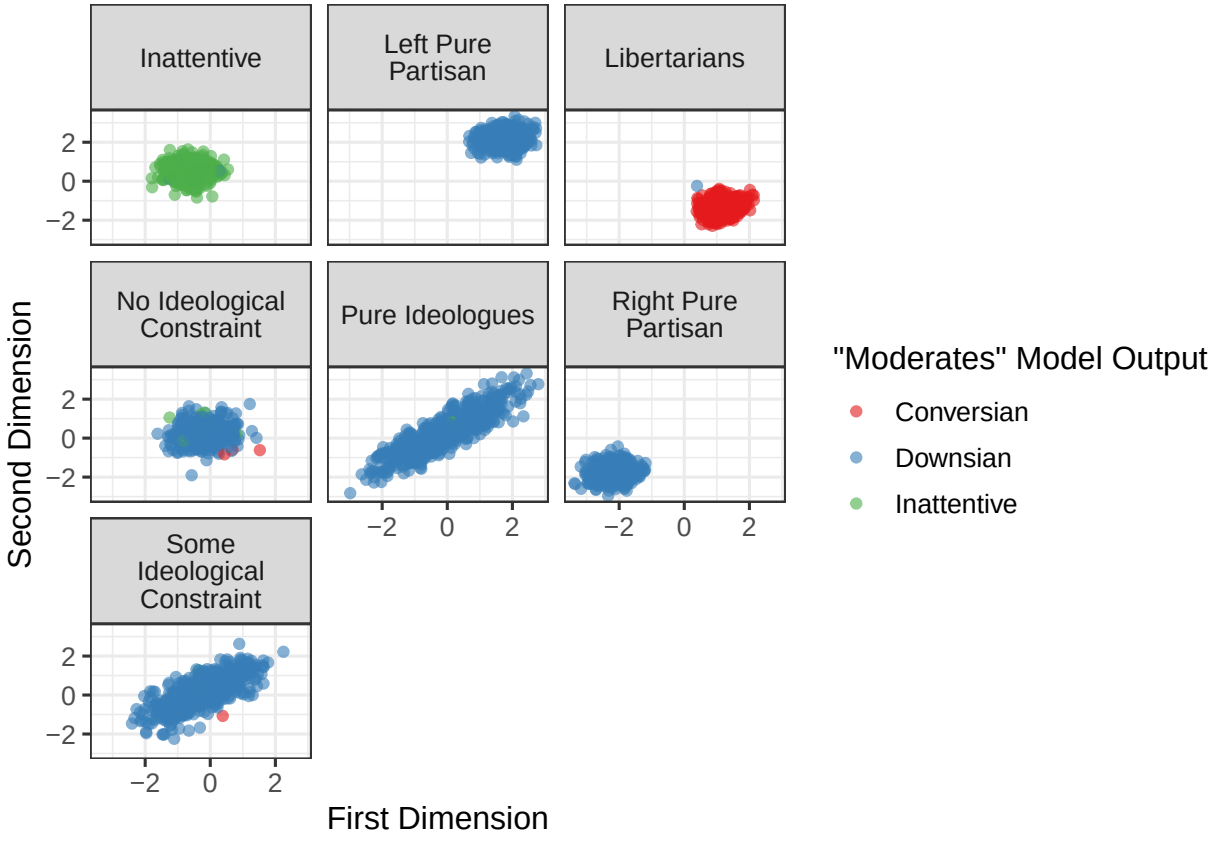
Figure OA3: Moderates Paper's Model When Applied to Simulation 2, Visualized



The below graph splits this by voters' true underlying types, shown on the top of the facets:

```
sim2.results %>%
  mutate(true.type = str_wrap(true.type, 15)) %>%
  sample_frac(.1) %>%
  ggplot(aes(x = d1, y = d2, color = modpaper_category)) +
  geom_point(alpha = .6) +
  xlab("First Dimension") +
  ylab("Second Dimension") +
  scale_color_brewer('"Moderates" Model Output', palette = "Set1") +
  facet_wrap(~true.type) +
  theme_bw()
```

Figure OA4: Moderates Paper's Model When Applied to Simulation 2, Visualized, Split by True Type



B.4 Simulation 3

B.4.1 DGP 8: Non-1D Constrained by Views Towards Social Groups

Our third simulation pursues a different approach than the first two. In this simulation, the source of the correlations between non-ideological respondents' views across issues is *not* that each question is a function of one of many possible uncorrelated issue domains (as in Simulations 1 and 2). Instead, drawing on Converse (1964), in this DGP there are instead 25 social groups towards which respondents may have attitudes. For each of the 25 social groups, there are a random 1000 of the 5000 respondents whose views toward the social group affect their attitudes on a random subset of 20 of the issue questions. For example, suppose one of the social groups is unions, that most people don't think about unions when they answer survey questions, but for a minority of respondents, their attitudes about unions inform their views on some of the survey questions (and there are 24 additional such groups). This produces correlations across questions (i.e., constraint across questions) for reasons unrelated to one-dimensional ideology. This allows us to test whether the bias in the "Moderates" model we discuss only arises when there are multiple groups of multiple issue questions which are each are determined by views in a single latent issue dimension within each group of questions (as in the prior simulation), or whether it also arises when a single question can be affected by multiple latent dimensions in public attitudes (e.g., towards multiple different groups). I.e., in this DGP, issue questions are not nested within single policy domains but rather affected by multiple different latent attitudes.

We simulate this DGP as follows:

```
data.non.1d.constraint <- data.frame(true.type = rep("Non-1D Constraint", n.respondents))

latent.issue.question.views <- matrix(nrow = n.respondents, ncol = n.issue.questions)
for (j in 1:n.issue.questions) latent.issue.question.views[, j] <- 0

for (i in 1:n.social.groups) {
  respondents.in.group <- sample(1:n.respondents, respondents.per.group)
  relevant.issues <- sample(1:n.issue.questions, n.relevant.issues.per.group)
  respondent.level.views <- rnorm(respondents.per.group)
  for (j in relevant.issues) {
    latent.issue.question.views[respondents.in.group, j] <-
 ↪  latent.issue.question.views[respondents.in.group, j] + respondent.level.views
  }
}

for (j in 1:n.issue.questions) data.non.1d.constraint[, paste0("latentissueopinion_", j)]
 ↪  <- latent.issue.question.views[, j] / sd(latent.issue.question.views[, j]) #
 ↪  standardize to SD 1
```

We also again simulate a DGP of 5000 pure ideologues.

```
data.ideologues <- data.frame(true.type = rep("Pure Ideologues", n.respondents))
ideology <- rnorm(n.respondents)
for (j in 1:n.issue.questions) {
  data.ideologues[, paste0("latentissueopinion_", j)] <- ideology
}
```

We perform the same mapping of latent attitudes to binary survey questions as employed in the previous simulations.

```
# draw question difficulty parameters
bs <- rnorm(n.issue.questions)

# function to implement Rasch model
convert.column.to.binary <- function(theta.vec, b) {
  as.numeric(
    exp(theta.vec - b) / (1 + exp(theta.vec - b)) > runif(length(theta.vec))
```

```
  )
}
```

```
reverse.coded.issue.cols <- sample(1:n.issue.questions, n.issue.questions / 2) + 1 +
 ↪  n.issue.questions

implement.rasch.model <- function(df) {
  df.names <- names(df)

  # implement rasch model
  for (i in 2:ncol(df)) {
    df[, paste0("issueopinion_binary_", df.names[i])] <-
      convert.column.to.binary(df[, i], bs[i - 1]) # first column of data is the
       ↪  true.type label
  }

  # reverse code some items for realism
  for (i in reverse.coded.issue.cols) df[, i] <- 1 - df[, i]

  return(df)
}

df.list <- list(
  data.non.1d.constraint,
  data.ideologues
)
result.list <- lapply(df.list, implement.rasch.model)
```

B.4.2 Simulation Results

The results are similar to those in the previous simulations. Individuals in the DGP which have constraint in their attitudes for reasons unrelated to 1D ideology (i.e., multidimensional constraint driven by attitudes towards groups, as Converse envisioned) are again overwhelmingly miscategorized as one-dimensional Downsians despite that they are not one-dimensional ideologues. This is driven by the fact that the "Moderates" model does not allow individuals' views within the Conversian category to be correlated across issues.

Ground Truth in DGP	Moderates Paper: Share Categorized As...		
True Type	Downsian	Conversian	Inattentive
Non-1D Constraint	90%	10%	0%
Pure Ideologues	98%	2%	0%

Table OA2: Simulation 3 Results

B.5 Robustness of Simulation 1 Results

In this section we demonstrate the robustness and scope conditions associated with the main simulation in the main text (Simulation 1, reported in Table 1 in the main text and discussed in further detail in section B.2 above). In particular, we demonstrate its robustness to a) random chance (demonstrating that the results of the particular simulation we showed in Table 1 were not unusual) and to b) alternative hyperparameters (in particular the number of issue domains and the number of questions per issue domain). In addition, consistent with our argument in the main text, we also show that the extent to which the Fowler et al. (2023) model overestimates the share of non-ideological voters who are Downsians depends upon the presence of correlations between the questions for reasons unrelated to one-dimensional ideology.

We use the following code to perform these simulations, which mirrors the approach in Simulation 1 but allows for specifying the hyperparameters dynamically.

```
# Returns the share of non-ideologues misclassified as Doumsians given a number of
 ↪  respondents, number of issue domains, and nubmer of issue questions per domain.
sim1.robustness.sim <- function(n.issue.domains, n.issue.questions.per.domain) {
  n.respondents <- 1000
  n.questions.total <- n.issue.domains * n.issue.questions.per.domain

  data.non.ideologues <- data.frame(true.type = rep("No Ideological Constraint",
 ↪  n.respondents))
  for (j in 1:n.issue.domains) {
    latent.issue.domain.views <- rnorm(n.respondents)
    for (k in 1:n.issue.questions.per.domain) {
      data.non.ideologues[, paste0("latentissueopinion_domain", j, ".issue", k)] <-
 ↪  latent.issue.domain.views
    }
  }

  data.some.constraint <- data.frame(true.type = rep("Some Ideological Constraint",
 ↪  n.respondents))
  latent.ideology <- rnorm(n.respondents)
  for (j in 1:n.issue.domains) {
    latent.issue.domain.views <- 1 / sqrt(2) * rnorm(n.respondents) + 1 / sqrt(2) *
 ↪  latent.ideology
    for (k in 1:n.issue.questions.per.domain) {
      data.some.constraint[, paste0("latentissueopinion_domain", j, ".issue", k)] <-
 ↪  latent.issue.domain.views
    }
  }

  data.ideologues <- data.frame(true.type = rep("Pure Ideologues", n.respondents))
  ideology <- rnorm(n.respondents)
  for (j in 1:n.issue.domains) {
    for (k in 1:n.issue.questions.per.domain) {
      data.ideologues[, paste0("latentissueopinion_domain", j, ".issue", k)] <- ideology
    }
  }

  # draw question difficulty parameters
  bs <- rnorm(n.questions.total)
  reverse.coded.issue.cols <- sample(1:n.questions.total, n.questions.total / 2) + 1 +
 ↪  n.questions.total

  df.list <- list(
    data.non.ideologues,
    data.some.constraint,
    data.ideologues
  )
  result.list <- lapply(df.list, implement.rasch.model, bs, reverse.coded.issue.cols,
 ↪  n.questions.total)

  # Add moderates estimates
  sim1.results <- do.call(rbind.data.frame, result.list) %>%
```

```
    add.moderates.estimates()

  share.non.ideologues.categorized.as.downsian <- sim1.results %>%
    filter(true.type == "No Ideological Constraint") %>%
    pull(downsian) %>%
    mean()

  return(c(share.non.ideologues.categorized.as.downsian =
    ↪  share.non.ideologues.categorized.as.downsian))
}

sims.to.run <- expand.grid(
  n.issue.domains = c(1, 2, 5, 10, 20),
  n.issue.questions.per.domain = c(1, 2, 3, 5, 10)
) %>%
  slice(rep(1:n(), each = 25))

sim.results <- adply(sims.to.run, 1,
  function(row) sim1.robustness.sim(row$n.issue.domains,
    ↪  row$n.issue.questions.per.domain),
  .parallel = TRUE
)
```

The Figure below shows the results of the simulations.

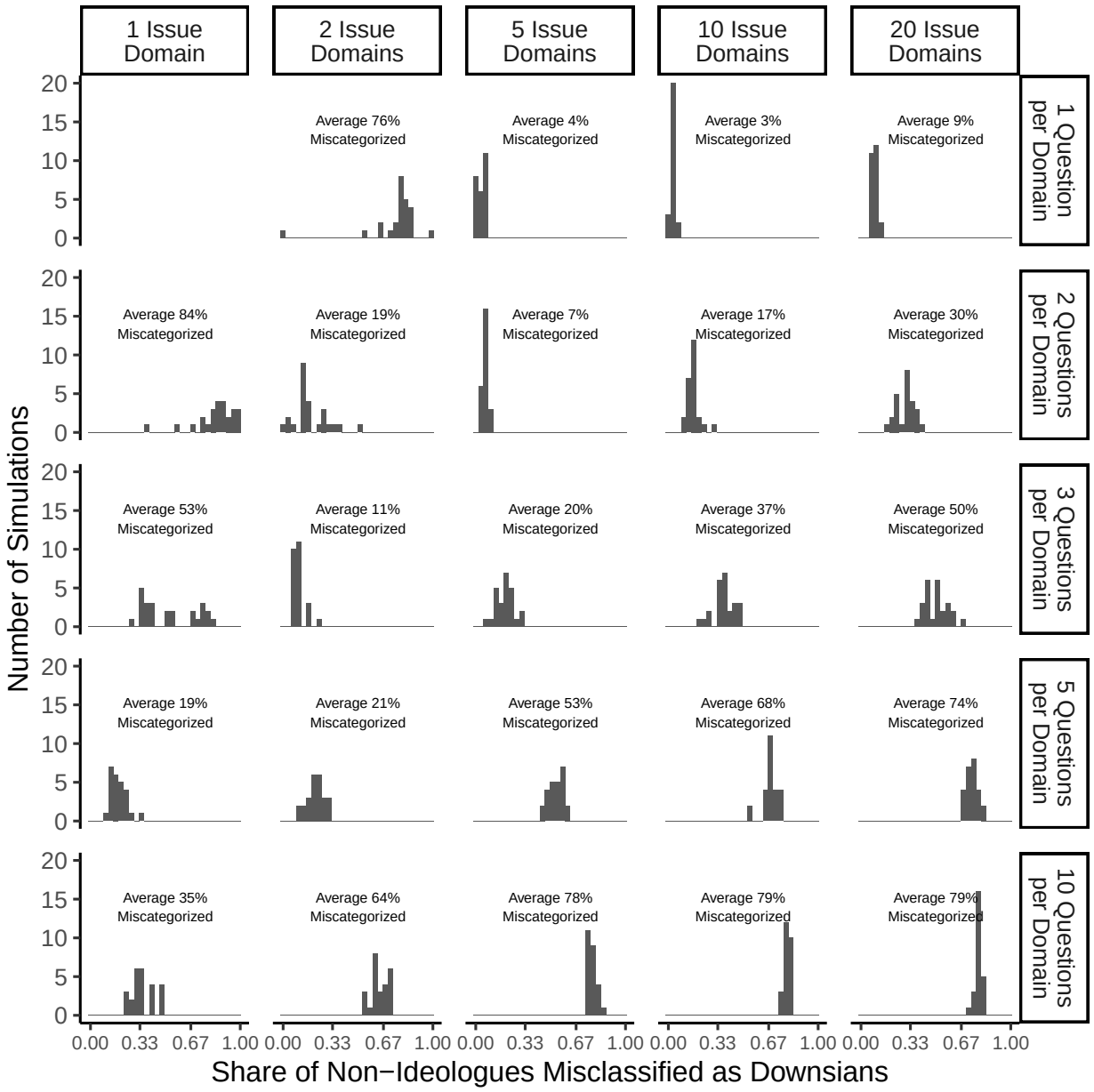
Recall that Simulation 1 is structured as follows. There are assumed to be a set number of latent issue domains on which citizens have latent preferences. There are then a set number of issue questions that tap their preferences on each of those domains (e.g., for Simulation 1, 15 issue domains with 5 questions tapping each of these domains). Further, there are three equally sized populations: pure ideologues, semi-ideologues, and pure non-ideologues. Pure ideologues rely only on one-dimensional ideology and not their issue-domain-specific preferences when responding to survey questions (or, equivalently, their issue-domain-specific preferences are entirely predicted by one-dimensional cross-issue ideology); semi-ideologues rely equally on one-dimensional ideology and their issue-domain-specific preferences when answering questions; and non-ideologues rely entirely on their issue-domain-specific preferences when answering questions and are not influenced at all by one-dimensional ideology.

A critical statistic is what proportion of true non-ideologues the Fowler et al. (2023) model misclassifies as ideologues (Downsians). In the Figure below, each column shows the results of simulations for a given value of the number of issue domains that are assumed to exist. Each row shows the results of simulations for a particular value of the number of issue questions within each domain. Finally, each cell shows a histogram with the results of 25 simulations under those hyperparameters—in particular, it shows the proportion of true non-ideologues that the Fowler et al. (2023) model misclassifies as ideologues in each of the 25 simulations. (The case with a value of 1 for both parameters is omitted because there is only one survey question in this case.) For example, the bottom-right panel shows this statistic in 25 simulations where there are assumed to be 20 issue domains, each assessed with 10 issue questions.

The results show that the Fowler et al. (2023) model misclassifies a substantial proportion of non-ideologues as ideologues under a broad range of parameter values. Generally speaking this bias is worse when the dataset is more multidimensional—i.e., when there are correlations between respondents' issue positions for reasons unrelated to one-dimensional ideology (in this simulation, the issue domains). For example, consider the last column, when there are twenty issue domains. When there is only one question per issue domain, the model performs well because there are no correlations between the questions for reasons unrelated to one-dimensional ideology. The non-ideologues are equivalent to a homogenous population that all have the same probability of saying yes to any given question, which is what the Fowler et al. (2023) model assumes is the case for Conversians. However, as the number of issue questions per domain increases (moving down the rows along the last column), the share of non-ideologues miscategorized as Downsians increases quickly, as there are now correlations between the questions for reasons unrelated to one-dimensional ideology. But the Fowler et al. (2023) model does not allow for such correlations to exist, and so assumes that to

the extent such correlations do exist they must exist due to one-dimensional ideology, leading non-ideologues to be misclassified.

Figure OA5: Robustness of Simulation 1 Results

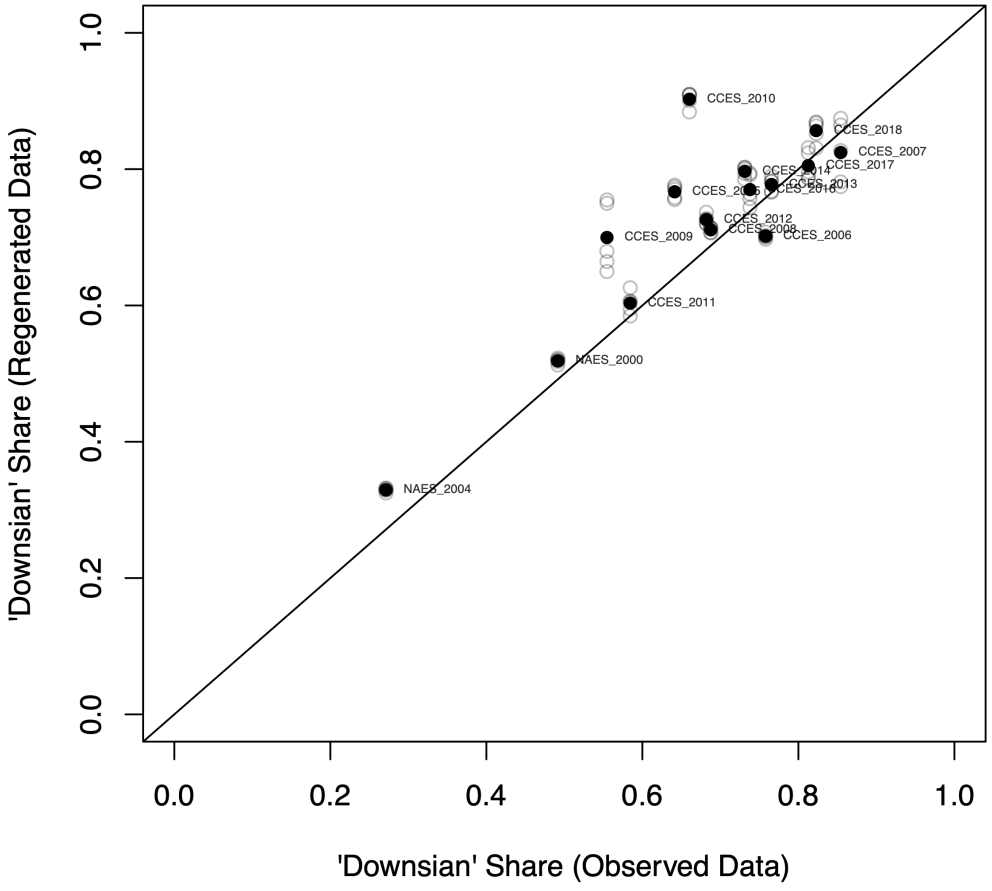


C Regeneration

We next further demonstrate that Fowler et al.'s (2023) model does not capture what it intends to in data that more closely approximates the data from the original study. To do so, we "regenerate" the data that Fowler et al. (2023) themselves analyze, holding constant the number of respondents, issues, and pattern of missing data exactly, as well as the average level of support for each issue and the pairwise correlations between issue positions probabilistically. We do this by estimating a latent multivariate normal model for their binary issue position data, which then enables us to generate new datasets with the same pairwise issue correlation structure as the observed data. Crucially, all respondents are now of a single type: their latent responses are drawn from a common multivariate normal model rather than having Downsian/Conversian types, and this distribution is not defined by a one-dimensional ideology. There are no "Conversians" or "Downsians" at all in this data generating process. Every respondent has views that are correlated with their other views according to a common model that applies to everyone.

We run this regeneration procedure five times on each of the datasets Fowler et al. (2023) analyze, and each time apply their model to the results. Figure OA6 plots the share of respondents who are categorized as "Downsian" in each of the regenerated datasets (y-axis), and compares the results to the the share categorized as "Downsians" in the original data (x-axis). Fowler et al.'s (2023) estimator categorizes nearly the same proportion of respondents as ideologues ("Downsians") in the regenerated datasets as when applied to each original dataset.

Figure OA6: Results When Regenerating Fowler et al. (2023)'s Data



This analysis shows that we can replicate the estimated proportions of ideologues ("Downsians") from datasets generated *without* different categories of respondents. This implies that the proportion of ideologues ("Downsians") that the authors estimated in their original datasets cannot reliably indicate how many respondents truly belong to this category. Instead of depending on the proportion of ideologues and non-ideologues in the population, the proportion of each type

their model estimates instead appears to depend almost entirely on the pairwise correlation structure of issue positions in the dataset, which was preserved in these simulations.

The below code shows how we created Figure OA6.

```
###
### Function to generate alternative roll call matrix
###

# matching on
#     + pattern of missingness
#     + pairwise item correlations
#     + item response frequencies
# but using latent multivariate normal response model

max_det_cor_impute <- function(Sigma) {
  n_vars <- ncol(Sigma)

  # identify the number of missing elements of Sigma
  n_missing <- sum(is.na(Sigma)) / 2

  # calculate negative determinant, based on filled in imputes
  detf <- function(imputes) {
    Sigma_Imputed <- Sigma
    i <- 1
    for (j in 1:(n_vars - 1)) {
      for (jj in (j + 1):n_vars) {
        if (is.na(Sigma[j, jj])) {
          Sigma_Imputed[j, jj] <- Sigma_Imputed[jj, j] <- imputes[i]
          i <- i + 1
        }
      }
    }
    return(-det(Sigma_Imputed))
  }

  # use optim to find maximum determinant solution
  start_tmp <- mean(as.vector(Sigma)[as.vector(Sigma) != 1], na.rm = TRUE)
  start_values <- rep(start_tmp, n_missing)
  optim_out <- optim(start_values, detf,
    method = "L-BFGS-B",
    lower = -1, upper = 1
  )

  # populate Sigma_MD with Max Det solution

  Sigma_MD <- Sigma
  i <- 1
  for (j in 1:(n_vars - 1)) {
    for (jj in (j + 1):n_vars) {
      if (is.na(Sigma[j, jj])) {
        Sigma_MD[j, jj] <- Sigma_MD[jj, j] <- optim_out$par[i]
        i <- i + 1
      }
    }
```

```
  }

  # if not positive definite, truncate negative eigenvalues and rescale to valid
   ↪  correlation matrix

  n <- dim(var(Sigma_MD))[1L]
  E <- eigen(Sigma_MD)
  Sigma_CM1 <- E$vectors %*% tcrossprod(diag(pmax(E$values, 0), n), E$vectors)
  Balance <- diag(1 / sqrt(diag(Sigma_CM1)))
  Sigma_CM2 <- Balance %*% Sigma_CM1 %*% Balance

  # return valid correlation matrix

  return(Sigma_CM2)
}

polychor_pairwise <- function(mat) {
  # calculate observed polychoric correlations for items using observed data
  n_vars <- ncol(mat)
  Sigma <- matrix(NA, n_vars, n_vars)
  # populate diagonal
  for (j in 1:n_vars) Sigma[j, j] <- 1
  # calculate pairwise polychoric correlations
  for (j in 1:(n_vars - 1)) {
    for (jj in (j + 1):n_vars) {
      if (j != jj) {
        # if pair of items appears, set correlation to observed
        tmp <- suppressWarnings(polychor(mat[, j], mat[, jj]))
        if (!is.na(tmp)) Sigma[j, jj] <- Sigma[jj, j] <- tmp
      }
    }
  }

  return(Sigma)
}

mvn_rc_generator <- function(mat) {
  n_vars <- ncol(mat)

  # calculate observed polychoric correlations for observed pairs
  Sigma <- polychor_pairwise(mat)

  # impute unobserved pairwise correlations to maximise correlation matrix determinant
  # see https://royalsocietypublishing.org/doi/full/10.1098/rsos.172348#FN2R for
   ↪  justification
  Sigma_imputed <- max_det_cor_impute(Sigma)

  # generate multivariate normal random latent votes
  y_star <- mvrnorm(nrow(mat), rep(0, n_vars), Sigma_imputed)

  # dichotomise latent votes into observed votes
  y <- matrix(NA, nrow(mat), ncol(mat))
  for (j in 1:n_vars) y[, j] <- y_star[, j] > qnorm(prop.table(table(mat[, j]))[1])
```

```r
  # match missingness of original data set
  mat_alt <- y * (mat >= 0)

  # return new roll-call matrix
  return(mat_alt)
}



###
### load all large survey data
###

# load("bigsurveys_recoded4.RData")
data <- read.dta("bigsurveys_recoded4.dta")

policy_ind <- 3:329
years <- unique(data$source)

table(years)

###
### Remove modules from datasets
###

for (year in years) {
  tmp <- data[, policy_ind]
  tmp <- tmp[data$source == year, ]
  allna <- is.na(colMeans(tmp, na.rm = T))
  tmp <- tmp[, !allna]

  # we need fewer responses than this
  # to determine that an item was asked
  # on a module
  threshold <- 5000

  module_items <- c()
  for (i in 1:dim(tmp)[2]) {
    vec <- ifelse(tmp[, i] == 0, 1, tmp[, i])
    if (sum(vec, na.rm = T) < threshold) {
      module_items <- c(module_items, i)
    }
  }
  if (length(module_items) > 1) {
    module_people <- which(!is.na(rowMeans(tmp[, module_items], na.rm = T)))
  } else {
    module_people <- c()
  }


  print(year)
  print(c("Total people, items:", dim(tmp)))
```

```
  print(c("Module people:", length(module_people)))
  print(c("Module items:", length(module_items)))

  # now that module items have been identified,
  # NA those items for those modules
  if (length(module_items) > 0) {
    ind <- which(names(data) %in% names(tmp)[module_items])
    data[data$source == year, ind] <- NA
  }
}


###
### run em_mix_irt on every included data set, and append estimates
###

# store all of the results in a list
reslist <- list()
reslist_alt <- list()
# store ideal points
data$x <- rep(NA, dim(data)[1])
# store probability of each voter type
data$w1 <- rep(NA, dim(data)[1])
data$w2 <- rep(NA, dim(data)[1])
data$w3 <- rep(NA, dim(data)[1])
# store individual log likelihoods
data$lk1 <- rep(NA, dim(data)[1])
data$lk2 <- rep(NA, dim(data)[1])


for (i in 1:length(years)) {
  year <- years[i]

  print(year)
  mat <- apply(data[data$source == year, policy_ind], 2, as.numeric)
  exclude <- !colnames(mat) %in% c("cces2012_immigration5", "cces2012_immigration6")
  notna <- which(!is.na(colMeans(mat, na.rm = T)) & exclude)
  print(length(notna))

  mat <- mat[, notna]
  res <- em_mix_irt(mat, iter = 100)
  reslist[[year]] <- res
  data$x[data$source == year] <- res$irt$x
  data$w1[data$source == year] <- res$w[, 1]
  data$w2[data$source == year] <- res$w[, 2]
  data$w3[data$source == year] <- res$w[, 3]
  data$lk1[data$source == year] <- res$irt$lk
  data$lk2[data$source == year] <- res$ivp$lk

  # analysis on re-generated roll-call matrix

  reslist_alt[[year]] <- list()
  reslist_alt[[year]]$observed_pairwise_polychor <- polychor_pairwise(mat)
```

```
  reslist_alt[[year]]$replicate_weight_estimates <- list()
  for (r in 1:5) {
    mat_alt <- mvn_rc_generator(mat)
    res_alt <- em_mix_irt(mat_alt, iter = 100)
    reslist_alt[[year]]$replicate_weight_estimates[[r]] <- colMeans(res_alt$w)
  }
}

###
### Identify the direction using pid7
###

data$pid7[data$pid7 == "__NA__"] <- NA
data$pid7 <- as.numeric(data$pid7)
sources <- unique(data$source)

for (i in sources) {
  ind <- data$source == i
  direction <- sign(cor(data$pid7[ind], data$x[ind], use = "pairwise.complete.obs"))
  data$x[ind] <- direction * data$x[ind]
}

for (i in sources) {
  ind <- data$source == i
  print(i)
  print(cor(data$pid7[ind], data$x[ind], use = "pairwise.complete.obs"))
}

###
### Save the results
###

save(reslist, reslist_alt, file = "all_results4_alt.RData")

save(data, file = "bigsurveys_recoded_plus_estimates4.RData")

write.dta(data, file = "bigsurveys_recoded4.dta")



###
### Comparison of Estimates on observed vs regenerated Data
###

if (FALSE) {
  average_weight_observed <- matrix(NA, length(reslist), 3)
  average_weight_regenerated <- matrix(NA, length(reslist), 3)

  for (i in 1:length(reslist)) {
    average_weight_observed[i, ] <- colMeans(reslist[[i]]$w)
    average_weight_regenerated[i, ] <- colMeans(reslist_alt[[i]]$w)
  }
```

```
  plot(average_weight_observed[, 1], average_weight_regenerated[, 1],
    xlim = c(0, 1), ylim = c(0, 1), xlab = "'Downsian' Share (Observed Data)", ylab =
     ↪  "'Downsian' Share (Regenerated Data)"
  )
  text(average_weight_observed[, 1], average_weight_regenerated[, 1], names(reslist), pos
   ↪  = 4, cex = 0.4, col = rgb(0, 0, 0, 0.6))
  abline(0, 1)
}
```

References

Converse, Philip E. 1964. The nature of belief systems in mass publics. In *Ideology and its discontents*, ed. David Apter. New York: Glencoe Free Press.

Fowler, Anthony, Seth J Hill, Jeffrey B Lewis, Chris Tausanovitch, Lynn Vavreck and Christopher Warshaw. 2023. "Moderates." *American Political Science Review* 117(2):643–660.