Advanced A/B Testing

Large Sample A/B Tests

Elea McDonnell Feit

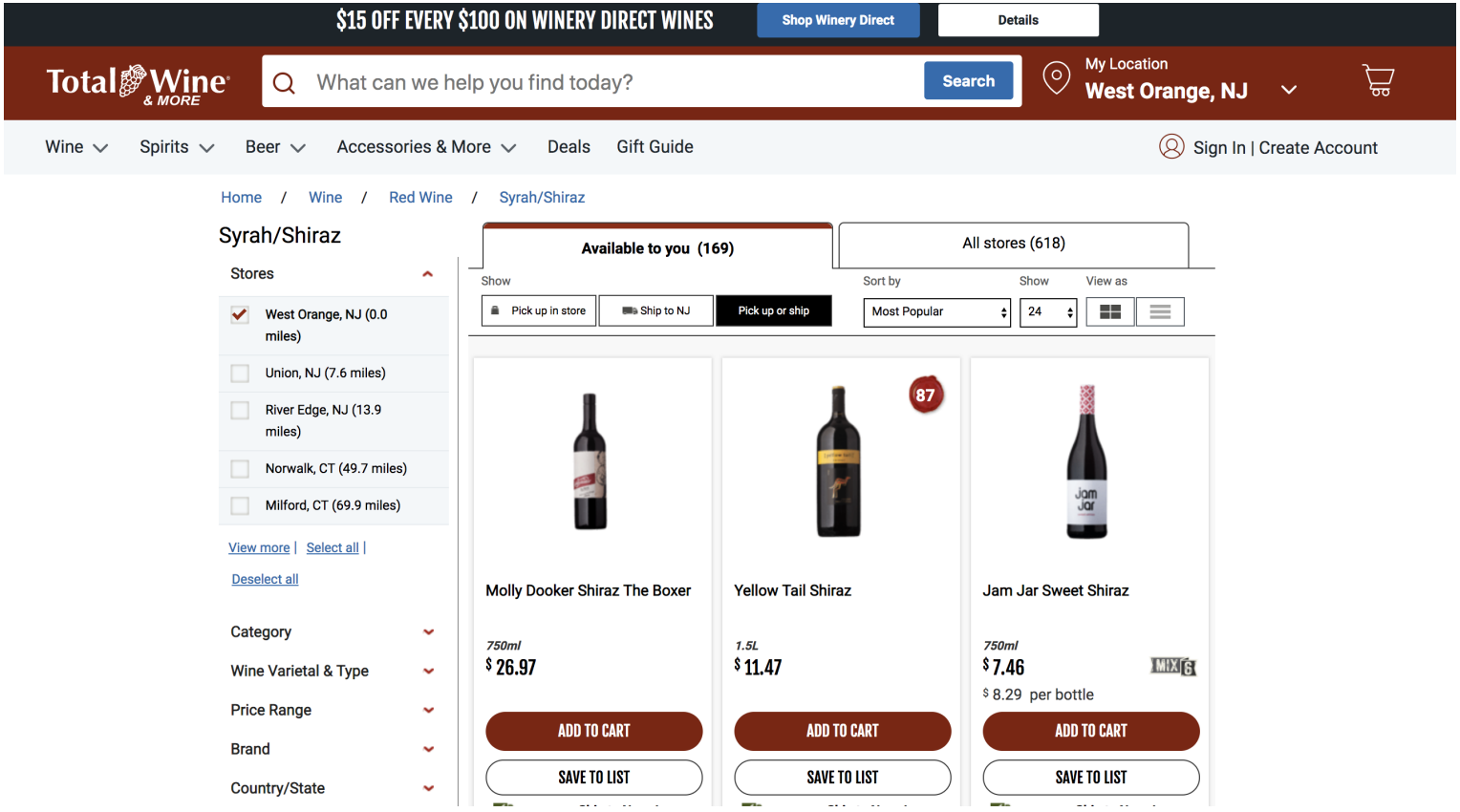
6/16/2019

"When customers are randomly assigned to treatment and control groups, and there are many customers in each group, then you may effectively have multiple experiments to analyze."

- Anderson and Simester (2011) A step-by-step guide to smart business experiments, *HBR*

Slicing and dicing

Wine retailer experiment



Wine retailer experiment

Test setting: email to retailer mailing list

Unit: email address

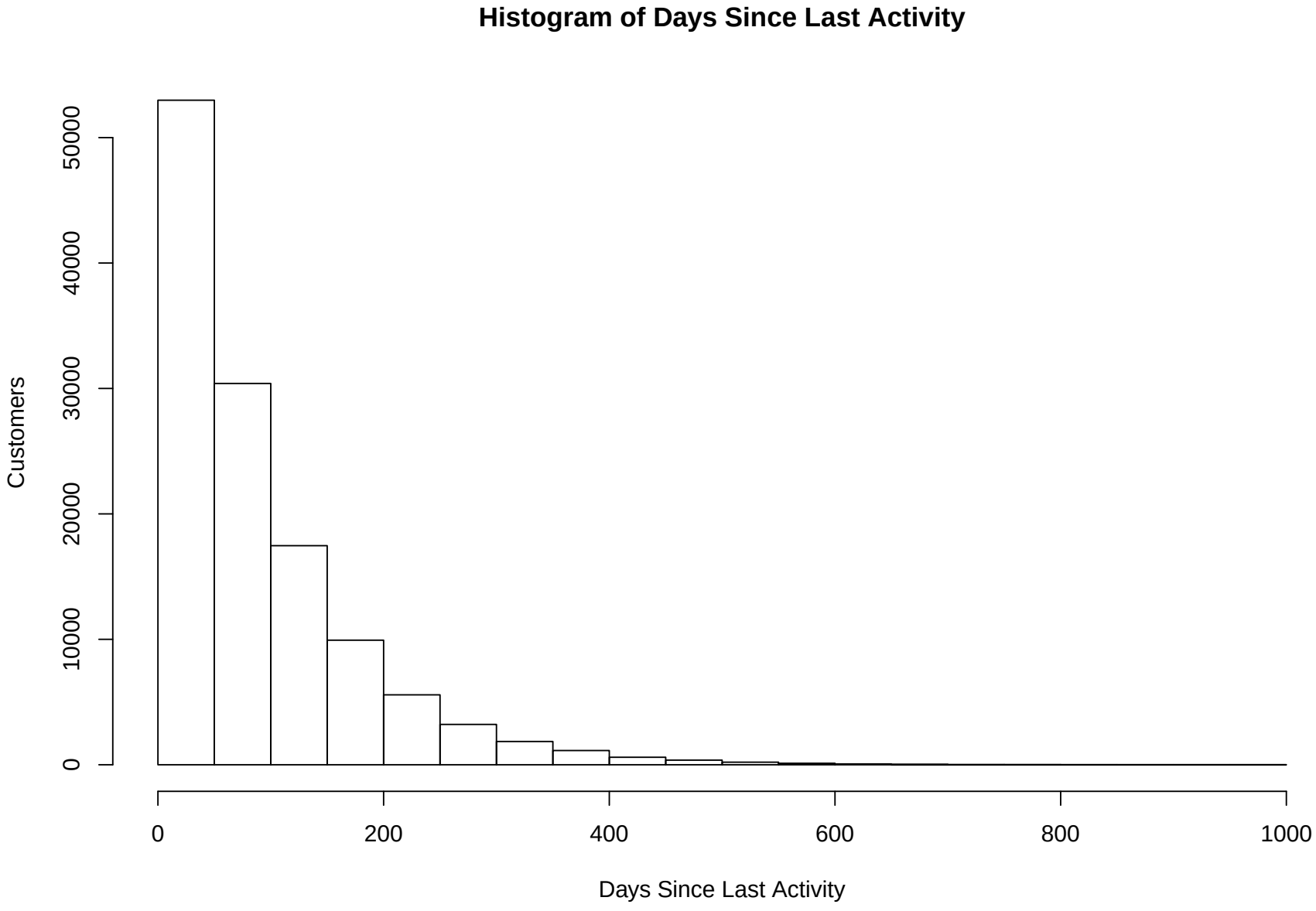
Treatments: email version A, email version B, holdout

Reponse: open, click and 1-month purchase ($)

Selection: all active customers

Assignment: randomly assigned (1/3 each)

Baseline variable: days since last activity



Experiments within experiments

Consider the customers who have been active in the last 60 days.

Within that subset, customers were randomly assigned to recieve email A, email B or no email.

So, we can analyze the data for a subgroup as it's own test test by slicing down and then re-analyzing.

However, we will only find signifcant results if we have enough sample in the subgroup.

Recent active versus aged customers

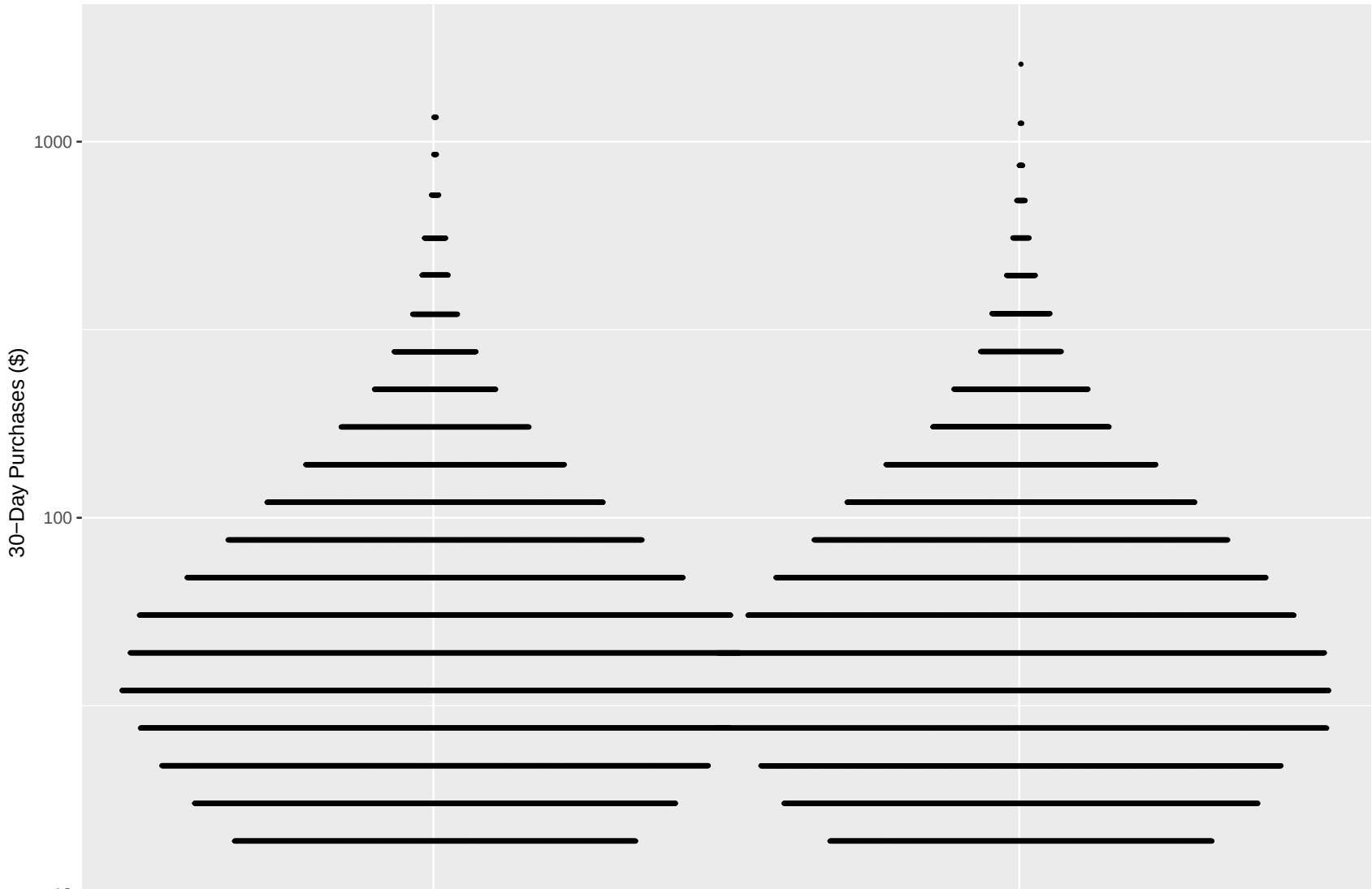
```
d %>% group_by((days_since < 60), group) %>% summarize(mean
```

```
## # A tibble: 6 x 5
## # Groups:   (days_since < 60) [2]
##   `(days_since < 60)` group   `mean(open)` `mean(click)`
##   <lgl>               <fct>          <dbl>          <dbl>
## 1 FALSE               ctrl           0              0
## 2 FALSE               email_A        0.582          0.106
## 3 FALSE               email_B        0.503          0.0715
## 4 TRUE                ctrl           0              0
## 5 TRUE                email_A        0.865          0.160
## 6 TRUE                email_B        0.812          0.117
```

- The email seems to produce a stronger effect on purchases for recently active customers.

Is email effective for recently actice?

```
d %>% filter(email==TRUE) %>% ggplot(aes(y=purch, x=group))
  geom_dotplot(binaxis='y', stackdir='center', stackratio=(
  ylab("30-Day Purchases ($)") + xlab("") + scale_y_log10()
```



Significance test: recently active

```
t.test(purch ~ email, data=d[d$days_since < 60,])
```

```
##
##  Welch Two Sample t-test
##
## data:  purch by email
## t = -33.51, df = 50513, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
##  -17.5776 -15.6350
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            18.48809             35.09439
```

Significance test: aged customers

```
t.test(purch ~ email, data=d[d$days_since > 60,])
```

```
##
##  Welch Two Sample t-test
##
## data:  purch by email
## t = -30.257, df = 56220, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
##  -10.752048  -9.443798
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            6.792411           16.890335
```

Every A/B test can be sliced

For example, we can look at the effect of the treatment separately for the green apples and the red apples.

Slicing is based on baseline variables

Anyone who keeps historic data on customers or visitors has lots of baseline variables available for slicing and dicing:

- data on previous website visits
- sign-ups
- geographic location
- source
- past purchase (by category)
- recency
- frequency

Exercise

Re-analyze the opens, clicks and purchases for people who have bought syrah in the past.

```
summary(d$syrah > 0)
```

```
##    Mode   FALSE    TRUE
## logical   88359   35629
```

```
mean(d$syrah > 0)
```

```
## [1] 0.2873585
```

Repeated significance testing

Slicing and dicing means you will run many significance tests.

You may remember from intro stats that 1 in 20 significance tests at 95% confidence will be significant, when there is no effect. You will get false positives, especially when slicing and dicing.

When you think you've found a golden ticket, re-test before betting the company.

Slicing and dicing: Summary

Slicing and dicing will reveal two things about subgroups of customers.

1. Subgroups will vary in how much they engage in behaviors
   - Recently active tend to have higher average purchases after the email
2. Subgroups vary in how they respond to treatments
   - Recently active are more affected by the email

Heterogeneous treatment effects

"Experiments are used because they provide credible estimates of the effect of an intervention for a sample population. But underlying this average effect for a sample may be **substantial variation in how particular respondents respond to treatments**: there may be **heterogeneous treatment effects**."

– Athey and Imbens, 2015

Heterogeneous treatment effects and targeting

Marketers should be interested in heterogeneous treatment effects when there is opportunity to apply different treatments to each subgroup (ie targeting).

email $\rightarrow$ high potential for targeting

website $\rightarrow$ less potential for targeting

Analyzing experiments with regression (pre-req)

Analyzing experiments with regression

We use a **regression model** to define a relationship between the response (y) and the treatment (x).

$y = a + b \times x + \varepsilon$

The model literally says that we get the average response by multiplying the treatment indicator x by b and adding that to a. When we fit a model, we use data to estimate a and b.

R formulas

In R, we shorthand the model equation with an R formula:

`purch ~ email`

This means exactly the same thing as:

$\texttt{purch} = a + b \times \texttt{email} + \varepsilon$

where we estimate a and b from data.

Analyzing an experiment with regression

```
m1 <- lm(purch ~ email, data=d)
summary(m1)
```

```
##
## Call:
## lm(formula = purch ~ email, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -25.74  -25.74  -12.42   -1.23 1581.66
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.4203     0.2679   46.36   <2e-16 ***
## emailTRUE    13.3243     0.3281   40.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
##
```

Regression versus significance test

Regression model

```
summary(m1)$coef
```

```
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 12.42029  0.2679092 46.36005        0
## emailTRUE   13.32428  0.3281218 40.60772        0
```

Significance test

```
t.test(purch ~ email, data=d, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  purch by email
## t = -40.608, df = 123986, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
##  -13.96739 -12.68116
```

Regression versus significance tests

If you like regression, you can use regression to analyze all your tests.

If you don't like regression, you should try it because it gives you the ability to pull in baseline variables.

Model with a baseline variable

```
m2 <- lm(purch ~ email + (days_since < 60), data=d)
summary(m2)$coef
```

```
##                          Estimate Std. Error  t value     Pr
## (Intercept)              4.764104  0.3031659 15.71451 1.3616
## emailTRUE               13.301929  0.3246010 40.97932 0.0000
## days_since < 60TRUE 15.929019  0.3062459 52.01382 0.0000
```

Aged customers in the control group purchased on average $5.55 in the 30-days after the email was sent. Recently active customers in the control group purchased an additional $13.55. The average effect of the email was $6.44.

Controlling for baseline variables increases the likelihood of finding significant effects. This is sometimes called "regression correction."

Incorporating heterogeneous treatment effects

To incorporate heterogeneous treatment effects, we need an **interaction** between the treatment effect (x) and a baseline variable (z).

When we interact to terms, we are defining a model that multiplies the two terms:

$y = a + bx + cz + d(xz) + \varepsilon$

The R formula for this model is:

```
purch ~ email + (days_since < 60)  +
email:(days_since < 60)
```

or equivalently

```
purch ~ email*(days_since < 60)
```

Incorporating heterogeneous treatment effects

```
m3 <- lm(purch ~ email + (days_since < 60) + email:(days_si
summary(m3)$coef
```

```
##                                        Estimate Std. Error    t v
## (Intercept)                            6.804775  0.3676230 18.51
## emailTRUE                             10.238134  0.4504495 22.72
## days_since < 60TRUE                   11.683315  0.5302628 22.03
## emailTRUE:days_since < 60TRUE          6.368163  0.6494166  9.80
```

The email effect is $5.36 for aged customers plus an additional $2.23 recent customers (total of $7.59).

Uplift modeling (finally!)

Uplift model for purchase amount

An **uplift model** is a regression model that incorporates many baseline variables. For example:

```
m4 <- lm(purch ~ email*(days_since < 60) + email*(past_purc
         data=d)
summary(m4)$coef
```

```
##                                          Estimate Std. Error    t
## (Intercept)                            -0.685228  0.7058724 -0.97
## emailTRUE                              -1.978801  0.8626010 -2.29
## days_since < 60TRUE                    11.743406  0.5236114 22.42
## past_purch > 50TRUE                     8.759740  0.5406815 16.20
## visits > 3TRUE                          2.653066  0.6819410  3.89
## emailTRUE:days_since < 60TRUE           6.321222  0.6412638  9.85
## emailTRUE:past_purch > 50TRUE           7.795652  0.6624463 11.76
## emailTRUE:visits > 3TRUE                9.260021  0.8351709 11.08
```

Scoring customers with an uplift model

If you have someone who wasn't in the test, but you know their baseline variables, you can use an uplift model to predict likely treatment effect.

```
new_cust <- data.frame(past_purch=rep(38.12,2), days_since=
(pred <- predict(m4, cbind(email=c(TRUE, FALSE), new_cust))
```

```
##        1        2
## 15.40060 11.05818
```

```
(lift <- pred[1] - pred[2])
```

```
##        1
## 4.342422
```

This new customer is predicted to buy $13.03 if they get an email or $12.40 without, for a uplift of $0.63.

Scoring for another (better) customer

```
new_cust <- data.frame(past_purch=rep(127.88,2), days_since
(pred <- predict(m4, cbind(email=c(TRUE, FALSE), new_cust))
```

```
##        1        2
## 43.86908 22.47098
```

```
(lift <- pred[1] - pred[2])
```

```
##        1
## 21.39809
```

This is a better target with an uplift of 11.61.

Uplift models and targeting

For costly treatments (eg catalogs, discounts) we should target customers customers that we predict will have a positive effect that exceeds costs.

Persuadables and do-not-disturb

Response if Treated			
	N	**Do-Not-Disturb** ***c***	**Lost Cause** ***d***
	Y	Sure Thing ***b***	Persuadable ***a***
		Y	N
		Response if <u>not</u> treated	

Source: Predictive Analytics Times

Uplift model for clicks

We can also build an uplift model for click probability, but we should use a logistic regression for binary outcomes.

```
m5 <- glm(click ~ group*(days_since < 60) + group*(past_pu
                  group*(syrah > 0) + group*(cab > 0) +
                  group*(sav_blanc > 0) + group*(chard > 0
           family = binomial,
          data=d[d$group != "ctrl",])
```

Uplift model for clicks

While email B has lower overall click rate, customers who have purchased syrah in the past are more likely to click if they get email B (which promoted syrah).

```
summary(m5)$coef
```

```
##                                                Estimate Std. Error
## (Intercept)                                -2.48230897 0.04332267
## groupemail_B                               -0.67064544 0.06848652
## days_since < 60TRUE                         0.48161494 0.02956872
## past_purch > 50TRUE                         0.41923694 0.04474487
## visits > 3TRUE                              0.01528802 0.03920836
## syrah > 0TRUE                               0.04624640 0.03409196
## cab > 0TRUE                                -0.01426477 0.03400629
## sav_blanc > 0TRUE                           0.04214018 0.03699615
## chard > 0TRUE                               0.11199369 0.03684271
## groupemail_B:days_since < 60TRUE            0.06953462 0.04562008
## groupemail_B:past_purch > 50TRUE           -0.07434038 0.06864675
## groupemail_B:visits > 3TRUE                 0.01599524 0.06104585
```

More baseline variables (features)

Uplift models can include many, many baseline variables. Creating these variables from source data (CRM, web analytics data, etc) is called **feature engineering**.

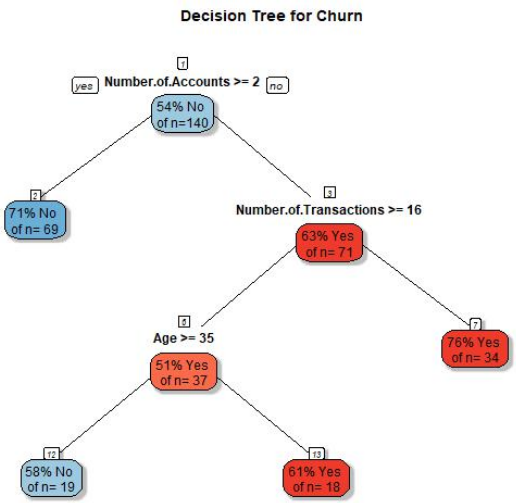
Causal forests

Causal forests

Causal forests are an alternative to regression for identifying heterogeneous treatment effects and scoring customers based on predicted treatment effect uplift.

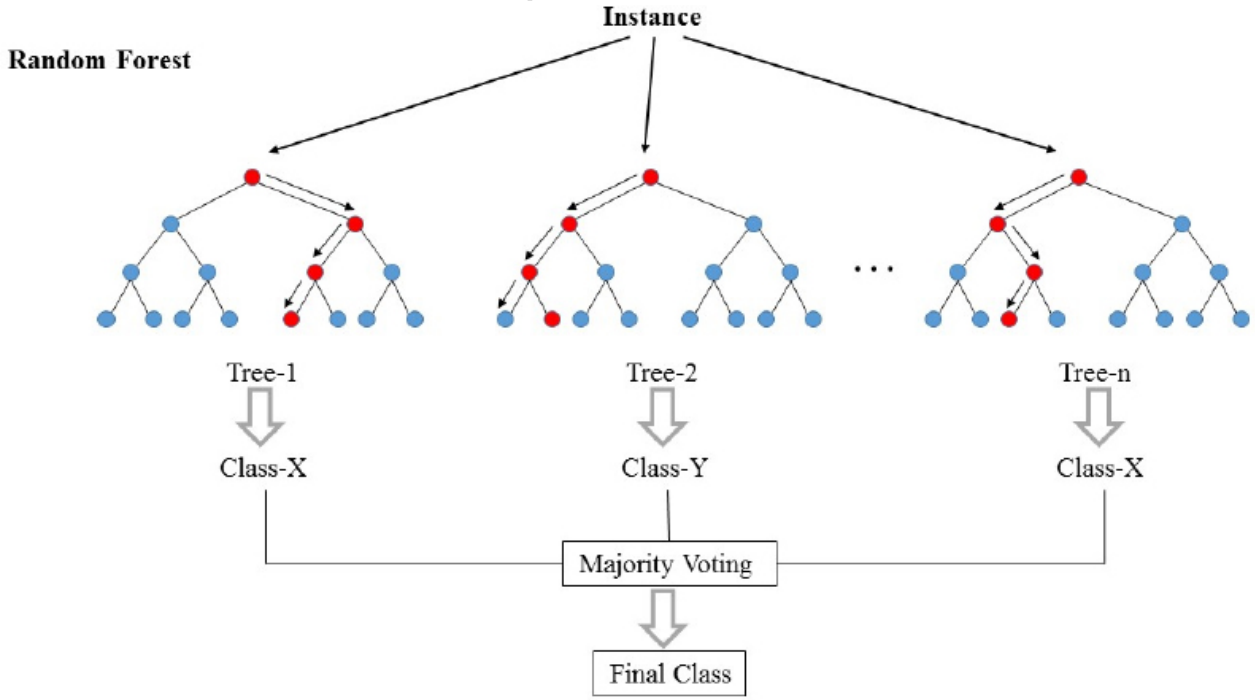
Preliminaries I: CART

Where regression models predict customer outcomes with a linear equation, cart trees predict customer outcomes using a tree structure. CARTs are estimated by finding the tree structure that seems to classify people correctly most of the time.



Preliminaries II: Random forests

Random forests are collections of different CARTs each fit to a subset of the data. Each tree in the forest classifies customers slightly differently. Unlike a regression, a random forest can pick up non-linear relationships.



Causal forests

Causal forests are random forests designed to categorize customers according to their **treatment effect** in an experiment. The customers in each leaf are assumed to have homogeneous treatment effects, with heterogeneous treatment effects between leaves.

Advantages
- Works well with a large number of baseline variables
- Doesn't require the analyst to define cut-offs for continuous baseline variables
- Will fit non-linear relationships between baseline variables and uplift

Causal forest for wine retailer experiment

```
treat <- d$email
response <- d$purch
baseline <- d[, c("days_since", "past_purch", "visits", "ch
cf <- causal_forest(baseline, response, treat)
print(cf)
```

```
## GRF forest object of type causal_forest
## Number of trees: 2000
## Number of training samples: 123988
## Variable importance:
##     1     2     3     4     5     6     7
## 0.164 0.667 0.088 0.021 0.040 0.012 0.009
```

Overall average treatment effect

```
average_treatment_effect(cf, method="AIPW")
```

```
##    estimate     std.err
## 13.3102571  0.2871159
```

This is similar to the estimate from our simple regression which was 6.42 (0.30).

Predicted uplift

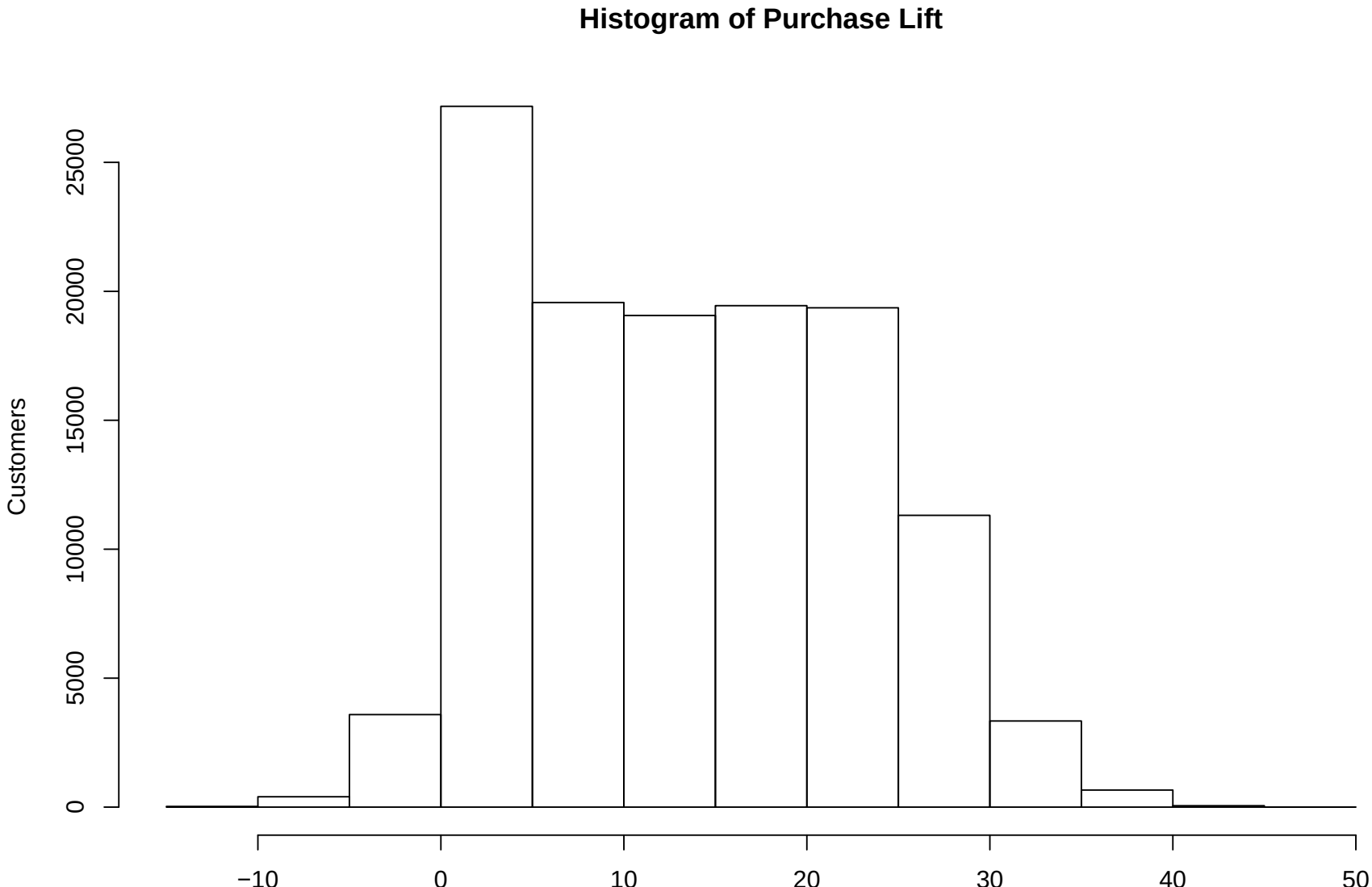
Just like any uplift model, we can use the model to predict the email effect for new customers.

```
new_cust <- data.frame(chard=38.12, sav_blanc=0, syrah=0, c
                       past_purch=38.12, days_since=19, vis
predict(cf, new_cust, estimate.variance = TRUE)
```

```
##   predictions variance.estimates
## 1 -0.01089267           12.74849
```

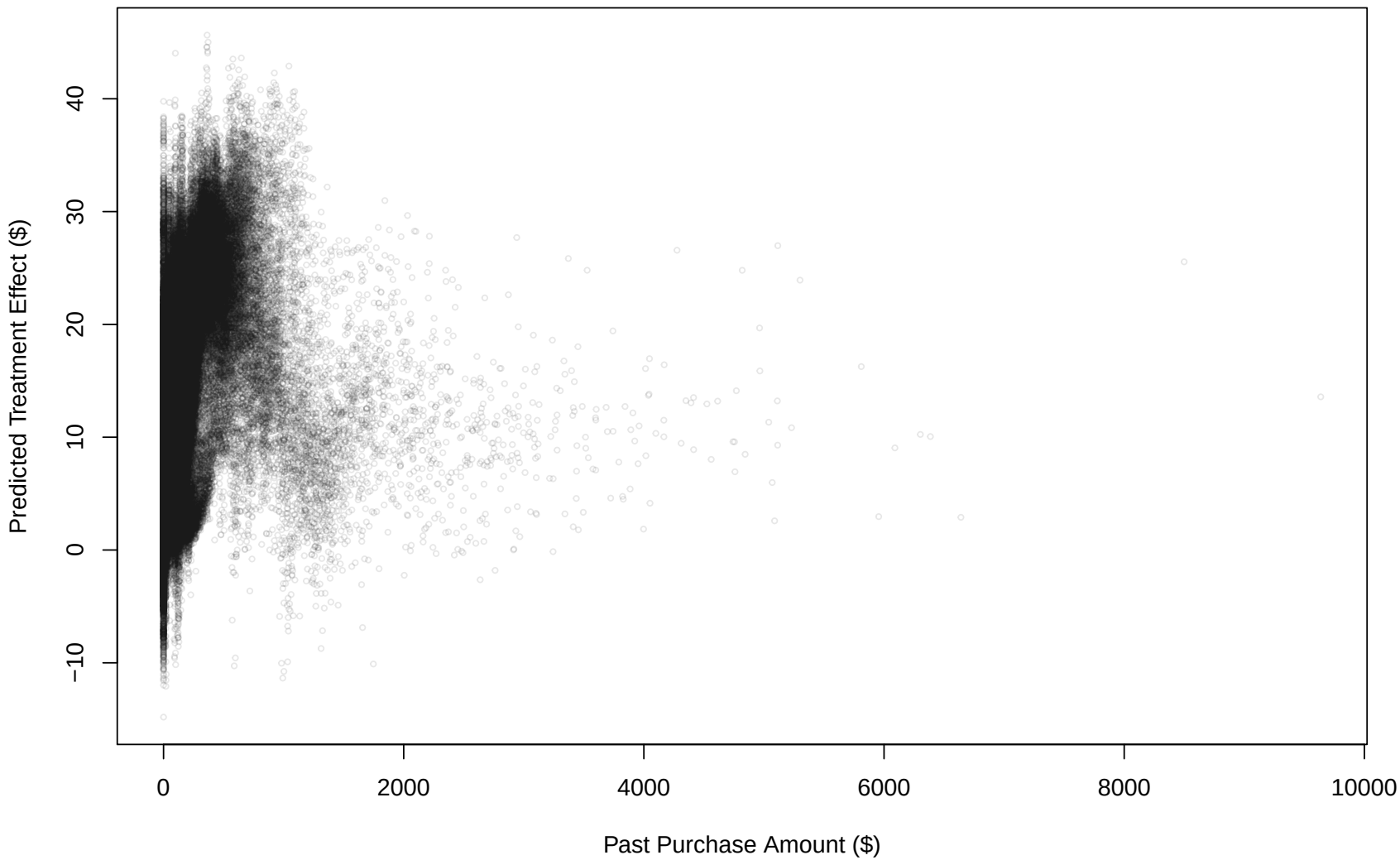
Predicted uplift for all customers in test

```
hist(predict(cf)$predictions,
     main="Histogram of Purchase Lift",
     xlab="Purchase Lift for Email", ylab="Customers")
```



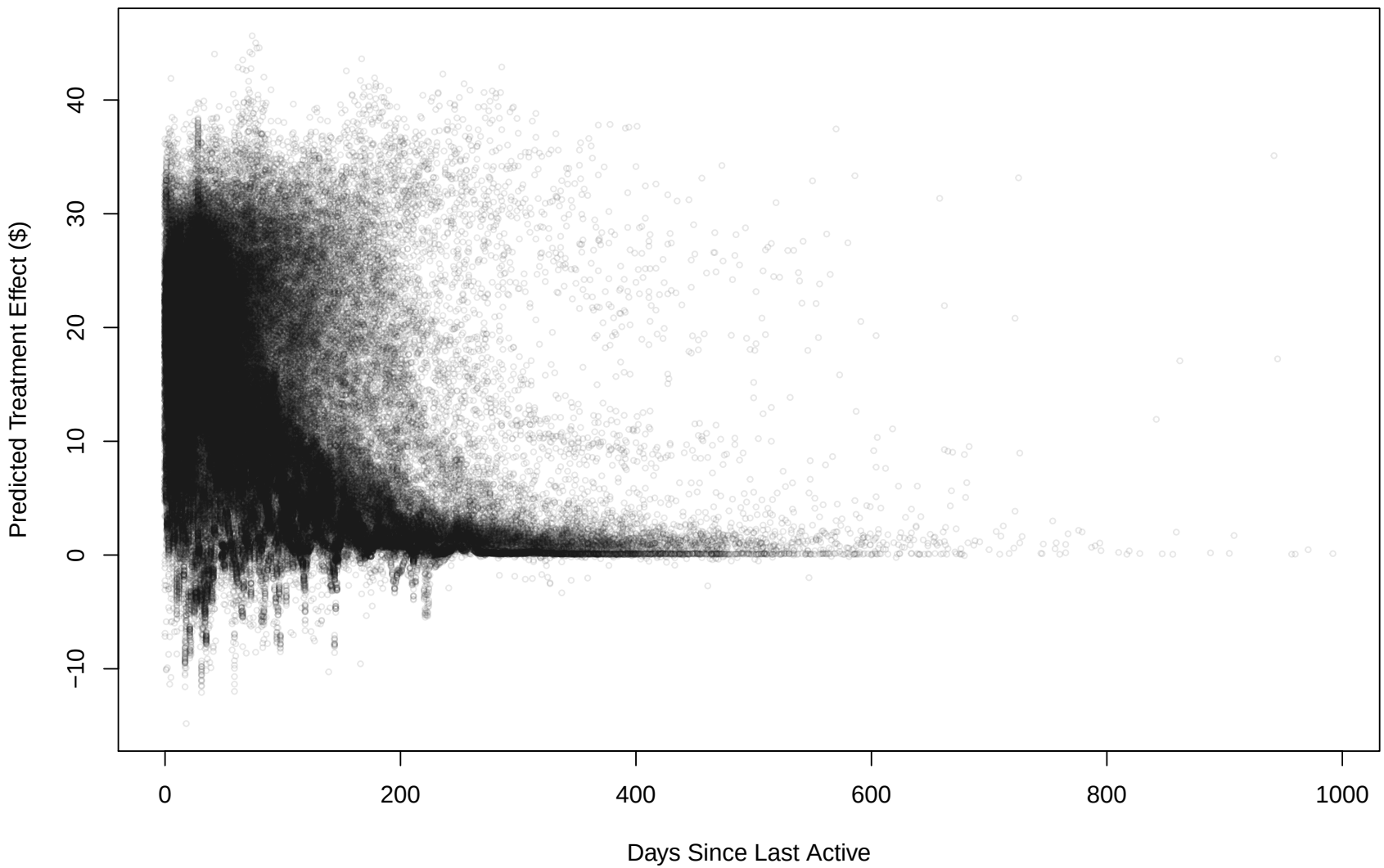
Predicted uplift versus past purchase amount

```
trans_gray <- rgb(0.1, 0.1, 0.1, alpha=0.1)
plot(d$past_purch, predict(cf)$predictions, cex=0.5, col=tr
     xlab="Past Purchase Amount ($)", ylab="Predicted Treat
```



Uplift versus days since last active

```
trans_gray <- rgb(0.1, 0.1, 0.1, alpha=0.1)
plot(d$days_since, predict(cf)$predictions, cex=0.5, col=tr
     xlab="Days Since Last Active", ylab="Predicted Treatme
```



Things you just learned

- Large sample $\rightarrow$ look for heterogeneous treatment effects using baseline variables
- Three ways to find heterogeneous treatment effects
  - Slicing and dicing: filter down then analyze sub-test
  - Uplift modeling
    - Build a regression with interactions between x and z's
    - Use logistic regression for binary response
  - Causal forests