Regressions: Linear, Logit, Deviance

Lecture 3

Connor Dowd

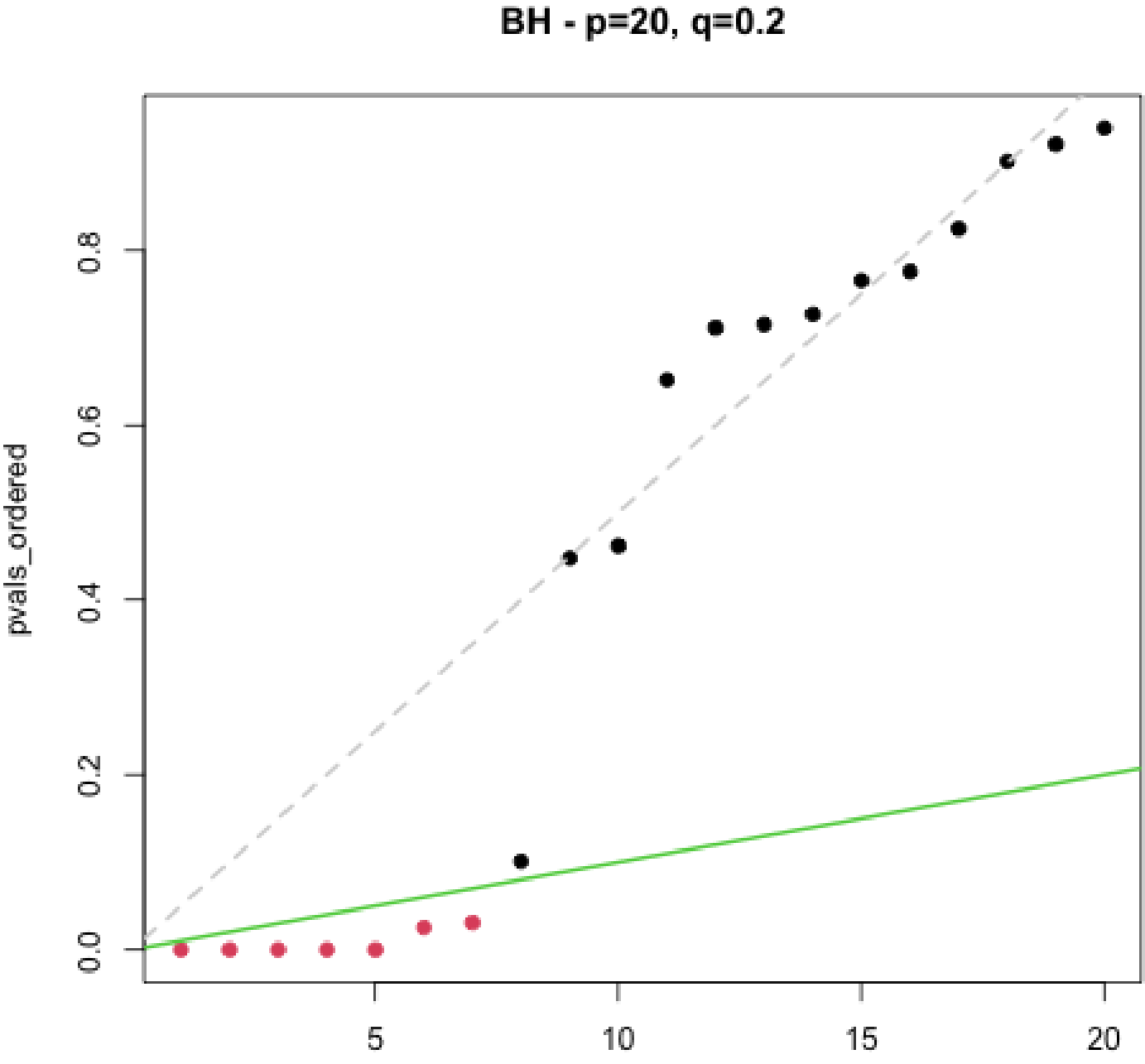
April 6th, 2021

Today's Class | Likely to spillover to Thursday

1. Quick Review
   - FDR
   - Loss
2. Regression Basics
   - Single redux
   - Multivariate
   - Interactions
   - Factors
3. Logistic Regression
4. Deviance
   - Out-of-sample

Quick Review

FDR Roundup

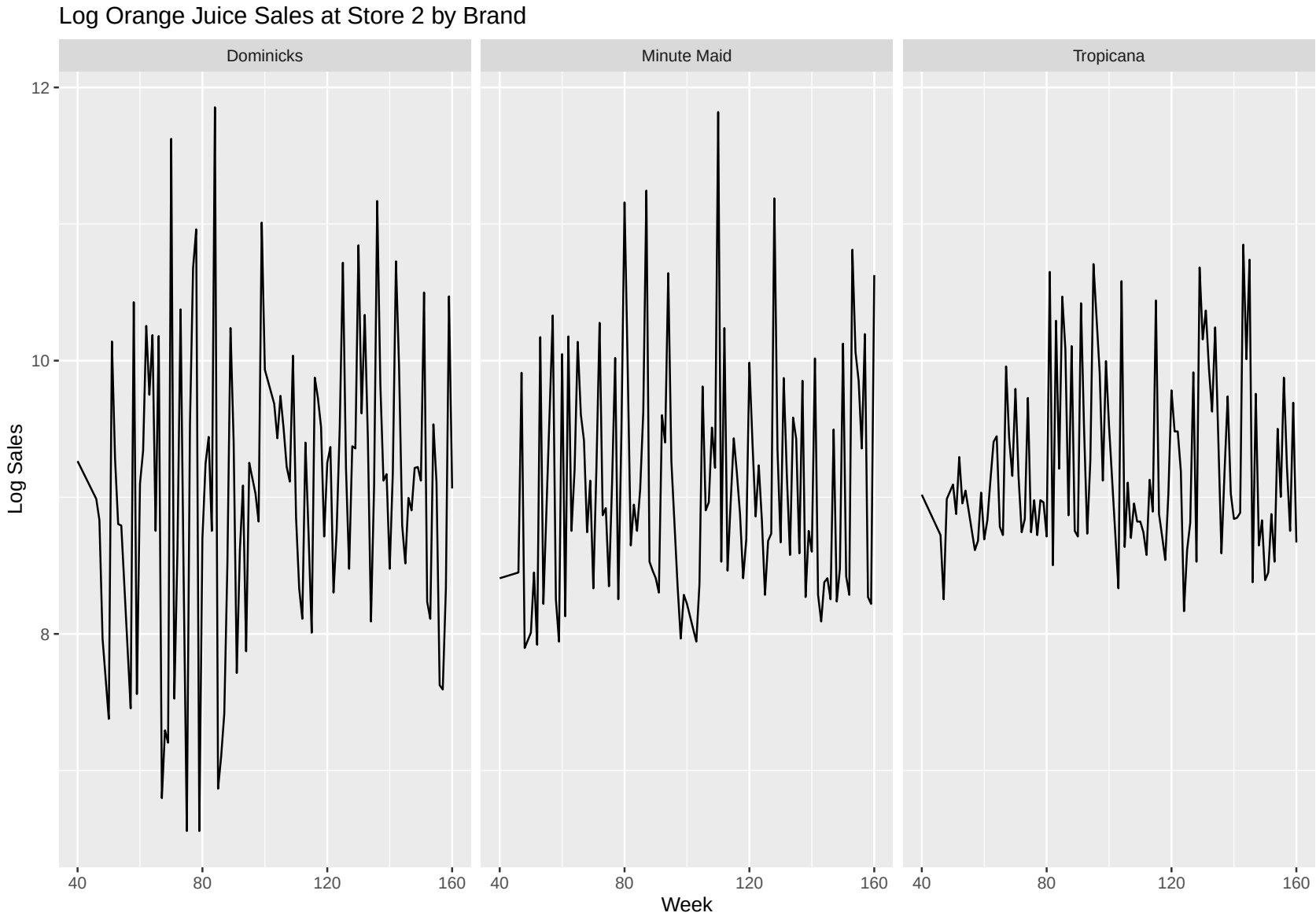


Loss

- Loss is a function both of our prediction and the true outcome
- More importantly, the driving feature of loss is our experience of making a certain error. Do we lose money? Time? Prestige?
- Our choice of procedure is driven by this loss.
- $l_p(Y, \hat{Y}) = l_p(Y - \hat{Y}) = l_p(e) = \left(\frac{1}{n}\sum_{i=1}^{n} |e|^p\right)^{\frac{1}{p}}$
  - E.g. $l_2(e) = \sqrt{(\frac{1}{n}\sum_{i=1}^{n} e^2)}$.

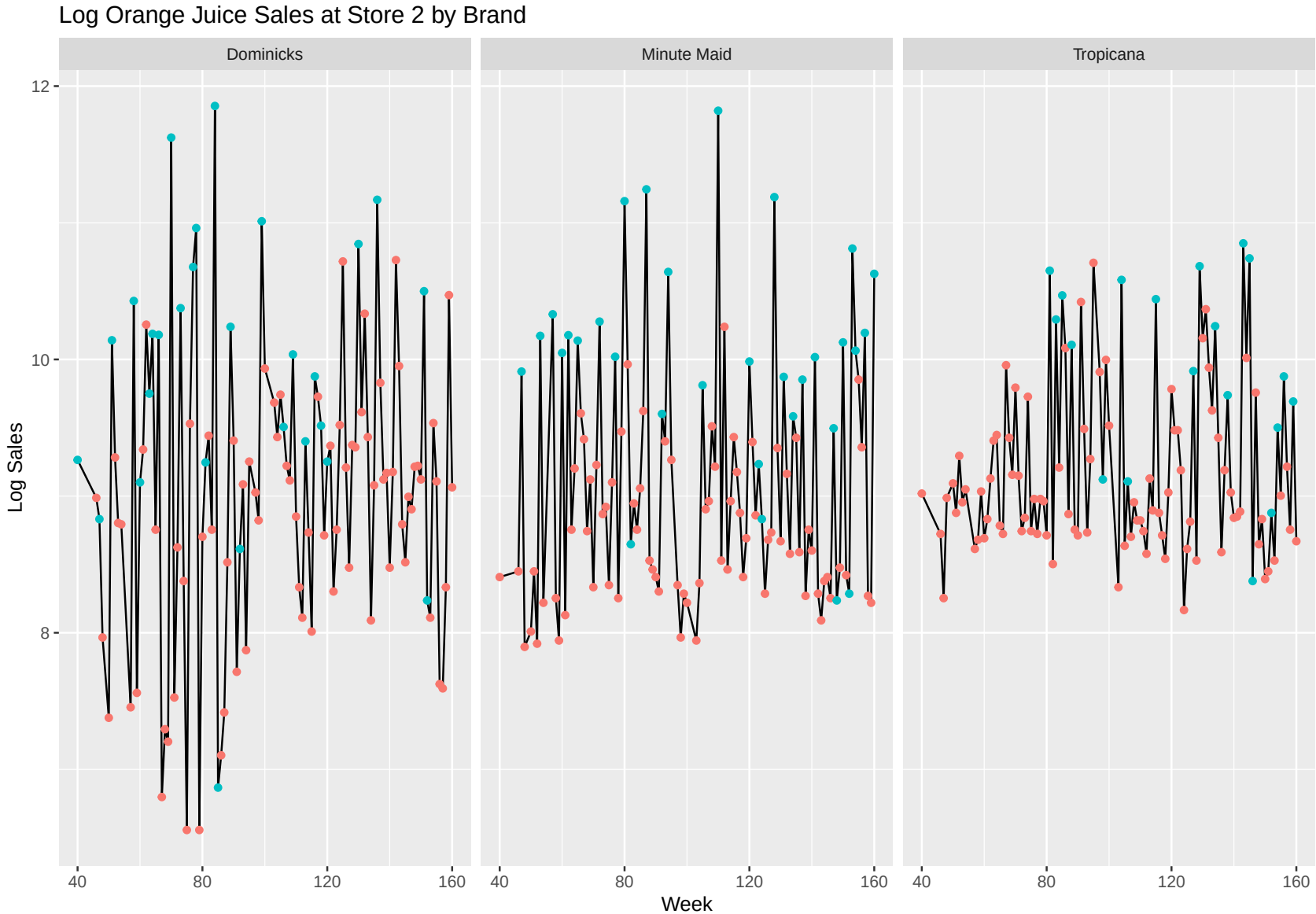
Regression

Motivation



What is driving sales? Brand differences? Price changes? Ads?

Motivation



Blue points are based on ongoing promotional activity.

It looks like ads are important.

Motivation

Fit a line for sales by brand controlling for promotional activity.

$$log(Sales) \approx \alpha + \gamma Brand + \beta Ads$$

$\alpha + \gamma_b$ are like our baseline sales. But we can bring in β more sales with some promotional activity.

Regression

- Regression through linear models
  - Implementation in R
- Complications:
  - Interaction
  - Factors
- Logistic Regression
- Estimation: Maximum likelihood, Minimum Deviance

This should be mostly review, but perhaps with a different emphasis.

Linear Models

Many problems involve a response or outcome ($\mathtt{y}$),
And a bunch of covariates or predictors ($\mathtt{x}$) to be used for regression.

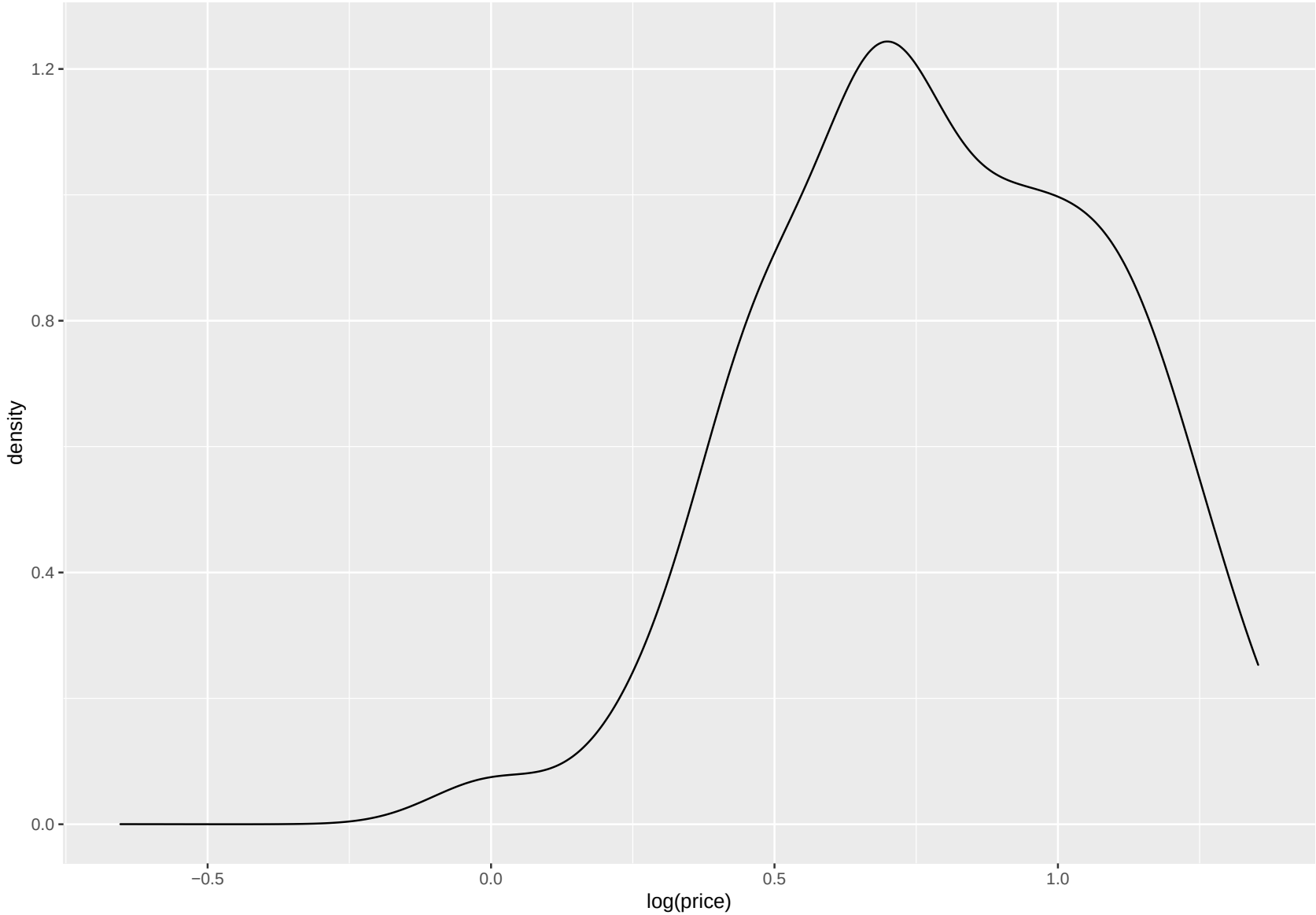
A general tactic is to deal in averages and lines.

$$E[y|x] = f(x'\beta)$$

Where $x = [1, x_1, x_2, x_3, ..., x_p]$ is our vector of covariates. (Our number of covariates is p again)
$\beta = [\beta_0, \beta_1, \beta_2, ..., \beta_p]$ are the corresponding coefficients.
The product $x'\beta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$.

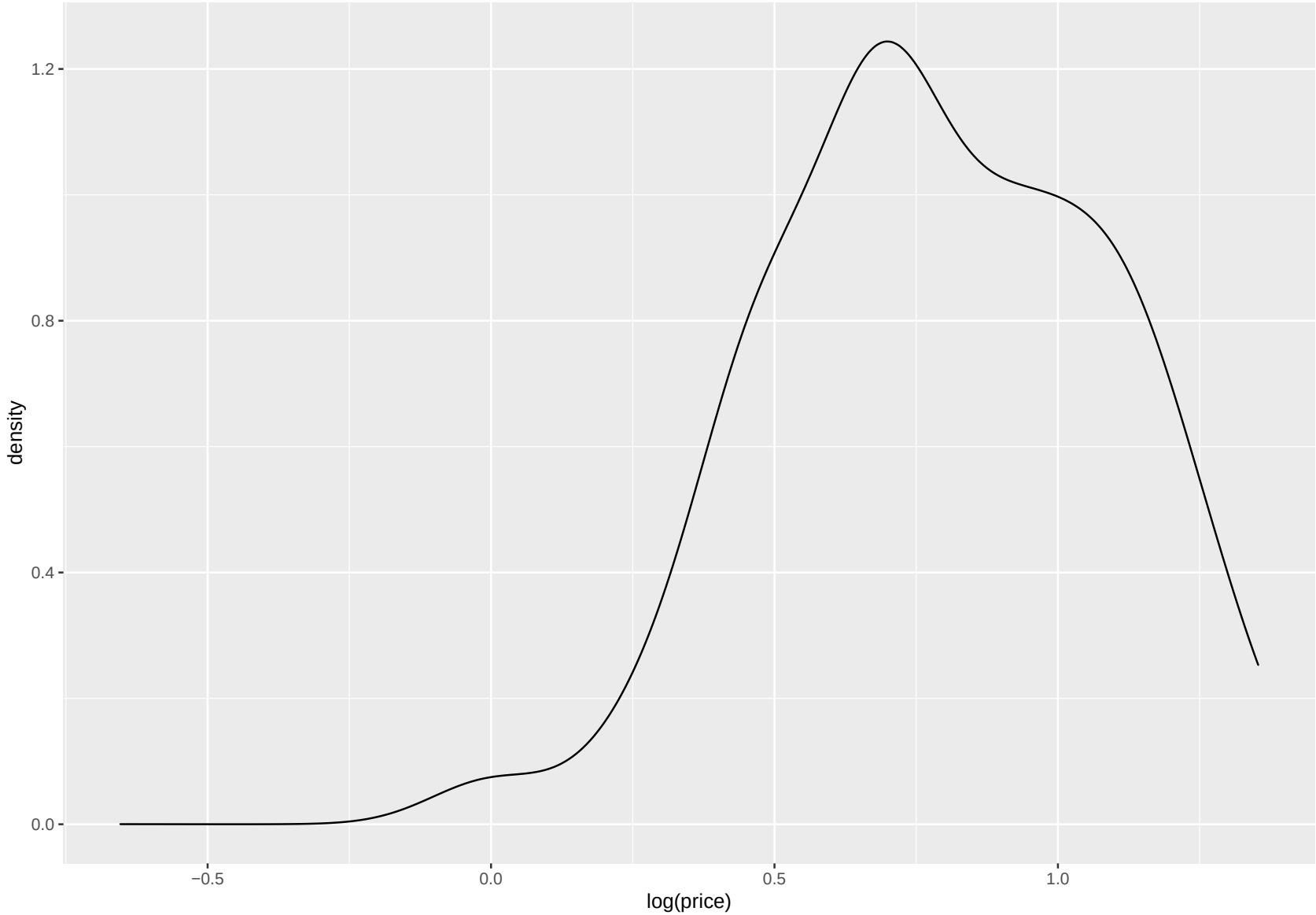
For simplicity we denote $x_0 = 1$ to estimate intercepts

Marginals and Conditional Distributions



Marginal Distribution shows us that prices are widely distributed

Marginals and Conditional Distributions



Marginal Distribution shows us that prices are widely distributed

Marginal vs Conditional

The Marginal mean (aka unconditional mean) is a simple number.

The conditional mean is a function that depends on covariates.

The data is distributed randomly around these means.

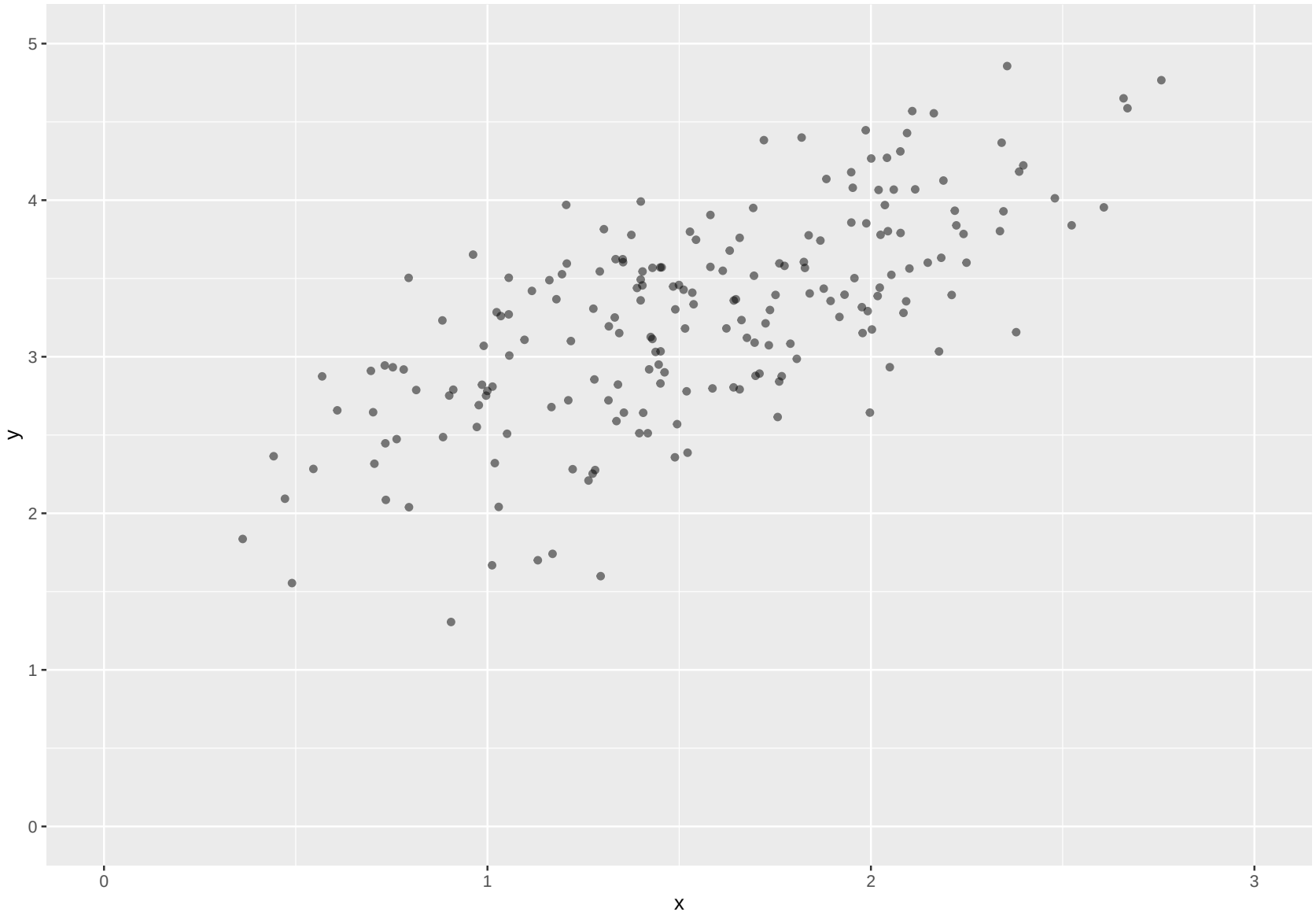
Generate some well-behaved data to demo.

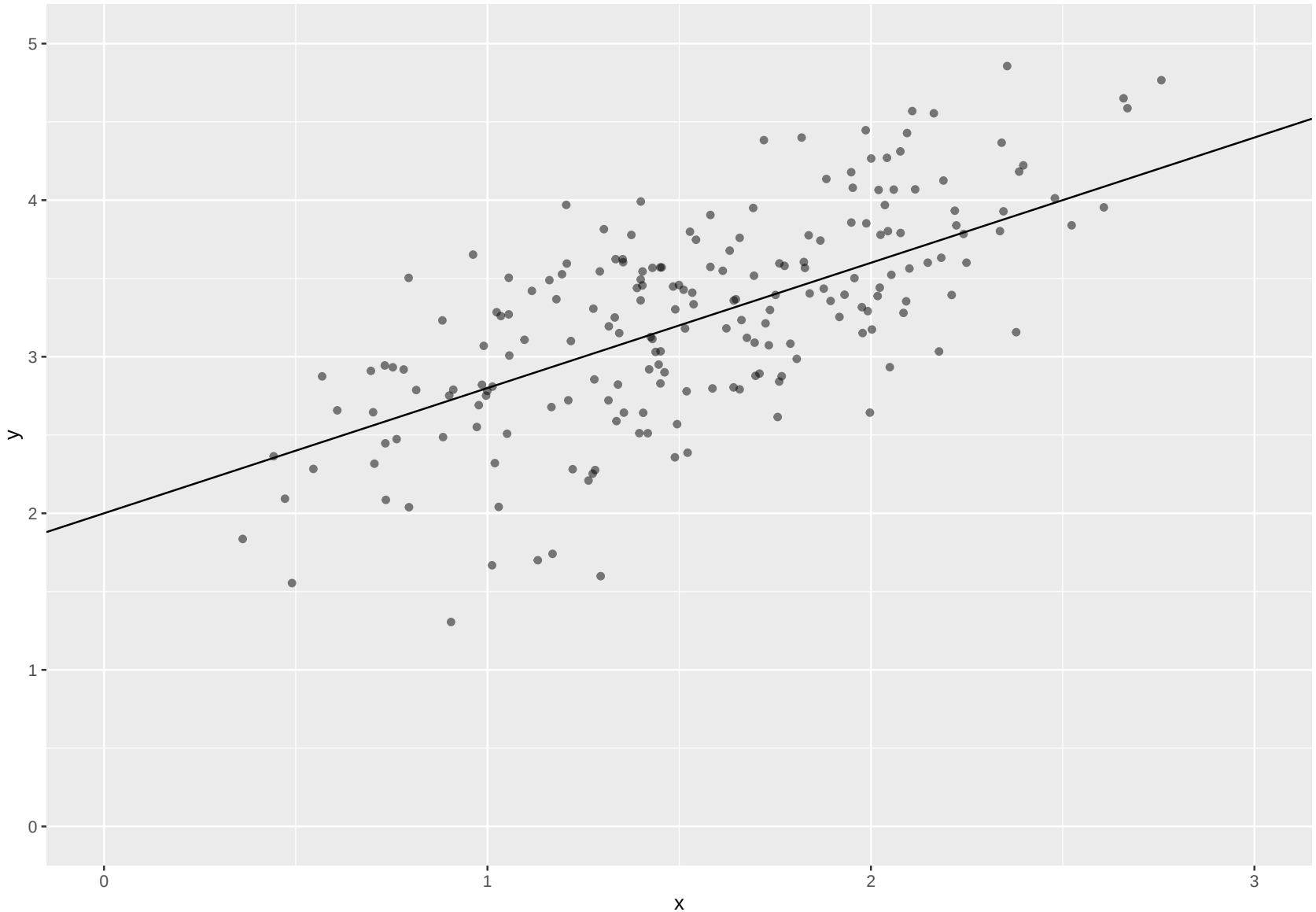
$$X \sim N(1.5, 0.5^2)$$

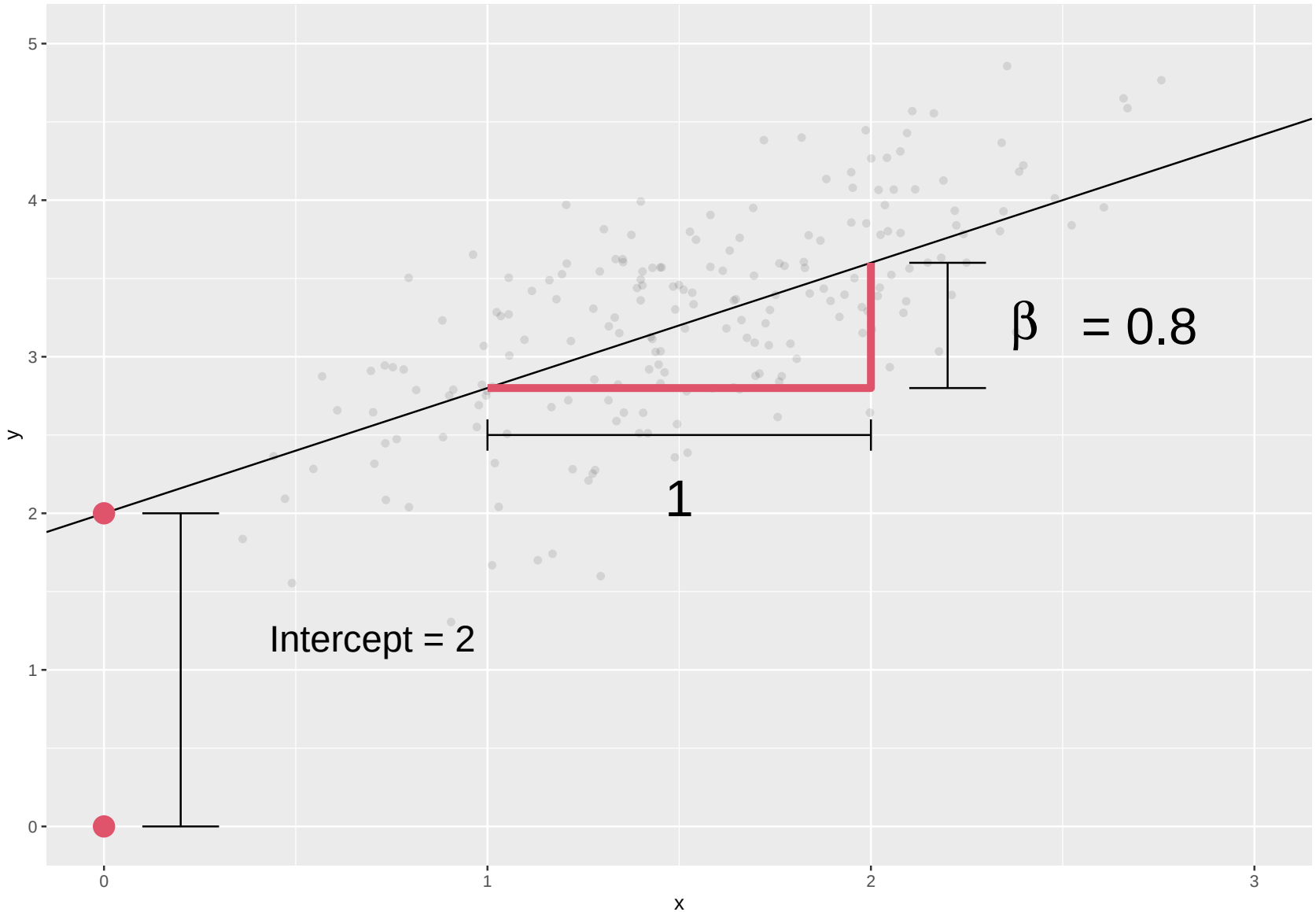
$$\epsilon \sim N(0, 0.5^2)$$

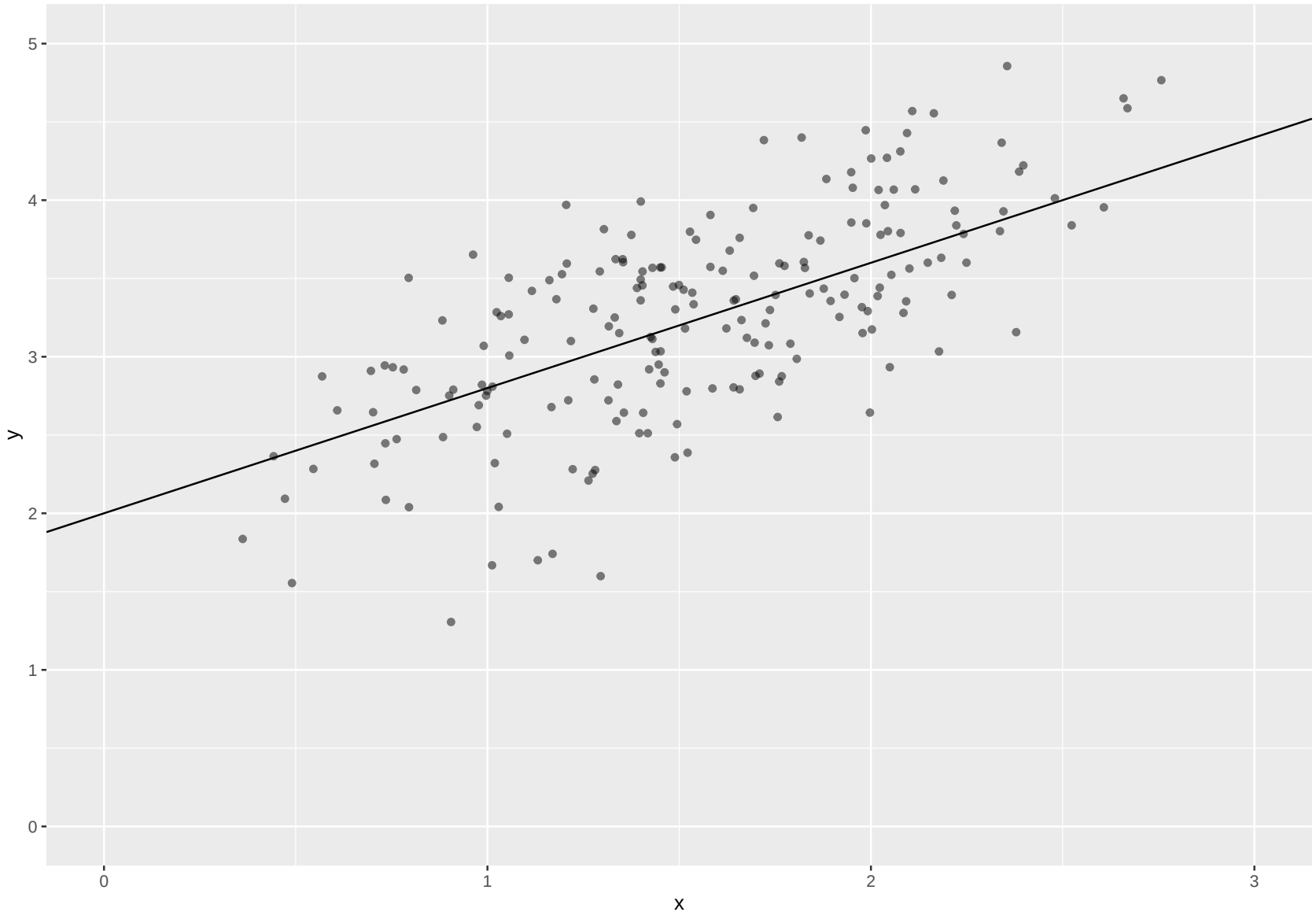
$$Y = 2 + 0.8X + \epsilon$$

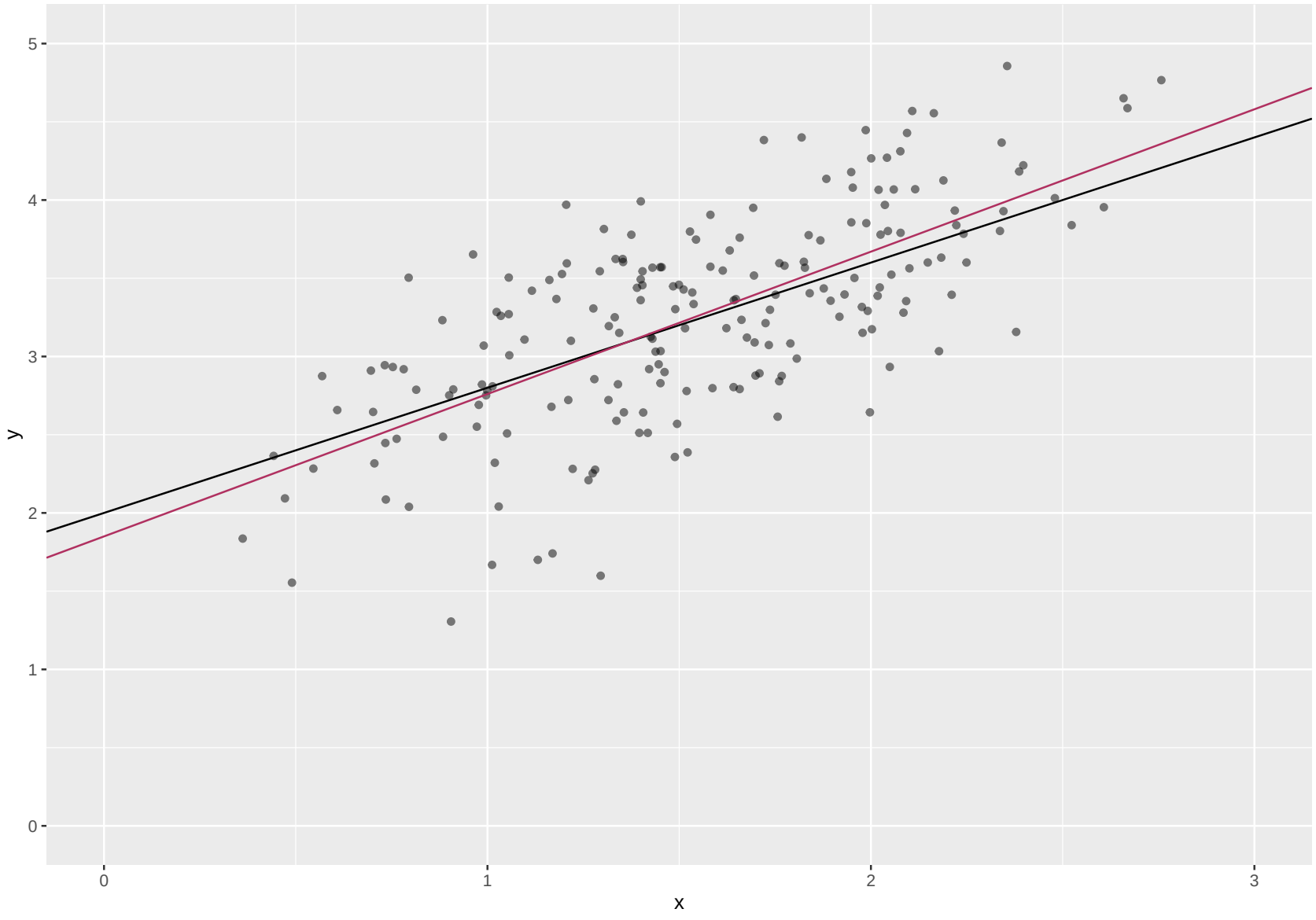
$$\therefore \quad Y|X \sim N(2 + 0.8X, 0.5^2)$$


5
4
3
2
1
0
y
0
1
2
3
x


5
4
3
2
1
0
y
0
1
2
3
x


5
4
3
2
1
0
y
0
1
2
3
x
β = 0.8
1
Intercept = 2


5
4
3
2
1
0
y
0
1
2
3
x


5
4
3
2
1
0
y
0
1
2
3
x

Coefficient Selection

The simplest form of regression is known as "Ordinary Least Squares" (OLS).

We select coefficents β to minimize the sum of the squared prediction errors.

$$\hat{\beta} = \underset{\beta}{argmin} \sum_{i=1}^{N} (y - \hat{y})^2 = \underset{\beta}{argmin} \sum_{i=1}^{N} (y - x'\beta)^2$$

$$= \underset{\beta}{argmin} \sum_{i=1}^{N} (y - (\beta_0 + \beta_1 x_1 + ... \beta_K x_K))^2$$

$$= \underset{\beta}{argmin} \left(\sum_{i=1}^{N} (y - x'\beta)^2 \right)^{\frac{1}{2}}$$

Implications

Standard regressions have a loss function built in alread. The l_2 loss. And they minimize it.

We minimized our l_2 loss and didn't wind up at the true value β. So the loss for β must be higher.

$\implies$ The variance for the true model is usually higher than the variance of our model's error terms.

- See *overfit* next week
- This is why we compare to a t-distribution.
  - We had to *estimate* the variance as well.

OLS vs MLE

There is a different way to choose coefficients, known as Maximum Likelihood.

Instead of minimizing a loss function, makes assumptions about error distribution. More later.

Basic Regressions in R

Orange Juice Sales

Three brands.

- Tropicana
- Dominicks
- Minute Maid

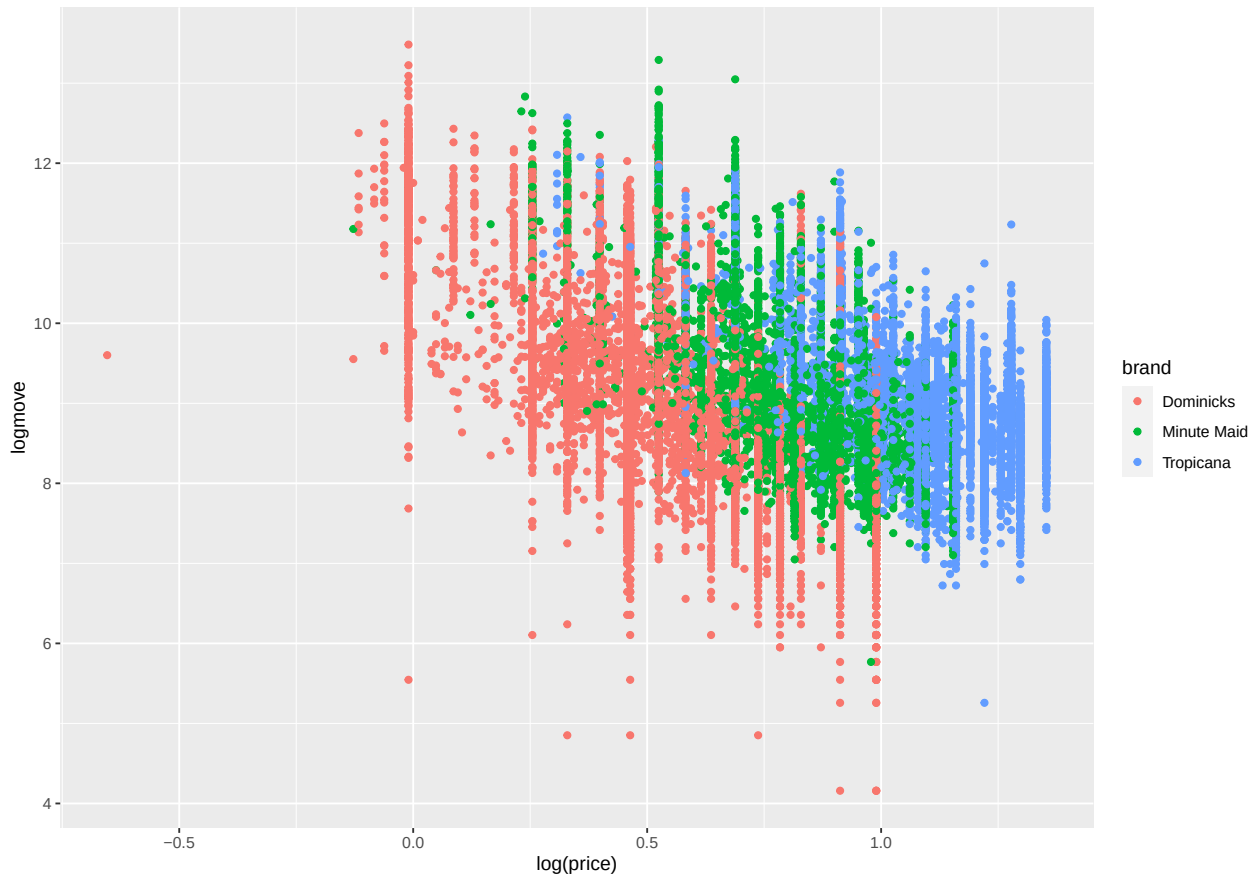
83 Chicagoland stores.
Demographics for each.

Price, sales (log units moved), and advertising activity (`feat`) on a number of dates.

Data in `oj.csv`.

Price, Brand, Sales

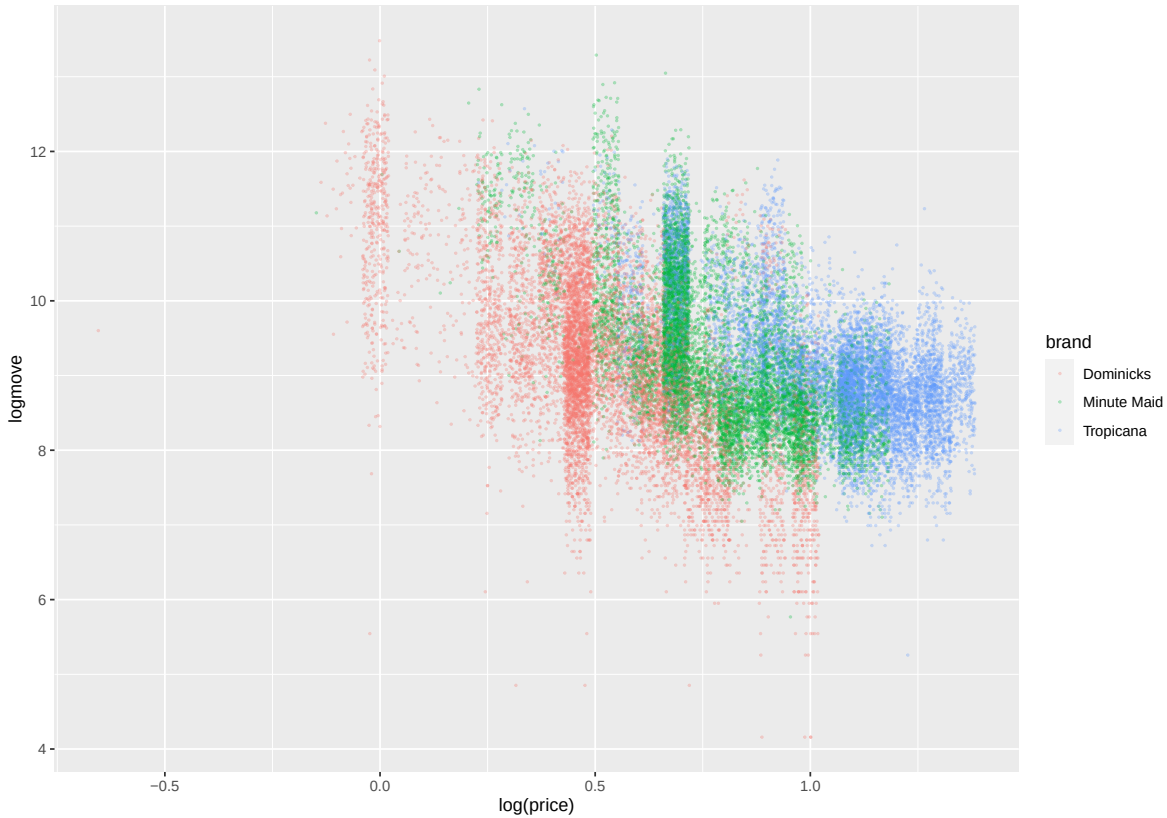
```
ggplot(oj, aes(y=logmove,x=log(price),col=brand))+
  geom_point()
```



Sales decrease with price. Each brand has its own price strategy.

Price, Brand, Sales

```
ggplot(oj,aes(y=logmove,x=log(price),col=brand))+
  geom_jitter(alpha=0.3,size=0.3,width=0.03)
```



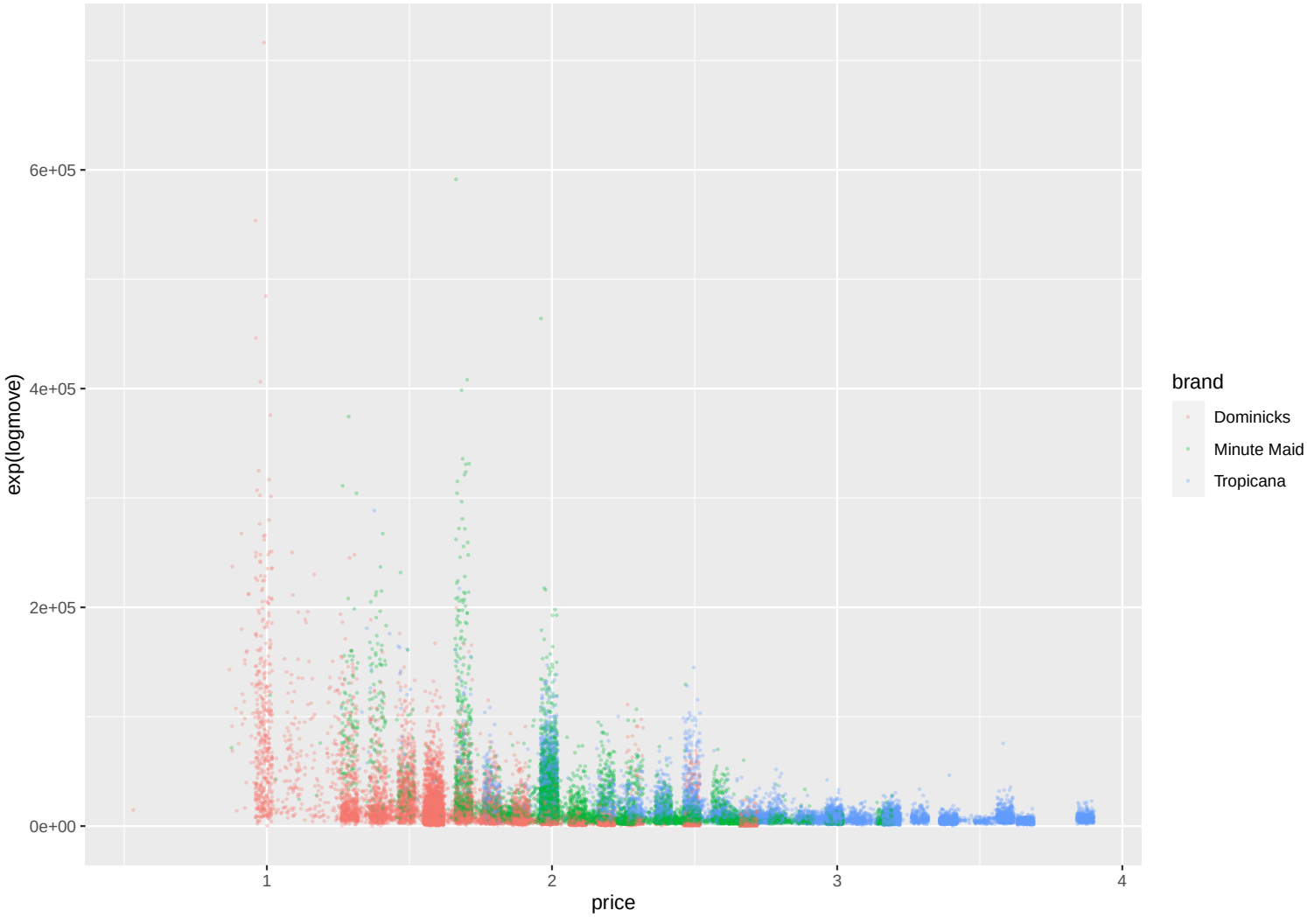
Sales decrease with price. Each brand has its own price strategy.

Scale

Why is this a log-log plot?

Scale

Why is this a log-log plot?



Log Linear

We often model the mean for $log(y)$ rather than y.

$$log(y) = log(a) + x\beta \iff y = ae^{x\beta}$$

Predicted y is multiplied by e^{β} after x increases by 1.

Recall: $log(y) = z \iff e^z = y$ where $e \approx 2.718$.
Further, $log(ab) = log(a) + log(b)$ and $log(a^b) = b\ log(a)$.

We define $log = ln$, the natural log. Using, e.g. log_2 would only change our intercept.

Whenever y changes on a percentage scale, use $log(y)$.

- prices: "Orange Juice is 10% off"
- sales: "Sales are up 10%"
- Most other things that are strictly positive may warrant a *log* scale.

Price Elasticity

A simple 'elasticity' model for orange juice sales y:

$$E[log(y)] = \gamma log(price) + x'\beta$$

Elasticities and log-log regressions: We can interpret γ as the % change in y associated with a 1% change in x.

In R:

```
coef(glm(logmove~log(price)+brand,data=oj))
```

```
##       (Intercept)       log(price) brandMinute Maid   bra
##        10.8288216        -3.1386914         0.8701747
```

We see that (with brand FEs), sales drop 3.1% for every 1% increase in price.

Regression in R

That one command is capable of doing a lot of work.

```
reg = glm(logmove~log(price)+brand,data=oj)
```

- `glm` stands for "Generalized Linear Regression
  - For this simple model, `lm` works too.
- `y ~ a + b` is the formula defining our regression.
- `y ~ .` will regress against everything in the dataset.
- Intercepts are implicitly included by default.

The variable `reg` is a list of useful things. (try `names(reg)`).

- `summary(reg)` prints a ton of information
- `coef(reg)` gives coefficients
- `predict(reg, newdata=newdataframe)` gives predictions for new data
  - `newdataframe` must be a dataframe with exactly the same format as mydata. (Variable names, factor levels, etc).

Factors

The Design Matrix

```
coef(reg)
```

```
##       (Intercept)       log(price) brandMinute Maid   bra
##        10.8288216        -3.1386914         0.8701747
```

What is `brandTropicana`?

Our regression formulas look like $\beta_0 + \beta_1 x_1 + \cdots$. But `brand` is not a number, so *brand* $\times \beta$ isn't sensible.

`glm`, `lm`, most other standard routines start by creating a numeric design matrix, which converts the factor `brand` into a lot of Boolean variables.

```
inds = sample(nrow(oj),10) #Grabbing some random observati
oj$brand[inds]
```

```
##  [1] Tropicana   Dominicks   Tropicana   Dominicks   Tro
##  [7] Minute Maid Tropicana   Tropicana   Minute Maid
## Levels: Dominicks Minute Maid Tropicana
```

The Design Matrix

They convert with `model.matrix`, which gives us these variables.

```
model.matrix(reg)[inds,]
```

```
##       (Intercept) log(price) brandMinute Maid brandTropi
## 9053            1  1.1600209                 0
## 6859            1  0.4637340                 0
## 6292            1  1.2556160                 0
## 9693            1  0.4574248                 0
## 26364           1  1.0952734                 0
## 18647           1  0.9001613                 1
## 13705           1  0.9631743                 1
## 23454           1  1.2208299                 0
## 8418            1  0.6881346                 0
## 8607            1  0.7839015                 1
```

These are numeric values x which we can multiply against coefficients β.

Intercepts

```
coef(reg)
```

```
##       (Intercept)       log(price) brandMinute Maid   bra
##        10.8288216        -3.1386914         0.8701747
```

```
model.matrix(reg)[1,]
```

```
##       (Intercept)       log(price) brandMinute Maid   bra
##          1.000000          1.353255         0.000000
```

Each factor's reference level is absorbed by the intercept. The coefficients are now "changes relative to intercept".

In this case, the intercept is "dominicks" brand OJ.

Releveling Factors

To check your factor's reference level, look at the first level in the factor levels.

```
levels(oj$brand)
```

```
## [1] "Dominicks"   "Minute Maid" "Tropicana"
```

You can change this with `relevel` (base R) or `fct_relevel` and `fct_reorder` in the `tidyverse`'s `forcats` package.

Interactions

Interactions

Beyond Additive effects: Variables change how other variable's act on y.

An Interaction is the product of two covariates.

$$E[y|x] = ... + \beta_j x_j + x_j x_k \beta_{jk}$$

So the effect on $E[y]$ of a unit increase in x_j is $\beta_j + x_k \beta_{jk}$

$\Longrightarrow$ It depends on x_k!

Interactions play a massive role in statistical learning, and are often central to social science and business questions.

- Does gender change the effect of education?
- Do older patients benefit more from a vaccine?
- How does advertisement affect price sensitivity?

Fitting interactions in R: use * in your formula.

```
reg.int = glm(logmove~log(price)*brand,data=oj)
coef(reg.int)
```

```
##                   (Intercept)                   log(price)
##                    10.95468173                    -3.37752963
##              brandMinute Maid                brandTropicana
##                     0.88825363                     0.96238960
## log(price):brandMinute Maid log(price):brandTropicana
##                     0.05679476                     0.66576088
```

This model is $E[log(y)] = \alpha_b + \beta_b log(price)$.
A separate slope and intercept for each brand!

Elasticities wind up as: dominicks: -3.4, minute maid: -3.3, tropicana: -2.7

Where do these numbers come from?

Advertisements

What changes when a brand is featured?
Here, we mean an in-store display promo or flier.

We could model an additive effect on sales:
$E[log(sales)] = \alpha_b + 1_{[feat]}\alpha_{feat} + \beta_b log(p)$

Or that and an elasticity effect:
$E[log(sales)] = \alpha_b + \beta_b log(p) + 1_{[feat]}(\alpha_{feat} + \beta_{feat} log(p))$

Or a brand-specific effect on elasticity.
$E[log(sales)] = \alpha_b + \beta_b log(p) + 1_{[feat]}(\alpha_{feat} + \beta_{b,feat} log(p))$

See the R code for each of these online.

Brand-specific Elasticities - Coefs

```
reg.bse = glm(logmove~feat*log(price)*brand,data=oj)
coefs = coef(reg.bse)
coefs
```

```
##                                   (Intercept)
##                                   10.40657579
##                                    log(price)
##                                   -2.77415436
##                                brandTropicana                                    fe
##                                    0.70794089
##                     featTRUE:brandMinute Maid                                  featTR
##                                    1.17294361
##                   log(price):brandMinute Maid                              log(pric
##                                    0.78293210
## featTRUE:log(price):brandMinute Maid   featTRUE:log(pric
##                                   -1.10922376
```

Brand-specific Elasticities - Coefs

```
## # A tibble: 2 x 4
##   feat  Dominicks `Minute Maid` Tropicana
##   <lgl>     <dbl>          <dbl>     <dbl>
## 1 FALSE    -2.77          0.783     0.736
## 2 TRUE     -0.471        -1.11     -0.986
```

Ads seem to increase the price-sensitivity.

Minute Maid and Tropicana have big jumps.

Actual Elasticities

```
## # A tibble: 2 x 4
##   feat  Dominicks `Minute Maid` Tropicana
##   <lgl>     <dbl>          <dbl>     <dbl>
## 1 FALSE     -2.77          -1.99     -2.04
## 2 TRUE      -3.24          -3.57     -3.50
```

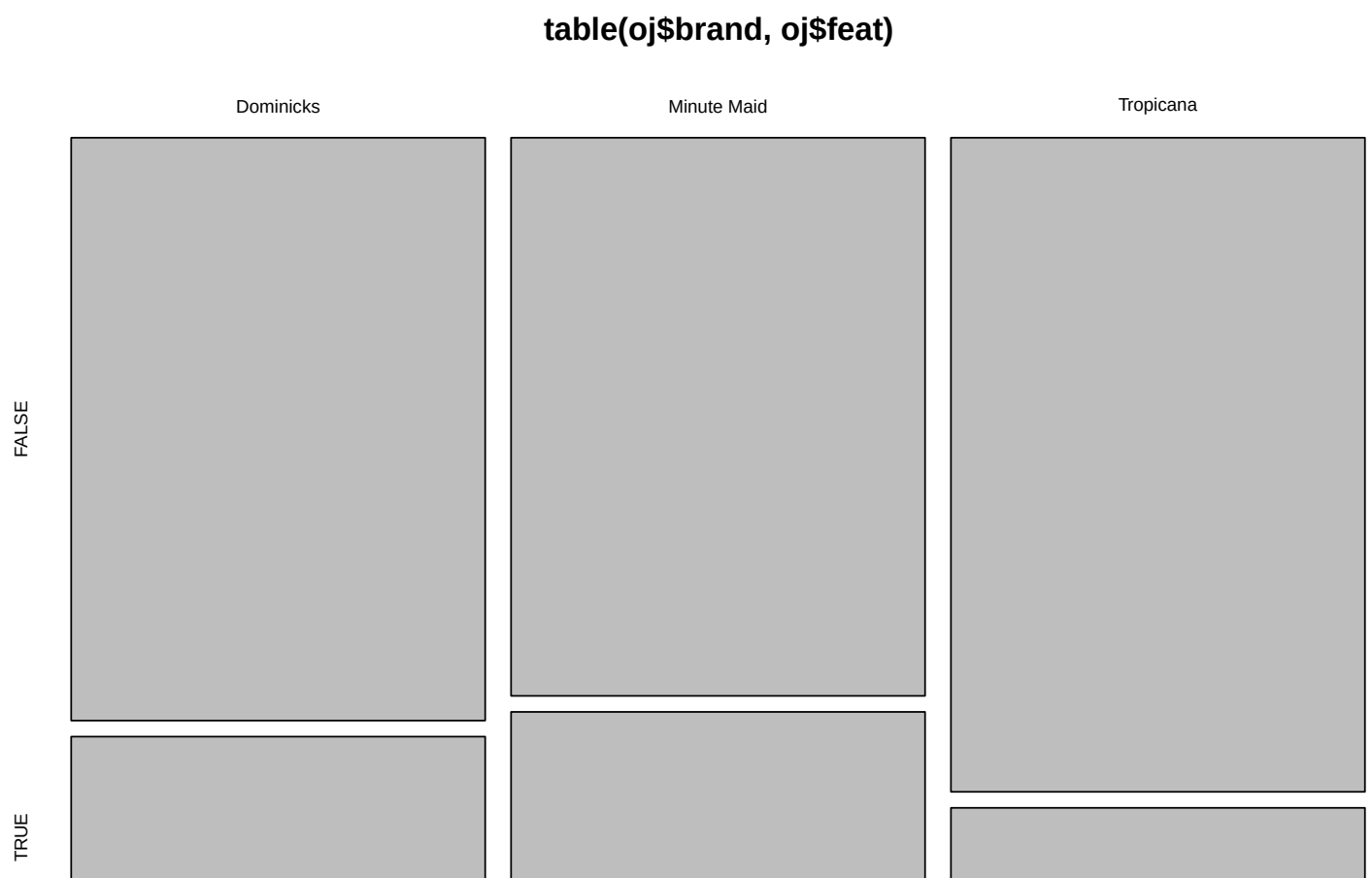
After marketing,

Why does marketing increase the price sensitivity? How will this influence strategy?

Confounding

There are differential rates of advertising, which makes going from these static coefficients, to statements about the companies, difficult.

```
plot(table(oj$brand,oj$feat))
```



Logistic Regression

Logits

Linear regression is just one linear model. Probably not the most used model.

Logistic Regression *when y is TRUE or FALSE (1/0).*

Binary Response as a prediction target:

- Profit or loss, greater or less than, pay or default
- Thumbs up or down, buy or not
- Win or not, Sick or healthy, Spam or not?

In high dimensions, simplifying to binary variables can help us interpret.

Building a linear model for **binary data**.
Recall our original specification: $E[Y|X] = f(x'\beta)$

The Response y is 0 or 1, leading to a conditional mean:

$$E[y|x] = P[y = 1|x] \times 1 + P[y = 0|x] \times 0 = P[y = 1|x]$$

$\Longrightarrow$ The expectation is a probability.

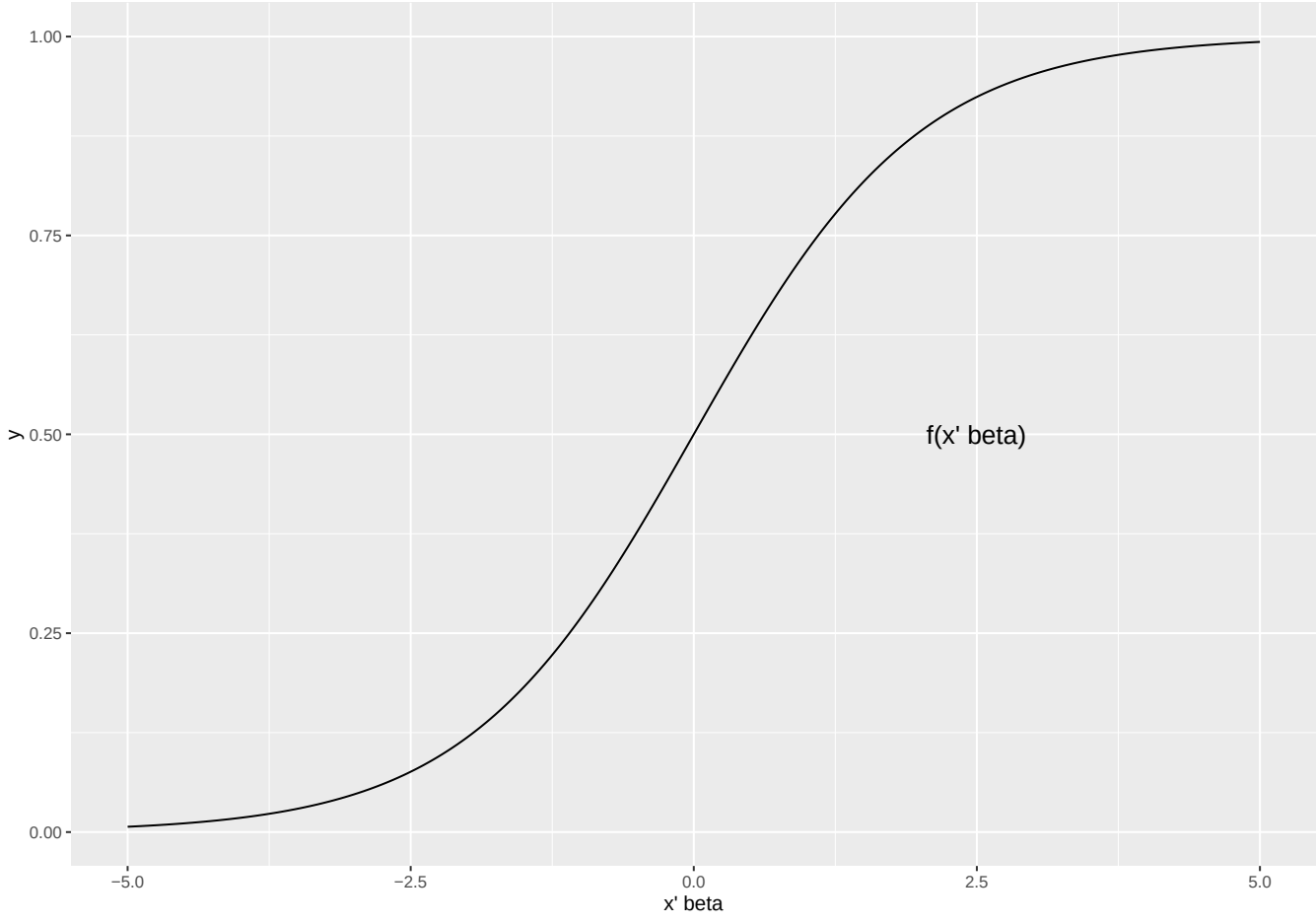
We will choose $f(x'\beta)$ to give values between 0 and 1.

Also note that the variance becomes related to the mean.

We want a binary choice model

$$p = P[y = 1|x] = f(\beta_0 + \beta_1 x_1 + ... + \beta_K x_K)$$

Where f is a function increasing in value from 0 to 1.



We will use the '**logit link**' function and do '**logistic regression**'.

$$P[y = 1|x] = \frac{e^{x'\beta}}{1 + e^{x'\beta}} = \frac{exp(\beta_0 + \beta_1 x_1 + ... + \beta_K x_K)}{1 + exp(\beta_0 + \beta_1 x_1 + ...\beta_K x_K)}$$

The 'logit' link is common, for a few reasons. One big reason? Some math shows:

$$log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_K x_K$$

So this is a linear model for log odds.

Spam Filters

Most inboxes use binary regression to predict whether an email is spam.
Say y=1 for spam, and y=0 for not spam.

`spam.csv` has 4600 emails with word and character presence indicators (1 if character is in message) and related info.

Logistic Regression fits $p(y = 1)$ as a function of email content.

Very easy in R.

```
glm(Y ~ X+Z, data=df, family=binomial)
```

The argument `family=binomial` tells `glm` that `spam` is binary.

The response can take a few forms:

- $y = 1, 1, 0, \ldots$ numeric vector
- $y = TRUE, FALSE, TRUE, \ldots$ logical vector
- \$y = 'spam','not', 'spam',... \$ factor vector Everything else is the same as for linear regression.

Perfect Separation

```
spam = glm(spam~.,data=email,family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or
```

Some emails are very easy to predict. Clearly spam or not. This is called 'perfect separation'. It can cause some numerical instability (convergence issues, standard errors can suffer, p-values, etc), but mostly doesn't matter much.

We see it here, because some words are excellent predictors.

```
table(email$word_george, email$spam,dnn=c("word_george","sp
```

```
##             spam
## word_george    0    1
##           0 2016 1805
##           1  772    8
```

Interpreting Coefficients

The model is:

$$\frac{p}{1-p} = exp\left(\beta_0 + \beta_1 x_1 + ... + \beta_K x_K\right)$$

So $exp(\beta_j)$ is the odds multiplier for a unit increase in x_j.

```
coef(spam)['word_george']
```

```
## word_george
##   -5.779841
```

So if the word 'george' occurs in an email, the odds of it being spam are multiplied by $exp(-5.8) \approx 0.003$.

What is the odds multiplier for a coefficient of 0?

Deviance

Deviance in R Summaries

The summary function gives coefficients, and a bunch more.

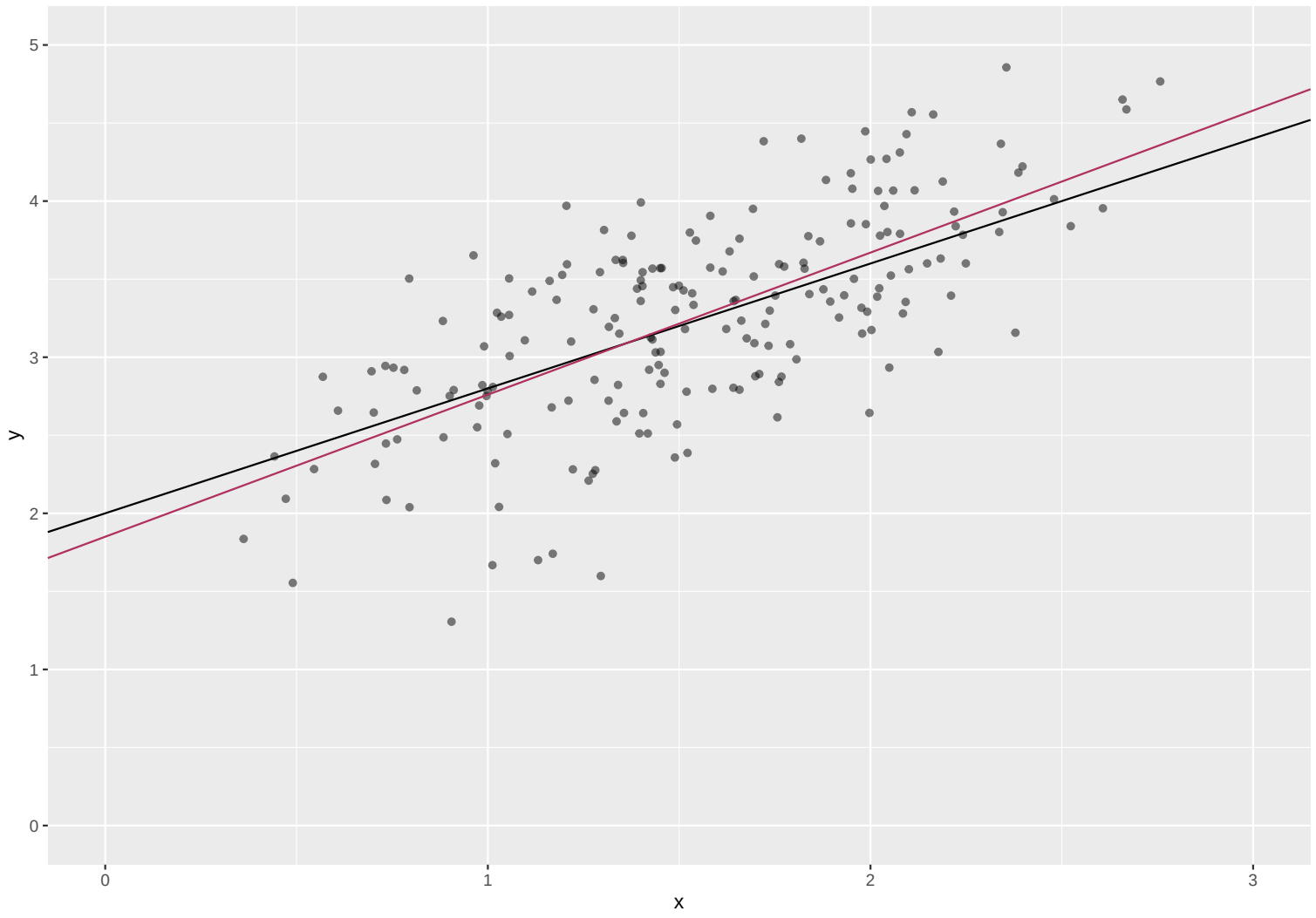
`summary(spam)`

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6170.2  on 4600  degrees of freedom
Residual deviance: 1548.7  on 4543  degrees of freedom
AIC: 1664.7

Number of Fisher Scoring iterations: 10
```

The linear OJ regression does the same `summary(reg.bse)`

```
(Dispersion parameter for gaussian family taken to be 0.4829706)

    Null deviance: 30079  on 28946  degrees of freedom
Residual deviance: 13975  on 28935  degrees of freedom
AIC: 61094
```

These are statistics telling us about our model fit. They are important in both linear and logistic regression. Understanding deviance will tie them together.


5
4
3
2
1
0
y
0
1
2
3
x

Estimation and Goodness-of-Fit

Two related concepts

Likelihood is a function of the unknown parameters β of a *statistical model*, given data:

$$L(\beta|data) = P[data|\beta]$$

Maximum Likelihood Estimation (MLE)

$$\hat{\beta} = argmax_{\beta} L(\beta|data)$$

MLE estimates $\hat{\beta}$ are the parameters that make our data *most likely*.

Maximum likelihood estimation *requires* making statements about distribution of error terms.

Error Distributions

Back to Gaussian linear model for a moment:

$$Y|X \sim N(x'\beta, \sigma^2)$$

$$\implies P[Y = y|X = x] = \phi\left(\frac{y - x'\beta}{\sigma}\right)$$

Then we can aggregate across all observations and pick a β.

$$\implies L(\beta|data) = P[data|\beta] = \prod_{i=1}^{N} P[Y = y|X = x]$$

$$= \prod_{i=1}^{N} \phi\left(\frac{y - x'\beta}{\sigma}\right)$$

Where ϕ is the normal distribution's pdf.

Estimation and Goodness-of-Fit

Two related concepts

Deviance refers to the distance between our fit and the data. You generally want to minimize it.

$$Dev(\beta) = -2log(L(\beta|data)) + C$$

We can ignore C for now.

Deviance is useful for comparing models. It is a measure of GOF that is similar to the residual sum of squares, for a broader class of models (logistic regression, etc).

We'll think about deviance as a cost to be minimized.

Minimize Deviance $\iff$ Maximize likelihood

Bringing this full circle.

$$\phi(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{x^2}{2}\right)$$

Given n independent observations, the likelihood becomes:

$$\prod_{i=1}^{n} \phi\left(\frac{y - x'\beta}{\sigma}\right) \propto \prod_{i=1}^{n} exp\left(-\frac{(y - x'\beta)^2}{2\sigma^2}\right)$$

$$\propto exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y - x'\beta)^2\right)$$

This leads to Deviance of:

$$Dev(\beta) = -2log(L(\beta|data)) + C = \frac{1}{\sigma^2}\sum_{i=1}^{n}(y - x'\beta)^2 + C'$$

So:
Min Deviance $\iff$ Max likelihood $\iff$ Min l_2 loss

This is just a particular loss function, which is driven by the distribution of the ϵ terms.

MLE for Logistic Regression

Our logistic regression has the following likelihood:

$$L(\beta) = \prod_{i=1}^{n} P[y_i|x_i] = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

$$= \prod_{i=1}^{n} \left(\frac{exp(x_i'\beta)}{1+exp(x_i'\beta)}\right)^{y_i} \left(\frac{1}{1+exp(x_i'\beta)}\right)^{1-y_i}$$

Thus the deviance to minimize is:

$$Dev(\beta) = -2\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i)log(1-p_i))$$

$$\propto \sum_{i=1}^{n}[log(1+exp(x_i'\beta)) - y_i x_i'\beta]$$

This is just taking the logs and removing the factor of 2.

Back to our summary outputs. We can print the same output for both linear and logistic regressions.

But the "dispersion parameter" is always 1 for the logistic regression. `summary(spam)`

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6170.2  on 4600  degrees of freedom
Residual deviance: 1548.7  on 4543  degrees of freedom
AIC: 1664.7

Number of Fisher Scoring iterations: 10
```

'degrees of freedom' is actually 'number of observations - df' where df is the number of coefficients estimated in the model.

Specifically `df(deviance) = nobs - df(regression)`

You should be able to back out number of observations from the R output.

Dispersion parameter for Linear regression?

```
summary(reg.bse)
(Dispersion parameter for gaussian family taken to be 0.4829706)

    Null deviance: 30079  on 28946  degrees of freedom
Residual deviance: 13975  on 28935  degrees of freedom
AIC: 61094
```

Remember our basic gaussian model was:

$$Y|X \sim N(X'\beta, \sigma^2)$$

And the implied deviance was:

$$Dev(\beta) = \frac{1}{\sigma^2}\sum_{i=1}^{n}(y - x'\beta)^2 + C'$$

σ is the dispersion parameter, and it is critical here. The logit has a mean-variance link, so we don't need the separate param.

Estimating σ

$$y_i = x_i'\beta + \epsilon_i; \quad \sigma^2 = Var(\epsilon)$$

Denote the residuals, $r_i = y_i - x_i'\hat{\beta}$.

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=1}^{n} r_i^2$$

R calls $\hat{\sigma}^2$ the dispersion parameter.

Critically, even if we know β, we only predict sales with uncertainty. E.g., approximately a 95% chance of sales in $x'\beta \pm 2\sqrt{0.48}$

R^2

Residual Deviance, D is what we've minimized using x.
Null Deviance D_0 is for the model without x (or more generally, the model under the null).
i.e. $\hat{y}_i = \bar{y}$

- $D_0 = \sum(y_i - \bar{y})^2$ in linear regression
- $D_0 = -2\sum[y_i log(\bar{y}) + (1 - y_i) log(1 - \bar{y})]$ in logits

The difference between D and D_0 comes from information in x.

Proportion of deviance explained by x is called the R^2 in a linear regression, "Pseudo-R^2" in logit.

$$R^2 = \frac{D_0 - D}{D_0} = 1 - \frac{D}{D_0}$$

This measures how much variability you explain with your model.

- In `spam`: $R^2 = 1 - 1549/6170 = 0.75$
- In OJ – `reg.bse`: $R^2 = 1 - 13975/30079 = 0.54$

R^2 in linear regression

Recall that for linear model, deviance is the sum of squared errors (SSE) and D_0 is the total sum of squares (TSS).

$$R^2 = 1 - \frac{SSE}{TSS}$$

You may also recall that $R^2 = corr(y, \hat{y})^2$.

```
cor(reg.bse$fitted,oj$logmove)^2
```

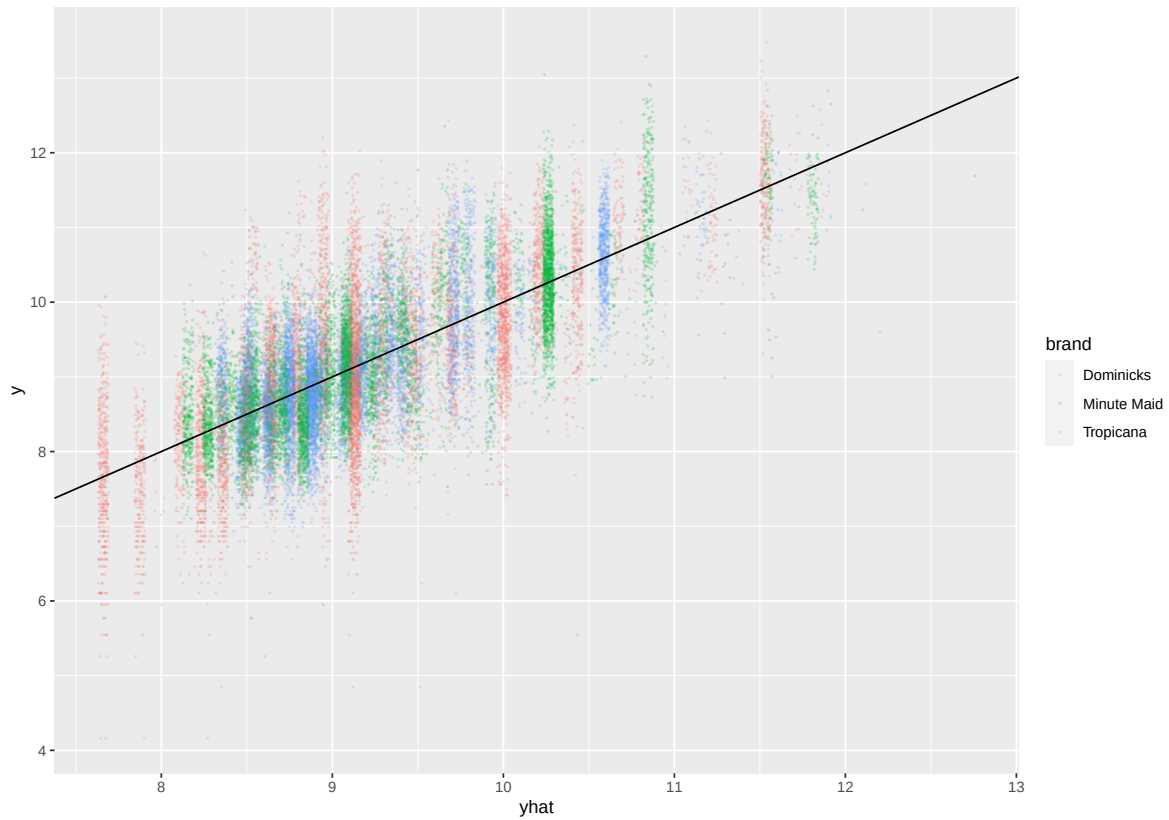
```
## [1] 0.5353939
```

For linear regression, min deviance = max corr$(y, \hat{y})$. $\implies$ if y vs $\hat{y}$ is a straight line, you have a perfect fit.

Also implies that R^2 (weakly) increases whenever we add another variable.

Fit plots

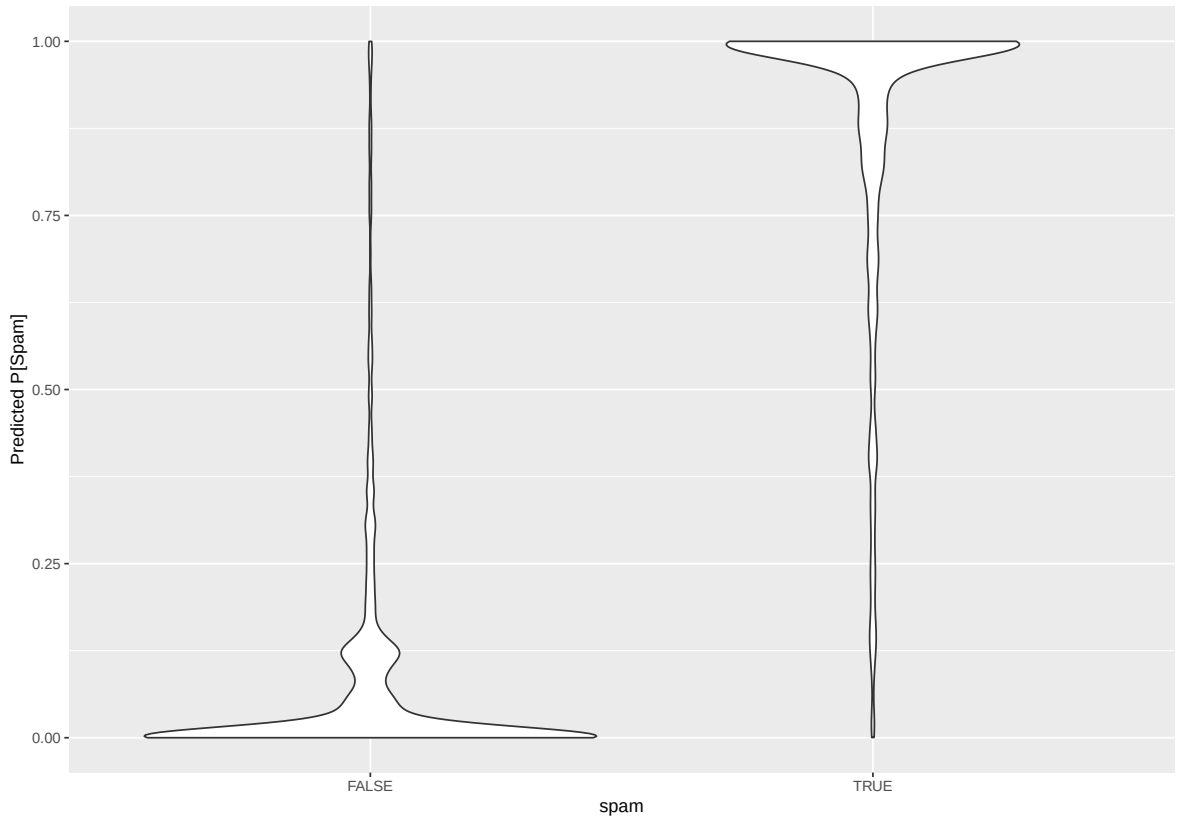
```
fitplotdf = data.frame(y = oj$logmove,yhat= predict(reg.bse
ggplot(fitplotdf,aes(y=y,x=yhat,col=brand)) +
  geom_jitter(alpha=0.2,size=0.2,width=0.03)+
  geom_abline(intercept=0,slope=1)
```



It is good practice to plot y vs $\hat{y}$. It helps you check for

Fit plots – logit

```
logit.fit.df = data.frame(spam=email$spam==1,yhat=predict(s
ggplot(logit.fit.df,aes(y=yhat,x=spam,)) +
  geom_violin()+ylab("Predicted P[Spam]")
```



New question: how do you choose the classification threshold?
When do you send to the spam folder?

Prediction

Prediction is easy with `glm`.

```
predict(spam,newdata=email[1:4,])
```

```
##         1         2         3         4
##  2.029963 10.956507 10.034045  5.656989
```

But this output is $x'\beta$. To get probabilities $exp(x'\beta)/(1 + exp(x'\beta))$, add `type="response"`

```
predict(spam,newdata=email[1:4,],type="response")
```

```
##         1         2         3         4
## 0.8839073 0.9999826 0.9999561 0.9965191
```

Newdata must match the format of the original data.

Out-of-Sample Prediction

We care about how well our model works out-of-sample.

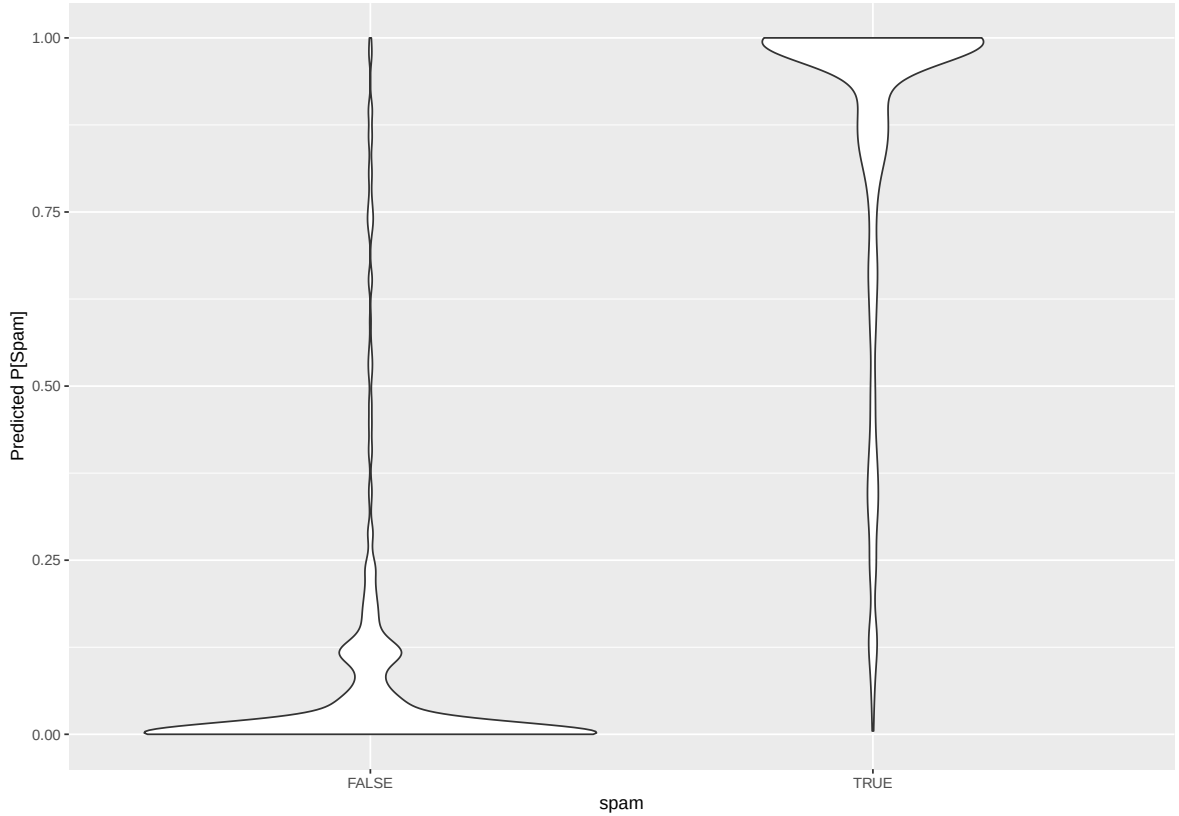
One way to test this is to use a "validation sample".

1. Randomly split your data into two samples
   - usually named "testing" and "training".
2. Fit your model using the "training" data
3. Test predictions on the left-out "testing" data.

```
#Sample 1000 indices
leaveout = sample.int(nrow(email),1000)
#Train the model
spam_train = glm(spam~.,data=email[-leaveout,],family="bino
#Predict performance on the test data.
test_preds = predict(spam_train,newdata=email[leaveout,],ty
```

Out-of-Sample Prediction

Fit plots for those left-out observations.



For the leave out data, we get $D_0 = 1360$ and $D = 362$ for $R^2 = 0.73$.

Since the leave-out sample is random, your results may vary.

Note: OOS R^2 is lower than in-sample $R^2 = 0.75$

Out-of-Sample

More on this next week during model selection.

You might see notions of this in my code for predicting vaccinations.

Wrap up

Things to do

Before Thursday:

- Literally nothing

Rehash

- Regressions minimize prediction errors, loss, or deviance, or they maximize some likelihood function.
- Interpretation of coefficients depends on variables: elasticities, levels, logs
- Factor coefficients wind up being "difference from some reference
- Interactions allow us to model the effect of one covariate on another
- Logistic regression lets us predict binary variables using a linear model of log-odds
- Deviance's are a measure of prediction error which generalize beyond linear regressions.
- Out-of-sample performance is key.

Bye!