Models

In-class questions: 2/29/2016

We're going to try something new today. If you have small technical questions during class, go to this link and ask away.

Statistical Models

"All models are wrong, but some are useful" -George E. P. Box

When we see a confidence interval or p-value, it means a probability distribution of some sort was used to quantify the null hypothesis. Many times deciding which probability distribution to use is relatively straightforward. For example, when betting on black on roulette we can use simple probability calculations to determine probability distribution of winnings.

The CLT is backed by theoretical results that guarantee that the approximation is accurate. However, we cannot always use this approximation, such as when our sample size is too small. Previously, we described how the sample average can be approximated as t-distributed when the population data is approximately normal. However, there is no theoretical backing for this assumption. We are now *modeling*. In the case of height, we know from experience that this turns out to be a very good model.

But this does not imply that every dataset we collect will follow a normal distribution. An examples are US incomes. The normal distribution is not the only parametric distribution that is available for modeling. Here we provide a very brief introduction to some of the most widely used parametric distributions and some of their uses in the life sciences. We focus on the models and concepts needed to understand the techniques currently used to perform statistical inference on high-throughput data. To do this we also need to introduce the basics of Bayesian statistics. For more in depth description of probability models and parametric distributions please consult a Statistics textbook such as this one.

Linear Models

We have implicitly been using a very simple linear models. We have been trying to predict a fixed parameter by taking samples. For example we have tried to estimate the probability of tossing a coin and observing a head, the proportion of blue beads in a jar, the average height of a population, and the difference in the proportion of votes two candidates, for example Obama and Romney , will receive. Here we will us general notation and represent the parameter we are trying to estimate with θ.

We then obtain draws from a sampling model. We will denote the observed values with Y and use indexes to denote the fact that we make, say, N observations:

$$Y_i = \theta + \varepsilon_i, i = 1, \dots N$$

Each observation has an *error* term ε. We assume that the expected value of the error is 0 and very often assume that the standard deviation is constant σ.

Let's consider some simple examples

Assessment This is a contrived example, but it will help us understand how we think about models and can use them in many situations.

If we are betting on black on roulette, we can let Y_i be either -1 or 1. Now what is θ and what is the distribution of our error? What is the standard deviation of the errors? Hint: the expected value of the error must be 0.

Assessment Let's consider the example of a demographer trying to estimate the average height of a population. We use the heights stored in R:

```
data("father.son", package="UsingR")
y <- father.son$sheight
```

If we now take a sample

```
set.seed(1)
Y <- sample(y, 25, replace = TRUE)
```

What are θ, N, and ε_1 ?

Assessment Note that in this example each individual in the population has an *error*. It may sound a bit strange but, if you are 1 inch taller than the average height, we call it an "error" of 1 inch. Compute all the errors for the population and explore the distribution. What is the standard deviation of this distribution?

A) Exactly normal with σ of about 3 inches

B) Approximately normal with σ of about 3 inches. Tails are slightly larger.

C) These are not averages or sums so it is not approximately normal.

D) Approximately normal with σ of about 9 inches. Tails are slightly larger.

Modeling Poll Results

Let's start by looking at our guess-the-percent-of-blue-beads competition data. Let's use the code from the previous section:

```
library(readr)
library(dplyr)

filename <- "https://raw.githubusercontent.com/datasciencelabs/data/master/blue-bead-com
tab <- read_csv(filename)
names(tab)<-c("timestamp", "name", "estimate","poll_sample_size","ci")
tab <- mutate(tab,estimate=ifelse(estimate<1, estimate*100, estimate)) %>%
  filter(estimate>20)
```

A total of 27 people used a sample size of about 100.

```
filter( tab, abs(poll_sample_size-100)<51) %>% nrow
```

```
## [1] 27
```

Very important: do not confuse the poll sample size with the number of observations in our model. Here each poll is an observation: $N = 27$ while the poll sizes are about 100.

Let's consider only those polls:

```
tab <- filter(tab, abs(poll_sample_size-100)<51)
```

So we can write a model:

$$Y_i = \theta + \varepsilon_i, i = 1, \ldots, N$$

with $N = 22$.

Assessment Using the CLT theory we have learned, how do we model ε, what is the expected value and standard deviation of ε?

Assessment We already revealed that $\theta = 0.534$. Use this information and data exploration to check if the model assumptions make sense.

Estimating parameters

Earlier we showed how for this particular case we can aggregate results using what amounts to a weighted average. For this particular case, where we have a defensible model we can create an estimate of θ using the standard approach to fitting linear models.

The standard approach is to use the value that that minimizes the least squares equation:

$$\sum_{i=1}^{N}(Y_i - \theta)^2$$

This is called the *least squares estiamte* (LSE). In this case is easy to show, with Calculus, that it is the sample average:

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} Y_i = \bar{Y}$$

Because this is a sample average from a sampling model we know its expectation is θ and it's standard error is $\sigma/\sqrt{N}$ with N the number of observations and σ the standard deviation of the distribution of ε. But what is σ?

In this particular case statistical theory tells us that σ should be:

$$\frac{\sqrt{\theta(1-\theta)}}{\sqrt{100}}$$

And this is extremely useful when we just have one poll. However, with many polls we can also use the data to estimate σ. The typical strategy is to use the the sample standard deviation:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$

We can compare this empirical approach to the more theoretical approach we have been using:

```
theta <- mean(y)
sd(Y) ##compared to
```

```
## [1] 3.530405
```

```
sqrt(theta*(1-theta))/sqrt(100) ##or to
```

```
## Warning in sqrt(theta * (1 - theta)): NaNs produced
```

```
## [1] NaN
```

```
theta_hat <- mean(Y)
sqrt(theta_hat*(1-theta_hat))/sqrt(100) ##or to
```

```
## Warning in sqrt(theta_hat * (1 - theta_hat)): NaNs produced
```

```
## [1] NaN
```

Even if we didn't know these data came from individual polls we can still see that our model works rather well. Here is a confidence interval build with just the data, not using sampling theory for the polls:

```
theta_hat <- mean(Y)
s <- sd(Y)
## confidence interval:
theta_hat + c(-1,1)*qnorm(0.975)*s/sqrt(length(Y))
```

```
## [1] 67.40331 70.17110
```

Here it is slightly more appropriate to use the t-distribution approximation:

```
## or using the t-distribution
theta_hat + c(-1,1)*qt(0.975, df=length(Y)-1)*s/sqrt(length(Y))
```

```
## [1] 67.32993 70.24449
```

Note that this is a somewhat artificial experiment. The data were generated from a model we constructed. Let's see what happens when we try these models "in the wild".

The 2008 Presidential Elections

We are going to use the 2008 presidential election as an example. To obtain the data we are going to scrape this web site.

Scrapping data from URLs

Data often appears in tables on the web. These are typically HTML pages. From looking at the code for these pages you can see that tables follow a specific format which implies we can write computer programs to extract the data. For *markdown languages* (ML) such as HTML the `XML` package is quite useful. The function `readHTMLTable` specifically targets tables.

```
library(XML)
theurl <- paste0("http://www.pollster.com/08USPresGEMvO-2.html")
html_tables <- readHTMLTable(theurl,stringsAsFactors=FALSE)
```

This produces a list and the components are tables. If we look at this object we see that the first and only table is the one we are after:

```
tab <- html_tables[[1]]
```

We want to have access to the dates of the polls. Take a look at how the save dates:

```
head(tab$Dates)
```

```
## [1] "11/3/08"   "11/2-3/08" "11/1-3/08" "11/1-3/08" "11/1-3/08" "11/1-3/08"
```

We are going to need some serious wrangling here. We will learn a couple of useful functions first.

Wrangling Dates

The `tidyr`, `stringr` and `lubridate` packages include powerful tools for dealing with dates.

```
library(dplyr)
library(tidyr)
library(stringr)
library(lubridate)
```

Let's go step-by-step:

```
##use separate to split start and end dates
### we convert it to a dplyr table:
tab <- tbl_df(tab)
tab <- tab %>% separate(col=Dates, into=c("start_date","end_date"), sep="-",
                        fill="right")
select(tab, start_date, end_date)
```

```
## Source: local data frame [543 x 2]
##
##    start_date end_date
##         (chr)    (chr)
## 1     11/3/08       NA
```

```
## 2        11/2     3/08
## 3        11/1     3/08
## 4        11/1     3/08
## 5        11/1     3/08
## 6        11/1     3/08
## 7       10/31  11/3/08
## 8       10/30  11/3/08
## 9     11/2/08       NA
## 10       11/1     2/08
## ..        ...      ...
```

```
## if no end_data it means only one day was provided so use that as end as well
tab <- tab %>% mutate(end_date = ifelse(is.na(end_date), start_date, end_date))
select(tab, start_date, end_date)
```

```
## Source: local data frame [543 x 2]
##
##    start_date end_date
##         (chr)    (chr)
## 1     11/3/08  11/3/08
## 2        11/2     3/08
## 3        11/1     3/08
## 4        11/1     3/08
## 5        11/1     3/08
## 6        11/1     3/08
## 7       10/31  11/3/08
## 8       10/30  11/3/08
## 9     11/2/08  11/2/08
## 10       11/1     2/08
## ..        ...      ...
```

```
## no use seprate again to get month, day and year for start_date
tab <- tab %>% separate(start_date, c("smonth", "sday", "syear"), sep = "/",
                        convert = TRUE, fill = "right")
select(tab, smonth:syear, end_date)
```

```
## Source: local data frame [543 x 4]
##
##    smonth  sday syear end_date
##     (int) (int) (int)    (chr)
## 1      11     3     8  11/3/08
## 2      11     2    NA     3/08
## 3      11     1    NA     3/08
```

```
## 4      11     1    NA     3/08
## 5      11     1    NA     3/08
## 6      11     1    NA     3/08
## 7      10    31    NA  11/3/08
## 8      10    30    NA  11/3/08
## 9      11     2     8  11/2/08
## 10     11     1    NA     2/08
## ..    ...   ...   ...      ...
```

```
### if end data has only 1 / then it is missing month, add it
tab <- tab %>% mutate(end_date = ifelse(str_count(end_date, "/") == 1,
                                        paste(smonth, end_date, sep = "/"), end_date))
select(tab, smonth:syear, end_date)
```

```
## Source: local data frame [543 x 4]
##
##    smonth  sday syear end_date
##     (int) (int) (int)    (chr)
## 1      11     3     8  11/3/08
## 2      11     2    NA  11/3/08
## 3      11     1    NA  11/3/08
## 4      11     1    NA  11/3/08
## 5      11     1    NA  11/3/08
## 6      11     1    NA  11/3/08
## 7      10    31    NA  11/3/08
## 8      10    30    NA  11/3/08
## 9      11     2     8  11/2/08
## 10     11     1    NA  11/2/08
## ..    ...   ...   ...      ...
```

```
## now use lubridate function mdy to conver to date
tab <- tab %>% mutate(end_date = mdy(end_date))
select(tab, smonth:syear, end_date)
```

```
## Source: local data frame [543 x 4]
##
##    smonth  sday syear   end_date
##     (int) (int) (int)     (time)
## 1      11     3     8 2008-11-03
## 2      11     2    NA 2008-11-03
## 3      11     1    NA 2008-11-03
## 4      11     1    NA 2008-11-03
## 5      11     1    NA 2008-11-03
```

```
## 6      11     1    NA 2008-11-03
## 7      10    31    NA 2008-11-03
## 8      10    30    NA 2008-11-03
## 9      11     2     8 2008-11-02
## 10     11     1    NA 2008-11-02
## ..    ...   ...   ...        ...
```

```
## add 2000 to year since it is currently 7 or 8
tab <- tab %>% mutate(syear = ifelse(is.na(syear), year(end_date), syear + 2000))
select(tab, smonth:syear, end_date)
```

```
## Source: local data frame [543 x 4]
##
##    smonth  sday syear   end_date
##     (int) (int) (dbl)     (time)
## 1      11     3  2008 2008-11-03
## 2      11     2  2008 2008-11-03
## 3      11     1  2008 2008-11-03
## 4      11     1  2008 2008-11-03
## 5      11     1  2008 2008-11-03
## 6      11     1  2008 2008-11-03
## 7      10    31  2008 2008-11-03
## 8      10    30  2008 2008-11-03
## 9      11     2  2008 2008-11-02
## 10     11     1  2008 2008-11-02
## ..    ...   ...   ...        ...
```

```
## now use unite to create a m/d/y string
tab <- tab %>% unite(start_date, smonth, sday, syear)
select(tab, start_date, end_date)
```

```
## Source: local data frame [543 x 2]
##
##    start_date   end_date
##         (chr)     (time)
## 1   11_3_2008 2008-11-03
## 2   11_2_2008 2008-11-03
## 3   11_1_2008 2008-11-03
## 4   11_1_2008 2008-11-03
## 5   11_1_2008 2008-11-03
## 6   11_1_2008 2008-11-03
## 7  10_31_2008 2008-11-03
## 8  10_30_2008 2008-11-03
```

```
## 9   11_2_2008 2008-11-02
## 10  11_1_2008 2008-11-02
## ..        ...        ...
```

```
## covert it to date class
tab <- tab %>% mutate(start_date = mdy(start_date))
select(tab, start_date, end_date)
```

```
## Source: local data frame [543 x 2]
##
##    start_date   end_date
##        (time)     (time)
## 1  2008-11-03 2008-11-03
## 2  2008-11-02 2008-11-03
## 3  2008-11-01 2008-11-03
## 4  2008-11-01 2008-11-03
## 5  2008-11-01 2008-11-03
## 6  2008-11-01 2008-11-03
## 7  2008-10-31 2008-11-03
## 8  2008-10-30 2008-11-03
## 9  2008-11-02 2008-11-02
## 10 2008-11-01 2008-11-02
## ..        ...        ...
```

Note: we don't have to go step by step. Instead rewrite the above as a series of pipes.

Extracting population size and type of poll If we wanted to know the sample size of the polls we look here:

```
head(tab$`N/Pop`)
```

```
## [1] "804 LV"  "400 LV"  "1100 LV" "981 LV"  "3000 LV" "1200 LV"
```

This column combines the sample size with the type population used by the poll. For example, LV means Likely Voters. We can use split again.

```
tab <- separate(tab, `N/Pop`, into=c("N","population_type"), sep="\ ", convert=TRUE, fil
```

Adding Columns We will add a couple more columns for what we are doing: the Obama - McCAin difference, the days left before the election and the weeks left before the election:

```
tab <- mutate(tab, Obama = as.numeric(Obama)/100,
              McCain=as.numeric(McCain)/100,
              diff = Obama - McCain,
              day=as.numeric(start_date - mdy("11/04/2008")),
              week = as.numeric(floor(day/7)))
```

Modeling Poll Results (continued)

Now consider all the polls of sample size 800. There are

```
filter(tab, N==800) %>% nrow
```

```
## [1] 22
```

of these. So we can write a model:

$$Y_i = \theta + \varepsilon_i, i = 1, \ldots, N$$

with $N = 22$.

Assessment Using the CLT theory we have learned, how do we model ε, what is the expected value and standard deviation of ε?

Assessment Use data exploration to check if the model assumptions make sense. Note that we have the benefit of hindsight. The election night result was $\theta = 52.9 - 45.7 = 7.2$.

A. The data are approximately normal with the expected value and standard error as predicted.

B. The expected value and standard error as predicted, but the data is not normal

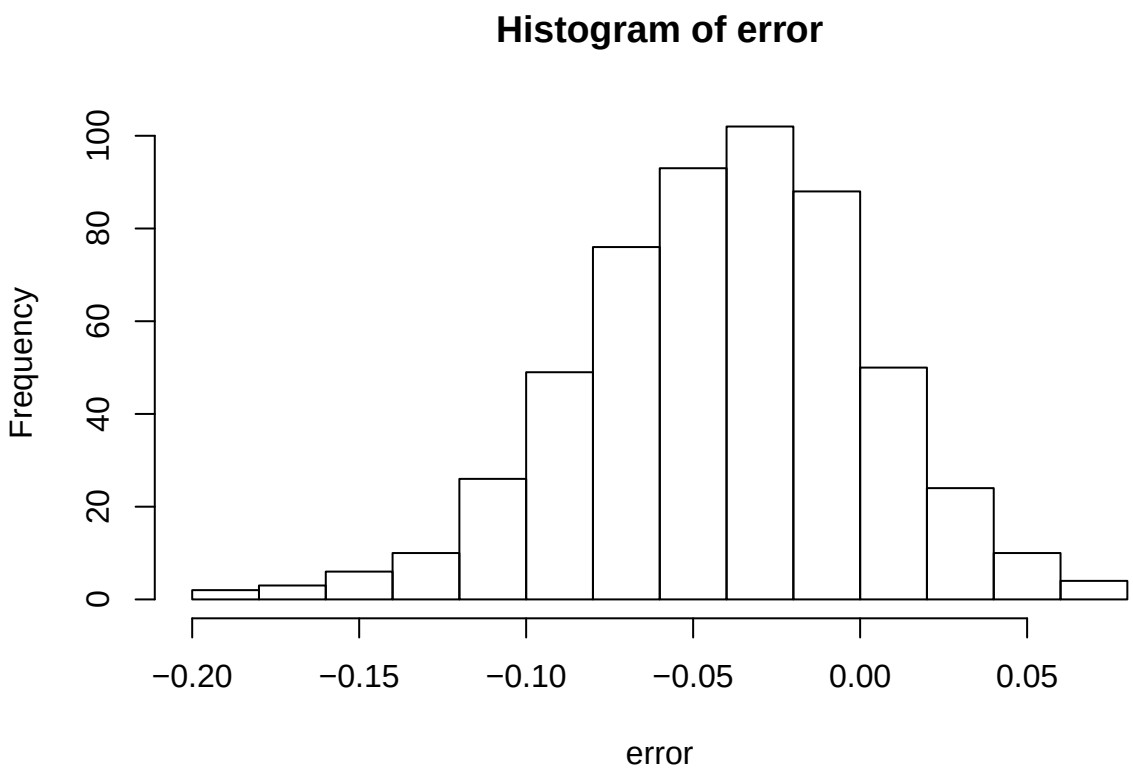
C. The data look approximately normal but the expected and standard error are not what the theory predicts

D. I have no idea what to do.

Explaining variance

In general we can see that the errors are not centered at 0 and have a much larger variability than expected:

```r
error <- tab$diff - 7.2/100
hist(error)
```



What could explain this variability?

The first thing that comes to mind is that we polls are taken for over a year and the θ six months before the election might be different.

Assessment Use ggplot to make a plot of the estimated difference versus day. What do you observe? Just form looking at the plot what would you say is that standard deviation of θ if we let it change with time?

The SD is at least 5%

Assessment Take a look at polls that happened in 2008.

There seems to be clear time effect. We could amend the model to be

$$Y_{t,i} = \theta + w_t + \varepsilon_{t,i}$$

We now have two indexes t denoting week and i an index for the i-th poll during week t.

Smoothing

To estimate the model above, we could go week by week and estimating $\theta_t = \theta + w_t$ separately. Our model for week t would be

$$Y_{t,i} = \theta_t + \varepsilon_{t,i}$$

and we can use the same approach as before. We can compute the estimates using the `group_by` and `summarize` approach.

`group_by` and `summarize` A very common operation performed in data analysis is to stratify data into groups and then obtain a summary statistic, such a mean and standard deviation, from each group. Here we want to group by week. So we simply do the following

```
group_by(tab, week)
```

```
## Source: local data frame [543 x 15]
## Groups: week [72]
##
##                          Pollster start_date   end_date     N
##                             (chr)     (time)     (time) (int)
## 1                  Marist College 2008-11-03 2008-11-03   804
## 2              GWU (Lake/Tarrance) 2008-11-02 2008-11-03   400
## 3  DailyKos.com (D)/Research 2000 2008-11-01 2008-11-03  1100
## 4                         IBD/TIPP 2008-11-01 2008-11-03   981
## 5                        Rasmussen 2008-11-01 2008-11-03  3000
## 6                              ARG 2008-11-01 2008-11-03  1200
## 7          Reuters/ C-SPAN/ Zogby 2008-10-31 2008-11-03  1226
## 8              Harris Interactive 2008-10-30 2008-11-03  3946
## 9                  Marist College 2008-11-02 2008-11-02   635
## 10                        NBC/WSJ 2008-11-01 2008-11-02    NA
## ..                            ...        ...        ...   ...
## Variables not shown: population_type (chr), McCain (dbl), Obama (dbl),
##   Barr (chr), Nader (chr), Other (chr), Undecided (chr), Margin (chr),
##   diff (dbl), day (dbl), week (dbl)
```

You can see that the 543 polls have been grouped into 72. Once this is done, if you call the `summarize` function it applies a summary to each group:

```
group_by(tab, week) %>% summarize(avg=mean(diff))
```

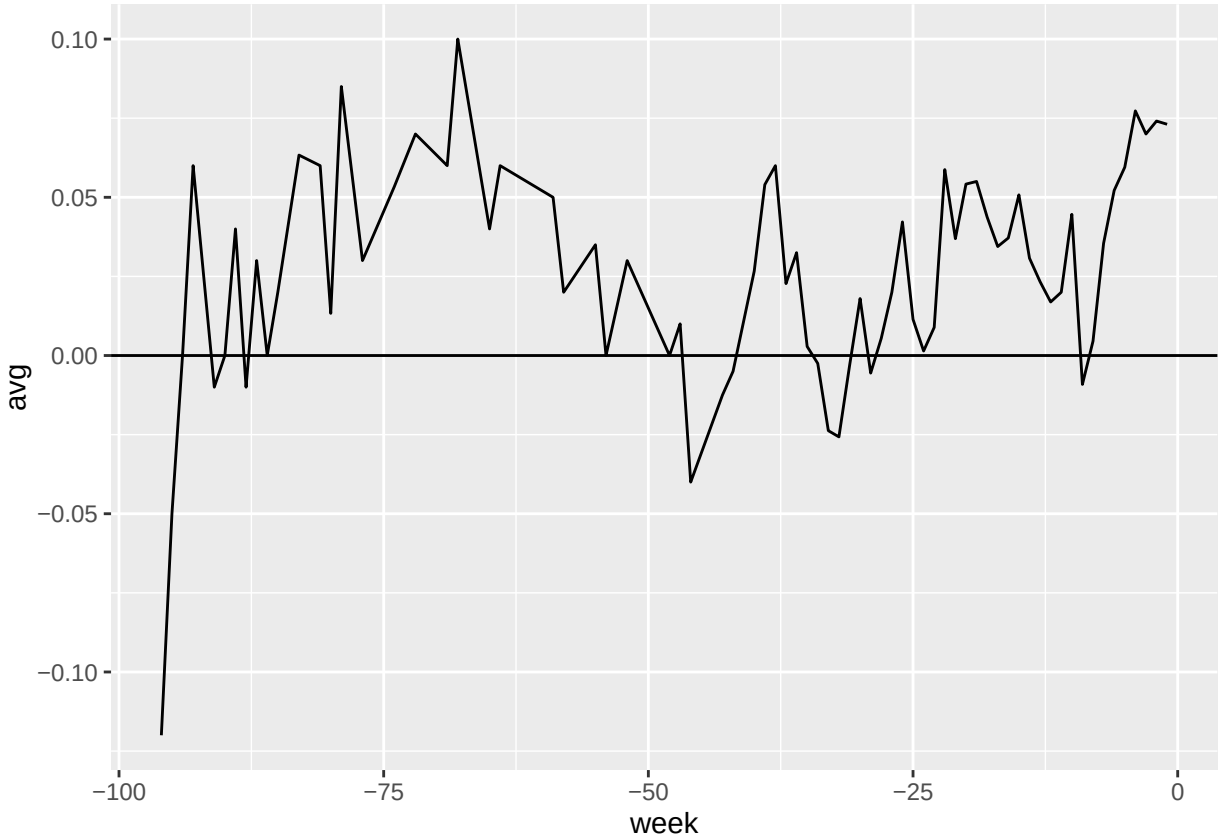
```
## Source: local data frame [72 x 2]
```

```
##
##     week           avg
##    (dbl)         (dbl)
## 1    -96 -1.200000e-01
## 2    -95 -5.000000e-02
## 3    -94 -1.387779e-17
## 4    -93  6.000000e-02
## 5    -91 -1.000000e-02
## 6    -90  0.000000e+00
## 7    -89  4.000000e-02
## 8    -88 -1.000000e-02
## 9    -87  3.000000e-02
## 10   -86  0.000000e+00
## ..   ...           ...
```

We can make a quick plot

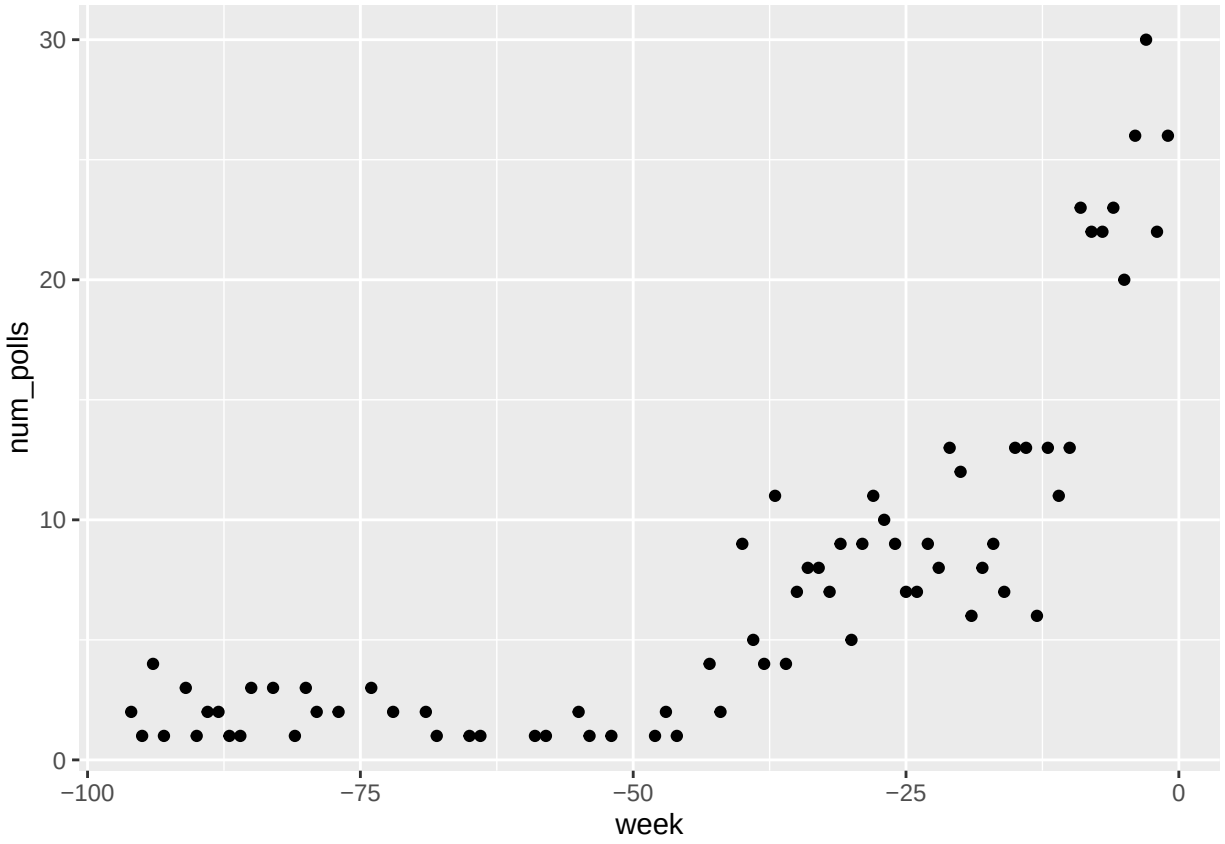
```
library(ggplot2)

group_by(tab, week) %>% summarize(avg=mean(diff)) %>%
  ggplot(aes(week, avg)) + geom_line() +
  geom_hline(aes(yintercept=0))
```



Note that the number of polls per week varies:

```
group_by(tab, week) %>% summarize(num_polls=n()) %>%
  ggplot(aes(week, num_polls)) + geom_point()
```



Supposed we only want to consider weeks in which we had 10 or more polls. We can easily group and count like this. But now the table is summarized. We don't need to summarize it. We can `ungroup` at the end.

```
group_by(tab, week) %>% mutate(num_polls=n()) %>% select(Pollster, num_polls) %>% ungrou
```

```
## Source: local data frame [543 x 3]
##
##     week                         Pollster num_polls
##    (dbl)                            (chr)     (int)
## 1     -1                  Marist College        26
## 2     -1              GWU (Lake/Tarrance)        26
## 3     -1 DailyKos.com (D)/Research 2000        26
## 4     -1                          IBD/TIPP        26
## 5     -1                         Rasmussen        26
## 6     -1                               ARG        26
## 7     -1          Reuters/ C-SPAN/ Zogby        26
```

```
## 8     -1             Harris Interactive        26
## 9     -1                 Marist College        26
## 10    -1                        NBC/WSJ        26
## ..   ...                            ...       ...
```

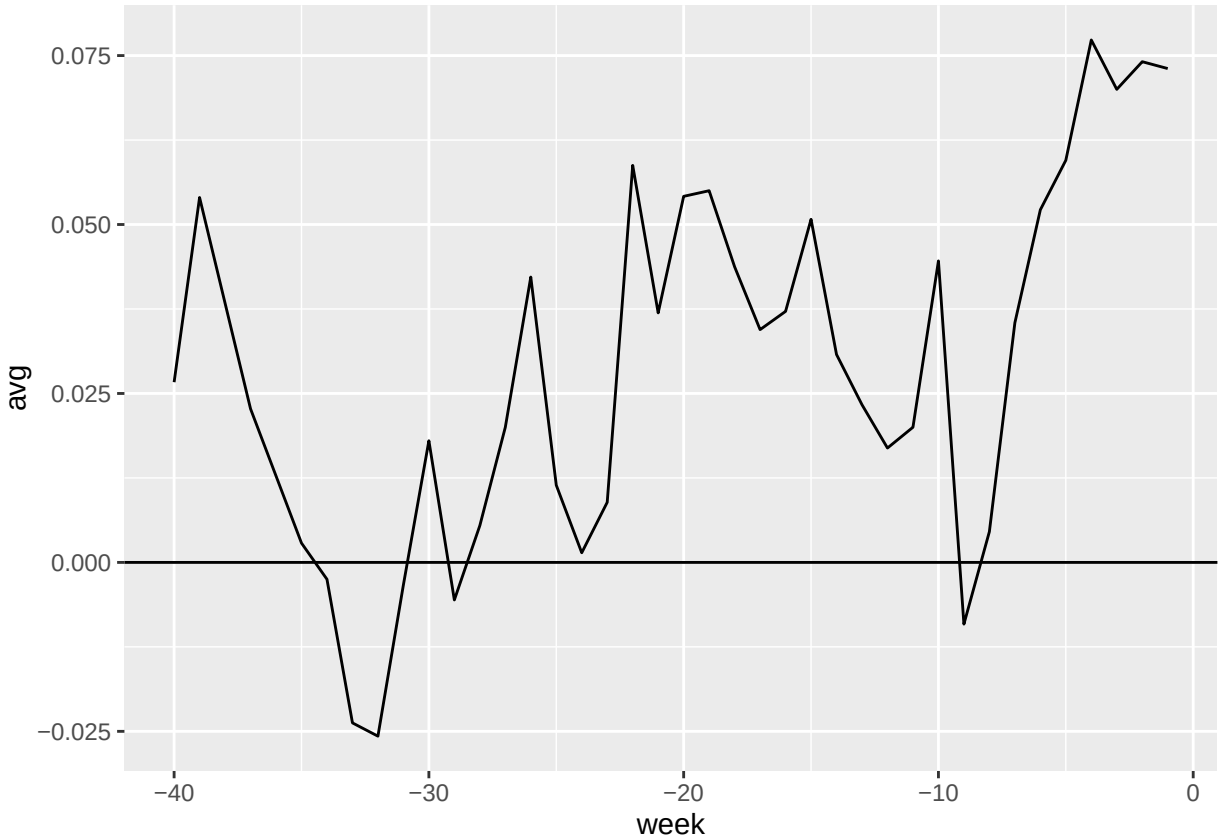
Which means we can use this to filter.

```
group_by(tab, week) %>% mutate(num_polls=n()) %>% select(Pollster, num_polls) %>% filte
```

```
## [1] 334
```

To make the plot for just these we can use:

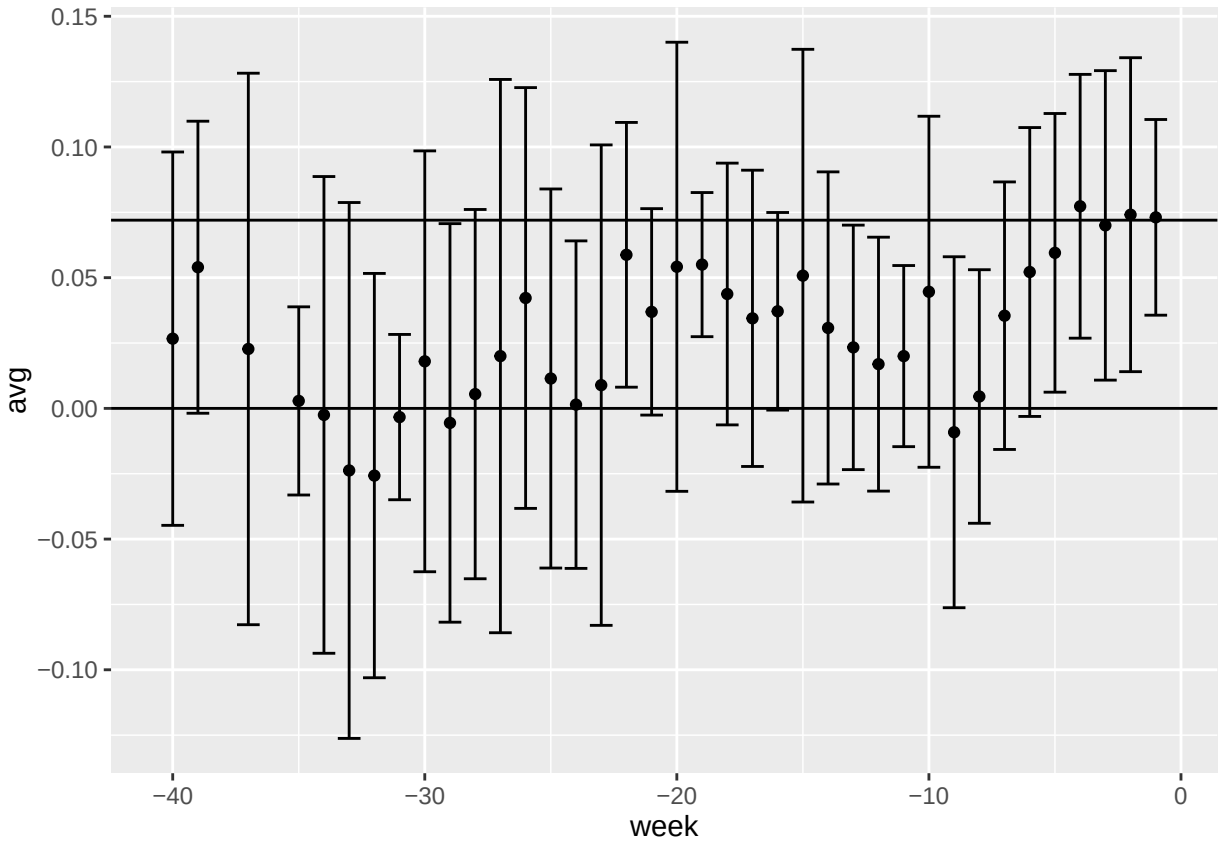
```
group_by(tab, week) %>% mutate(num_polls=n()) %>%
  filter(num_polls>=5) %>%
  summarize(avg=mean(diff)) %>%
  ggplot(aes(week, avg)) + geom_line() +
  geom_hline(aes(yintercept=0))
```



Assessment

Make the same plot to add bars that show 2 SD above and below the average. Answer the questions below.

```
group_by(tab, week) %>% mutate(num_polls=n()) %>%
  filter(num_polls>=5) %>%
  summarize(avg=mean(diff) , sd=sd(diff)) %>%
  ggplot(aes(week, avg, ymin=avg-2*sd, ymax=avg+2*sd)) +
  geom_point() + geom_errorbar() +
  geom_hline(aes(yintercept=0))+
  geom_hline(aes(yintercept=0.072))
```



We saw earlier that as the elections got closer many more plots are conducted. But the plot above does not show the standard deviations decreasing. At least not at the rate of $1/\sqrt{\text{number of polls}}$. Why is this?

A. The standard deviation is decreasing

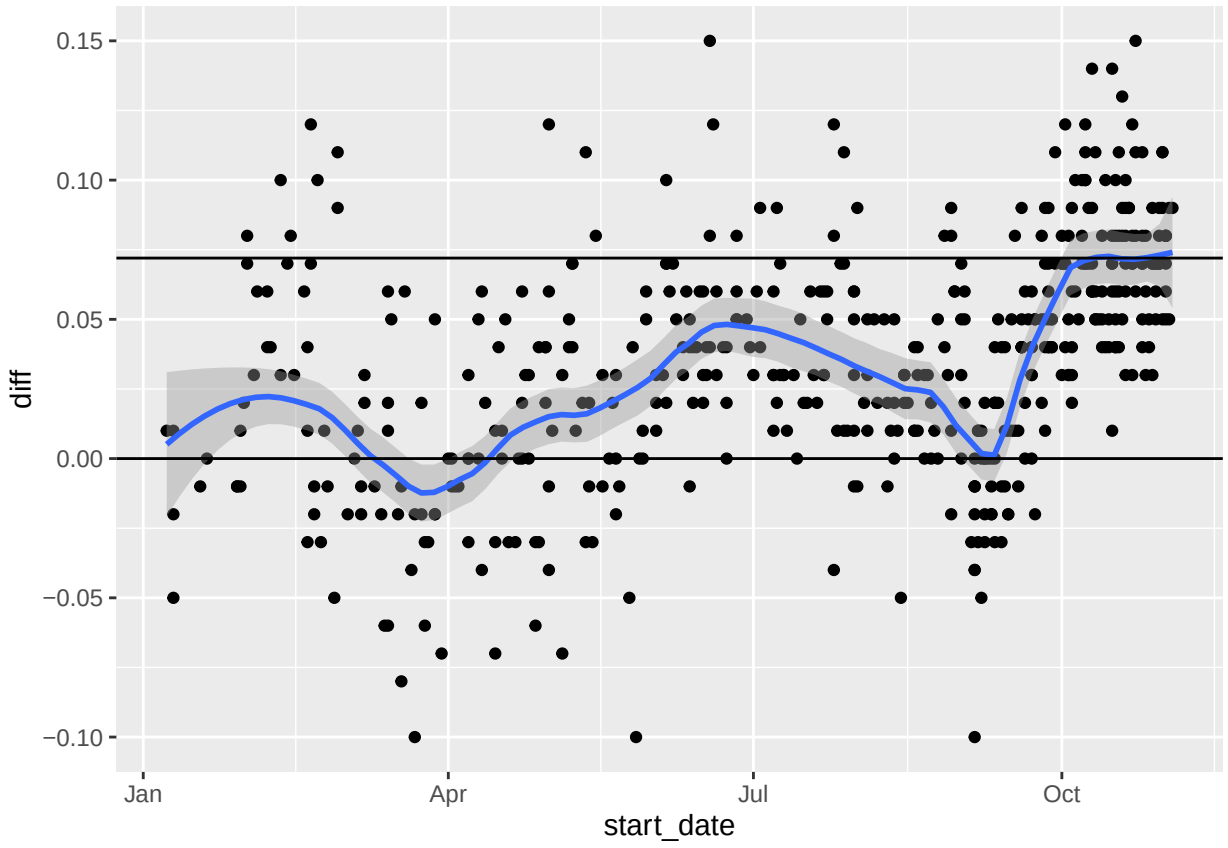
B. The standard deviation relates to the SE of each poll, which depends on poll size not number of polls

C. There is a bug in the code

D. I have no idea.

A similar plot to this can be generated with

```
filter(tab, start_date>"2008-01-01") %>% ggplot(aes(start_date, diff)) + geom_point() +
   geom_hline(aes(yintercept=0.072))
```



House Effect We have seen that there is a strong weekly effect. We saw how we could control it by, for example, stratifying the analysis. We noted that the last couple of weeks provides the original model may be "useful".

$$Y_i = \theta + \varepsilon_i$$

Now if we restrict ourselves to polls with sample sizes larger than 2,000, the the theory tells us that the standard error for the estimate of the percentage for Obama is $\hat{p}$ (notation from previous section) is

$$\sqrt{p(1-p)}/\sqrt{2000}$$

Here we are estimating the difference θ. The estimated is approximated by $\hat{p}-(1-\hat{p}) = 2\hat{p}-1$. This implies that the standard error is $2\sqrt{p(1-p)}/\sqrt{2000}$.

Because each Y_i a separate poll then the standard deviation of ε should be about $2\sqrt{p(1-p)}/\sqrt{2000}$:

```
2* sqrt(0.5*0.5) / sqrt(2000)
```

```
## [1] 0.02236068
```

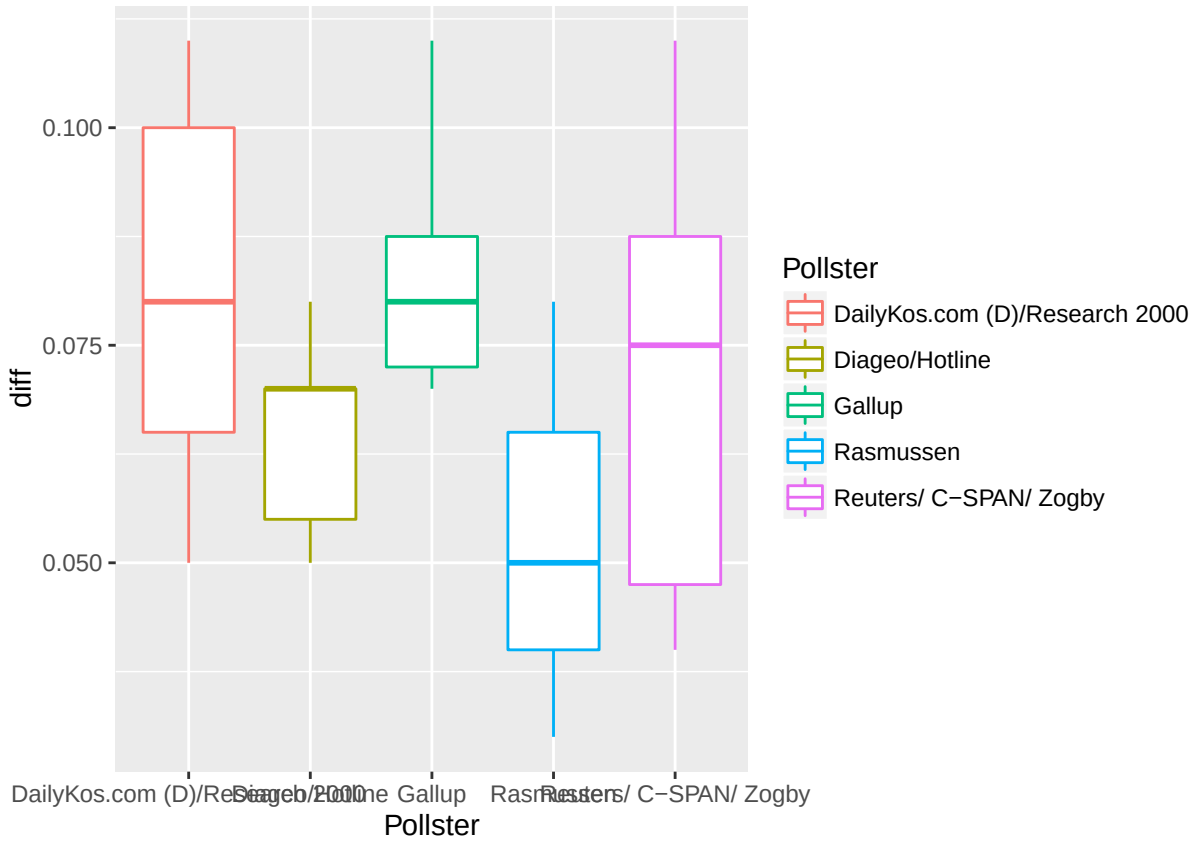
But the observed variance is slightly larger:

```
tab %>% filter( N>2000 & week > -4) %>% summarize(sd(diff))
```

```
## Source: local data frame [1 x 1]
##
##     sd(diff)
##        (dbl)
## 1 0.02624669
```

This could be due to what is referred to as the *house effect*. Note in the plot below how there appears to be a Pollster effect:

```
tab %>% filter(week > -4) %>% group_by(Pollster) %>% filter(n()>4) %>%
  ggplot(aes(Pollster, diff , col=Pollster)) + geom_boxplot()
```



An improved model will include a pollster effect

$$Y_{i,j} = \theta + p_j + \varepsilon_{i,j}$$

with p_j a pollster effect.

In more formal statistics classes you can learn about analysis of variance which does confirm strong week effect and small, but statistical significant, house effect:

```
tab2 <- filter(tab, start_date > "2008-01-01") %>% group_by(Pollster) %>% filter(n() > 1
fit <- lm(diff ~ week + Pollster, data=tab2)
summary( aov(fit) )
```

```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## week          1 0.1458 0.14579 122.478 < 2e-16 ***
## Pollster     14 0.0908 0.00649   5.448 1.9e-09 ***
## Residuals   342 0.4071 0.00119
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```