

# 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  $\theta$ .

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  $\varepsilon$ . We assume that the expected value of the error is 0 and very often assume that the standard deviation is constant  $\sigma$ .

Let's consider some simple examples

**Assessment 1:** <http://goo.gl/forms/xN35fstJky> 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  $\theta$  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.

Answer: Our parameter is  $\theta = -1/19$  and our errors are  $1 - \theta$  with probability  $9/19$  and  $-1 - \theta$  with probability  $10/19$ .

Double check that the expected value is 0:

```
20/19 * 9/19 - 18/19*10/19
```

```
## [1] 0
```

The standard deviation we know is  $2\sqrt{9/19 \times 10/19}$ . We can double check with a Monte Carlo

```
B <- 10^5
theta = -1/19
Y <- sample( c(-1,1), B, replace=TRUE, prob=c(10/19, 9/19))
error <- Y - theta
prop.table(table(error))
```

```
## error
## -0.947368421052632   1.05263157894737
##               0.52861           0.47139
```

```
sqrt(mean(error^2))
```

```
## [1] 0.9983721
```

```
2*sqrt(10/19*9/19)
```

```
## [1] 0.998614
```

**Assessment 2:** <http://goo.gl/forms/XLQQh69MUO> 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  $\theta$ ,  $N$ , and  $\varepsilon_1$  ?

```
## N is
length(Y)

## [1] 25

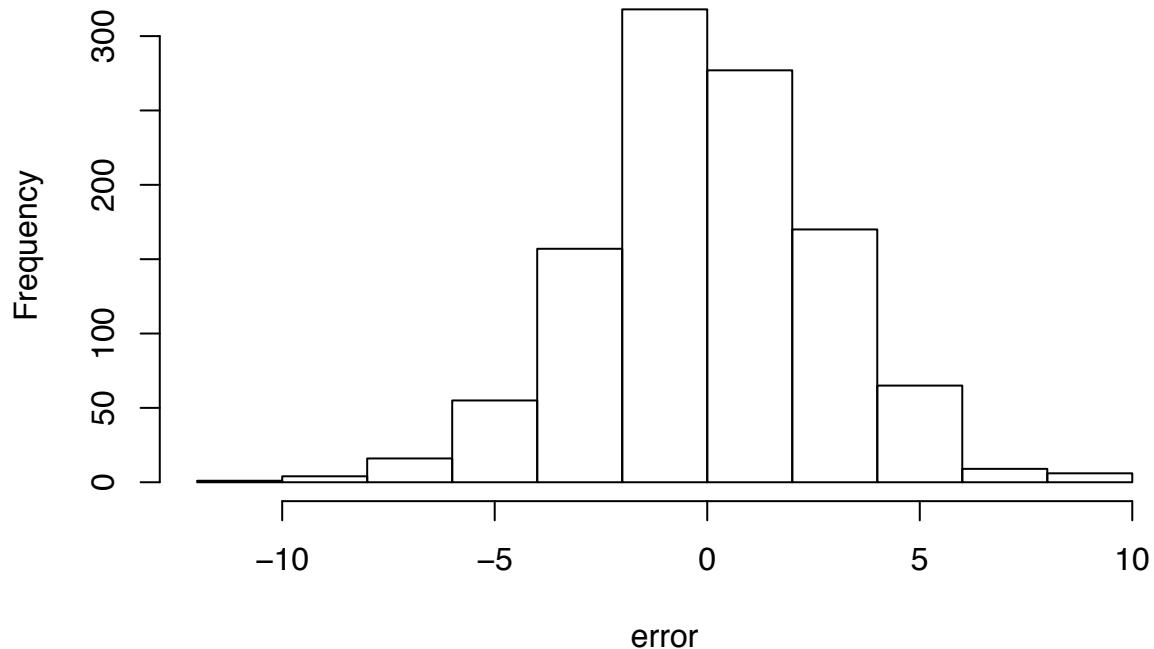
## theta is
theta <- mean(y)
## the first error is
error_1 <- Y[1] - theta
```

**Assessment 3:** <http://goo.gl/forms/CfamwJBctA> 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  $\sigma$  of about 3 inches
- B) Approximately normal with  $\sigma$  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  $\sigma$  of about 9 inches. Tails are slightly larger.

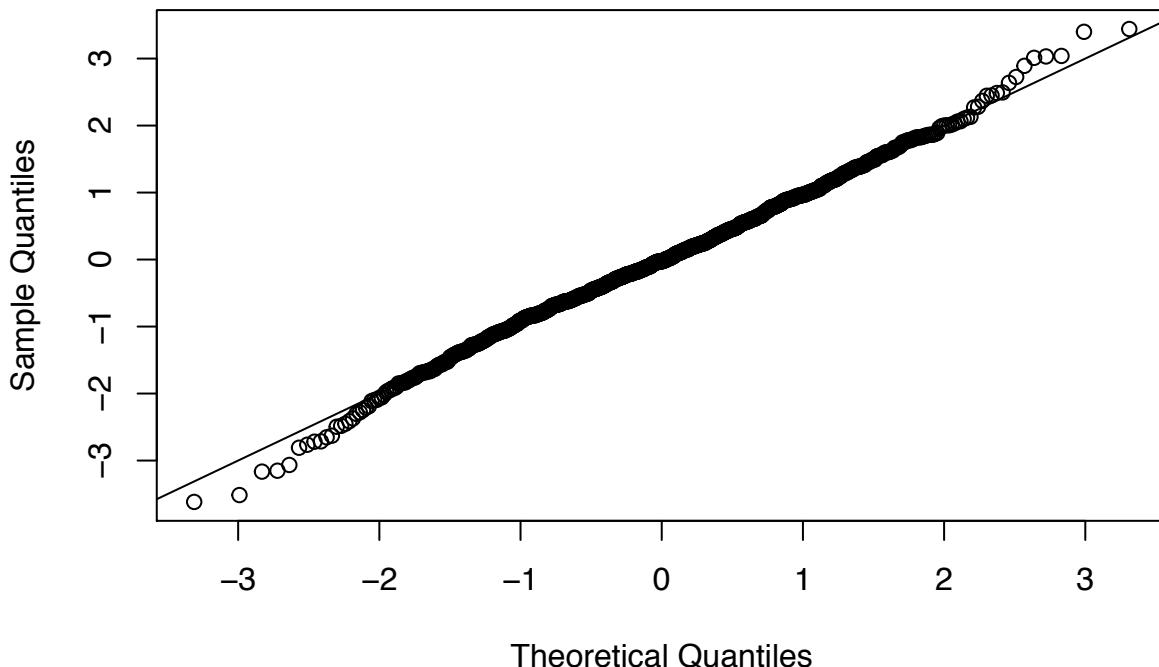
```
error <- y - mean(y)
hist(error)
```

## Histogram of error



```
##the standard error is:  
sigma <- sqrt( mean( (error)^2 ) )  
qqnorm(error/sigma)  
abline(0,1)
```

## Normal Q-Q Plot



## 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/datasets/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:

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

with  $N = 22$ .

**Assessment 4:** <http://goo.gl/forms/iyXi76a4b0> Using the CLT theory we have learned, how do we model  $\varepsilon$ , what is the expected value and standard deviation of  $\varepsilon$ ?

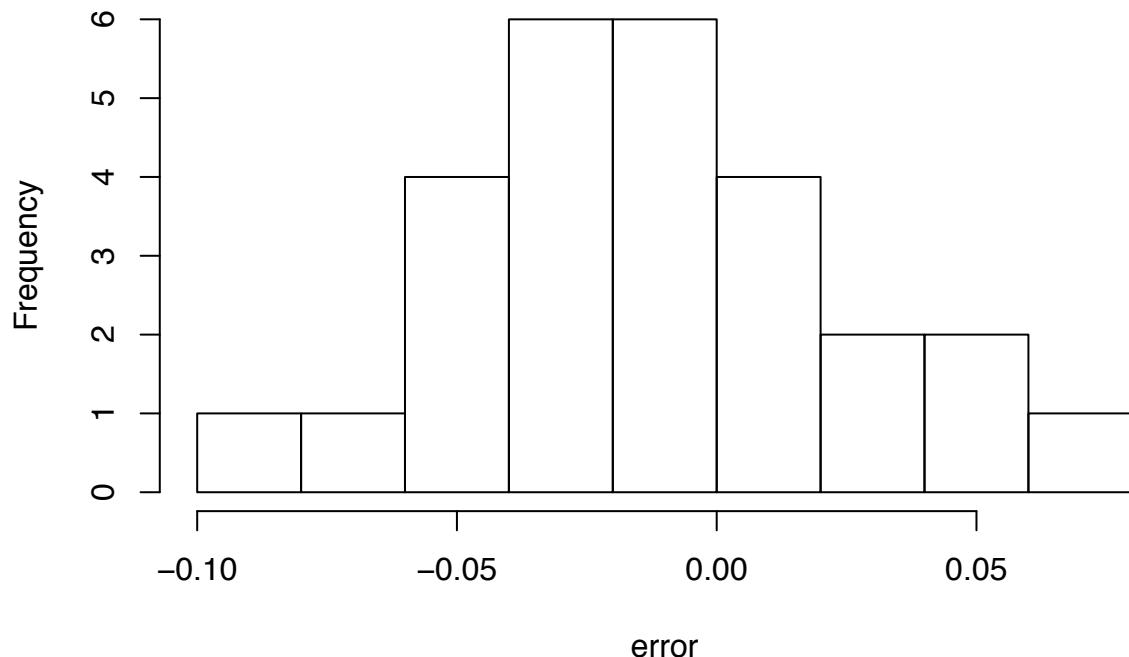
Answer: According to CLT

$$\varepsilon \sim \text{Normal} \left( 0, \frac{\sqrt{\theta(1-\theta)}}{\sqrt{100}} \right)$$

**Assessment 5:** <http://goo.gl/forms/0sYp0gFnxZ> We already revealed that  $\theta = 0.534$ . Use this information and data exploration to check if the model assumptions make sense.

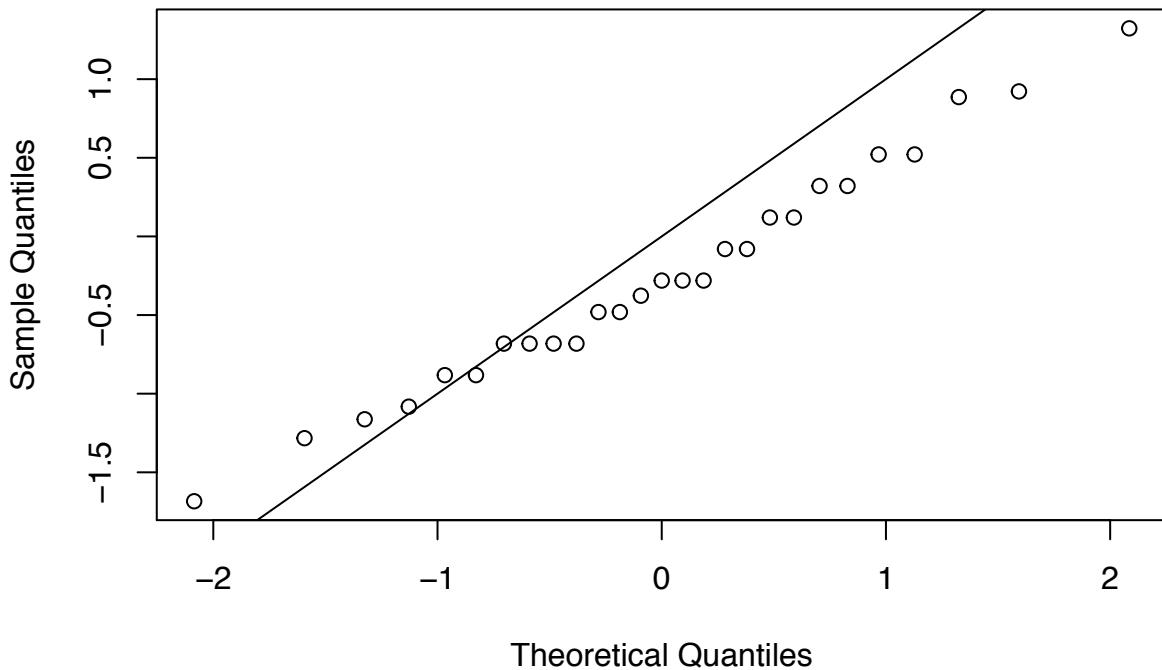
```
theta <- 0.534
Y <- tab$estimate/100
error <- Y - theta
hist(error)
```

Histogram of error



```
se <- sqrt(theta*(1-theta))/sqrt(100)
qqnorm(error/se)
abline(0,1)
```

## Normal Q-Q Plot



### 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  $\theta$  using the standard approach to fitting linear models.

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

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

This is called the *least squares estimate* (LSE). In this case it 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  $\theta$  and its standard error is  $\sigma/\sqrt{N}$  with  $N$  the number of observations and  $\sigma$  the standard deviation of the distribution of  $\varepsilon$ . But what is  $\sigma$ ?

In this particular case statistical theory tells us that  $\sigma$  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  $\sigma$ . The typical strategy is to use 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] 0.03611641

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

## [1] 0.04995551
```

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] 0.5074660 0.5347118
```

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] 0.5068017 0.5353761
```

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.

### Scraping 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)
```

```
## Loading required package: methods
```

```
theurl <- paste0("http://www.pollster.com/08USPresGEMv0-2.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 `tidyverse`, `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
## ...    ...    ...

## convert 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, fill=NA)
```

**Adding Columns** We will add a couple more columns for what we are doing: the Obama - McCains 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:

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

with  $N = 22$ .

**Assessment 6:** <http://goo.gl/forms/BYmGozyUYX> Using the CLT theory we have learned, how do we model  $\varepsilon$ , what is the expected value and standard deviation of  $\varepsilon$ ?

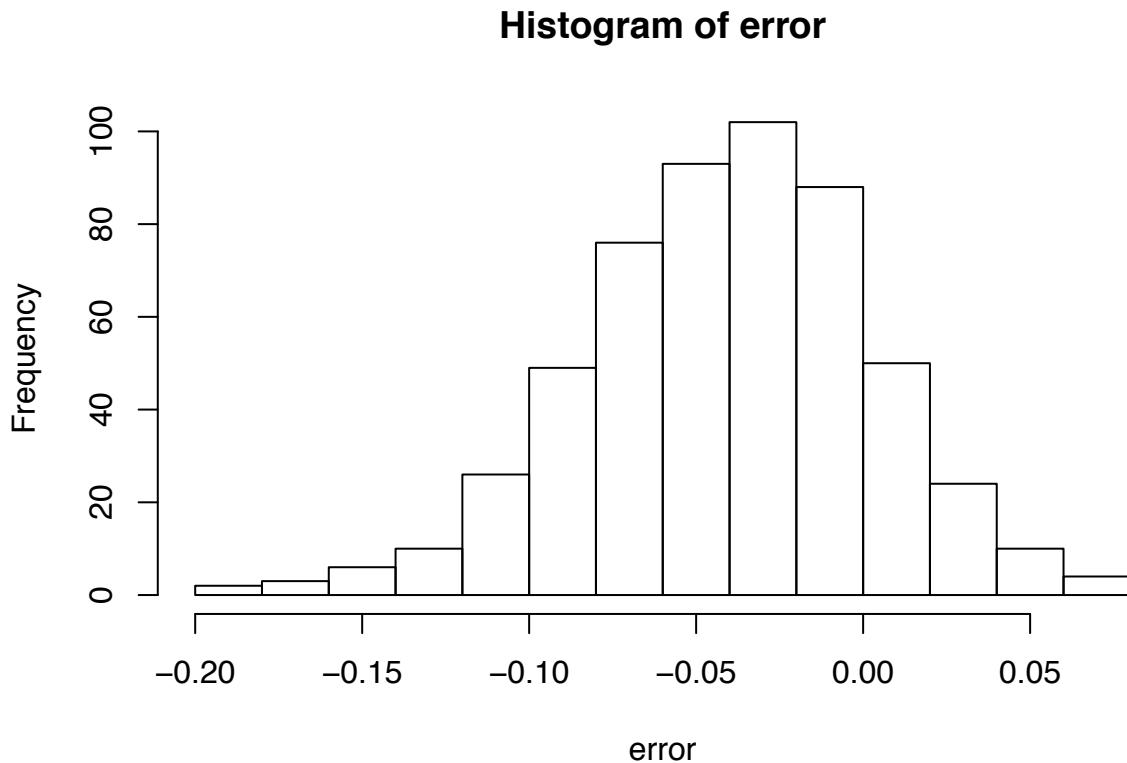
**Assessment 7:** <http://goo.gl/forms/ZAbd1crfK3> Use data exploration to check if the model assumptions make sense. Note that we have the benefit of hindsight. The election night result was  $\theta = 52.9 - 45.7 = 7.2$ .

- 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:

```
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  $\theta$  six months before the election might be different.

**Assessment 8:** <http://goo.gl/forms/GVpfBv7V5> 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  $\theta$  if we let it change with time?

The SD is at least 5%

**Assessment 9:** <http://goo.gl/forms/9q5SSXk6iM> 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
## #> ... ...

```

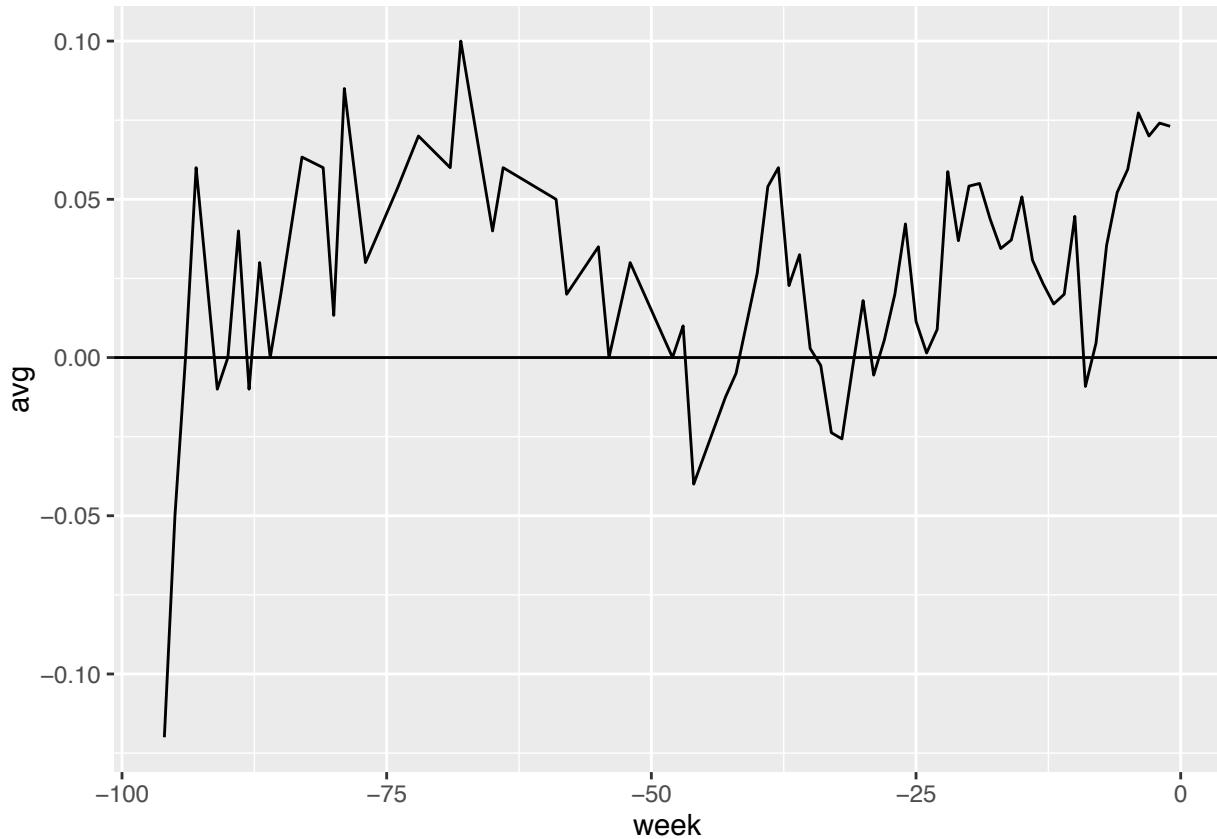
We can make a quick plot

```

library(ggplot2)

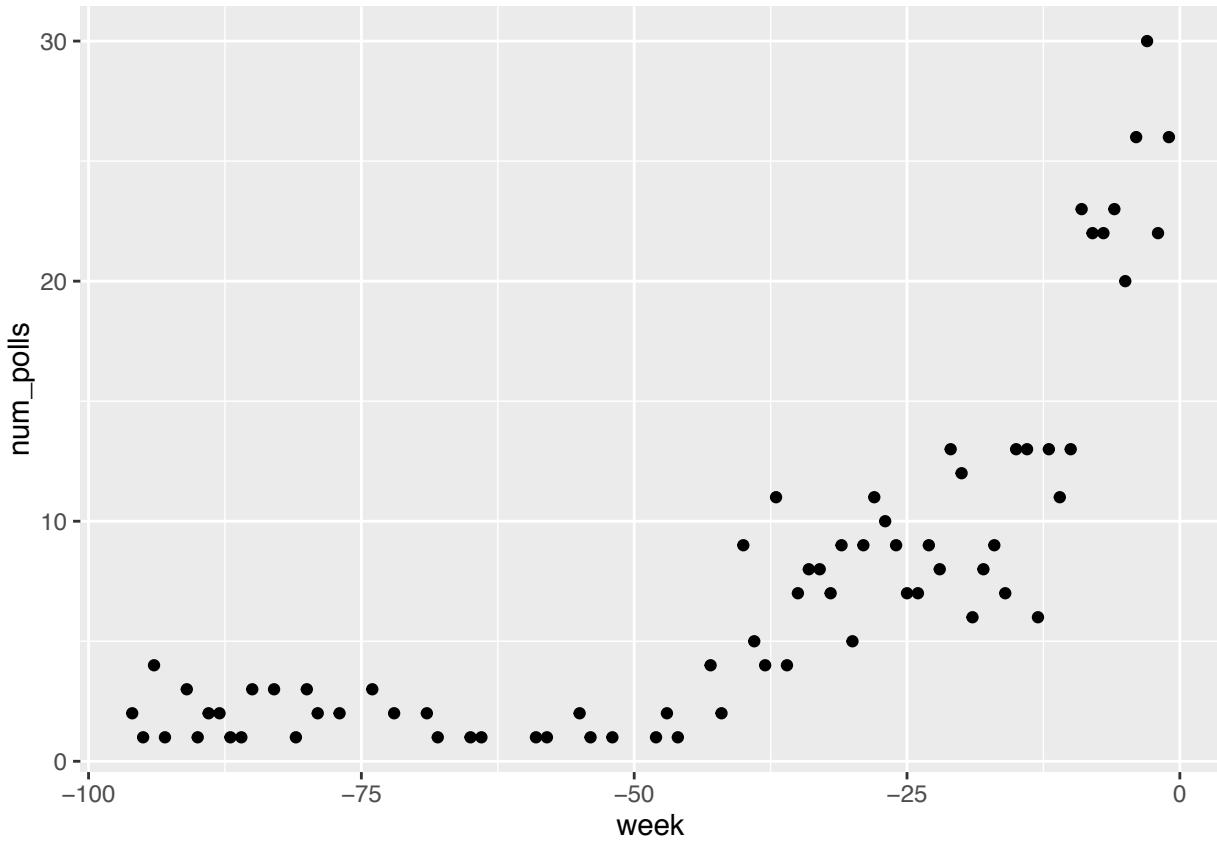
group_by(tab, week) %>% summarize(avg=mean(diff)) %>%
  ggplot(aes(week, avg)) + geom_line() +
  geom_hline(aes(yintercept=0))

```



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) %>% ungroup
```

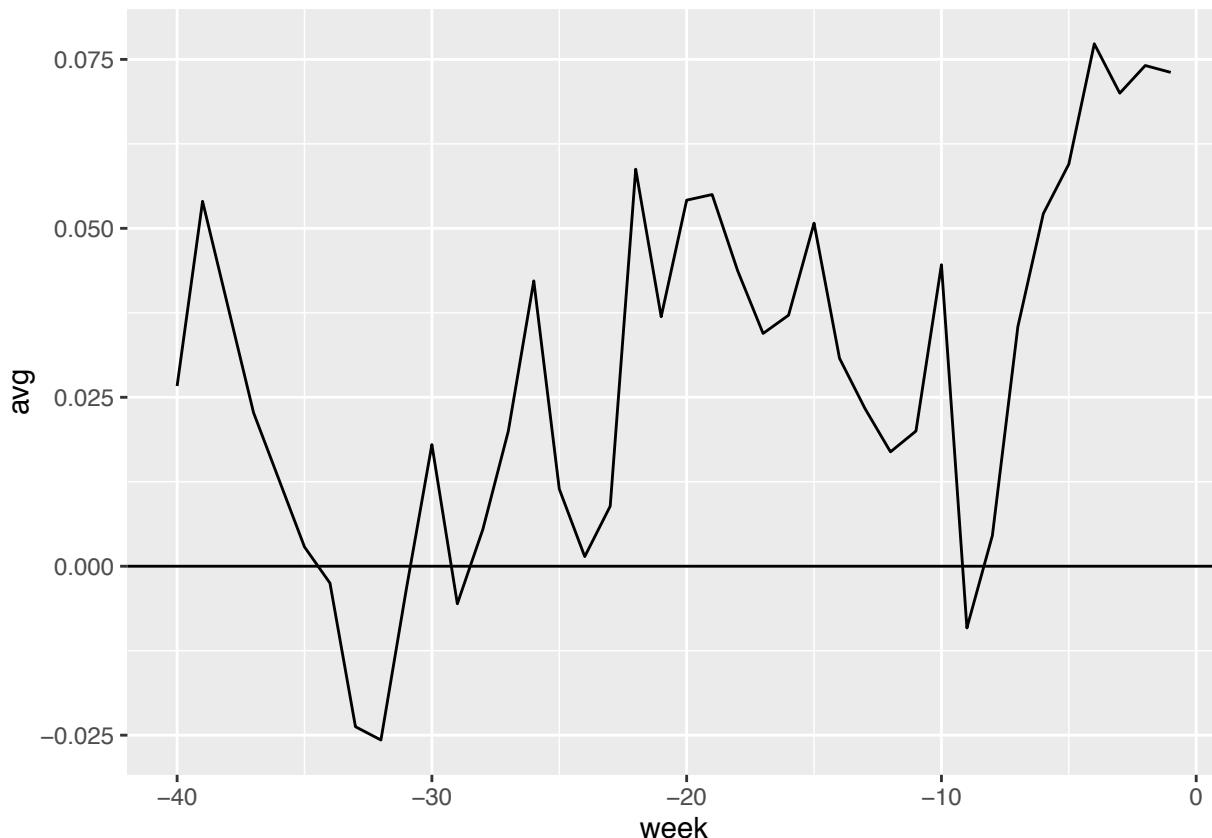
```
## Source: local data frame [543 x 3]
## 
##   week           Pollster num_polls
##   (dbl)          (chr)      (int)
## 1 -1            Marist College    26
## 2 -1            GWU (Lake/Tarrance) 26
## 3 -1 DailyKos.com (D)/Research 2000 26
## 4 -1            IBD/TIPP        26
## 5 -1            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) %>% filter(num_polls >= 5)
## [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))
```

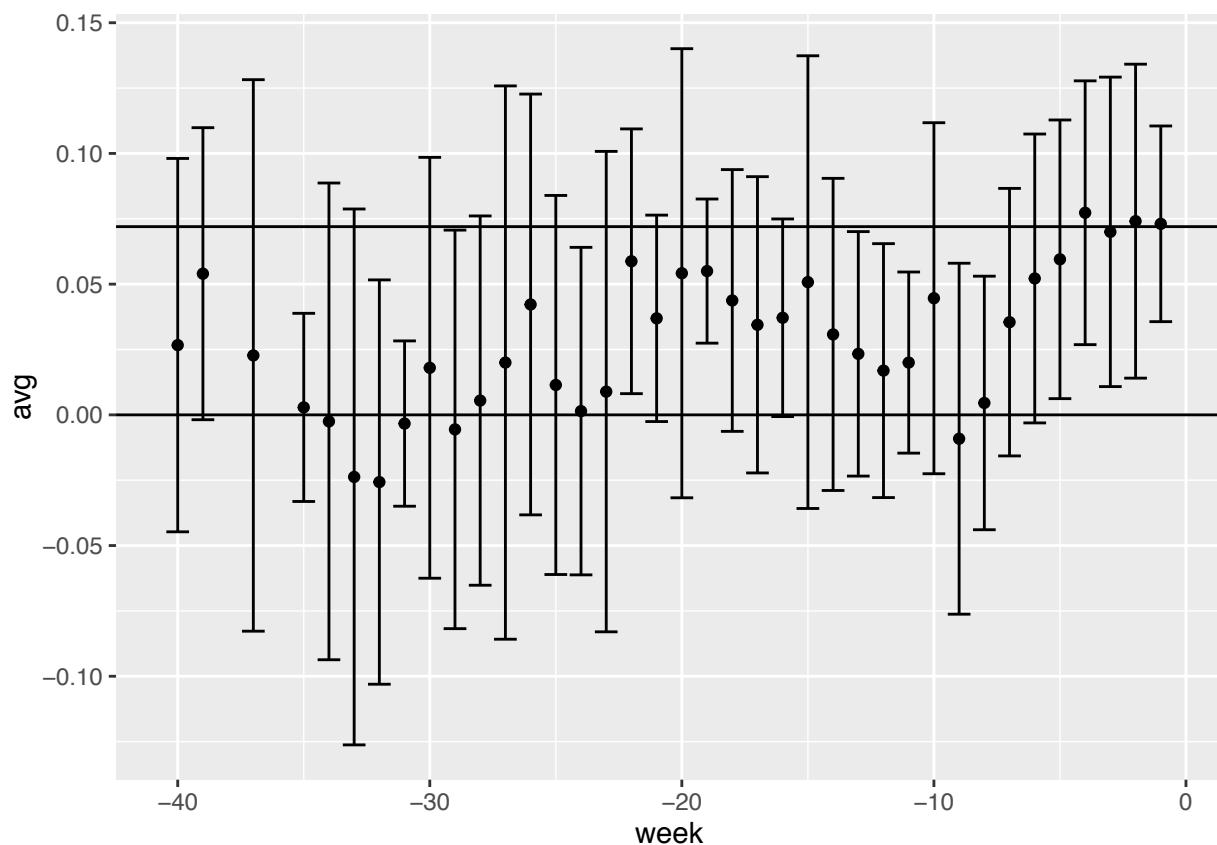


Assessment 10: <http://goo.gl/forms/ntaC4r5H7m>

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))
```

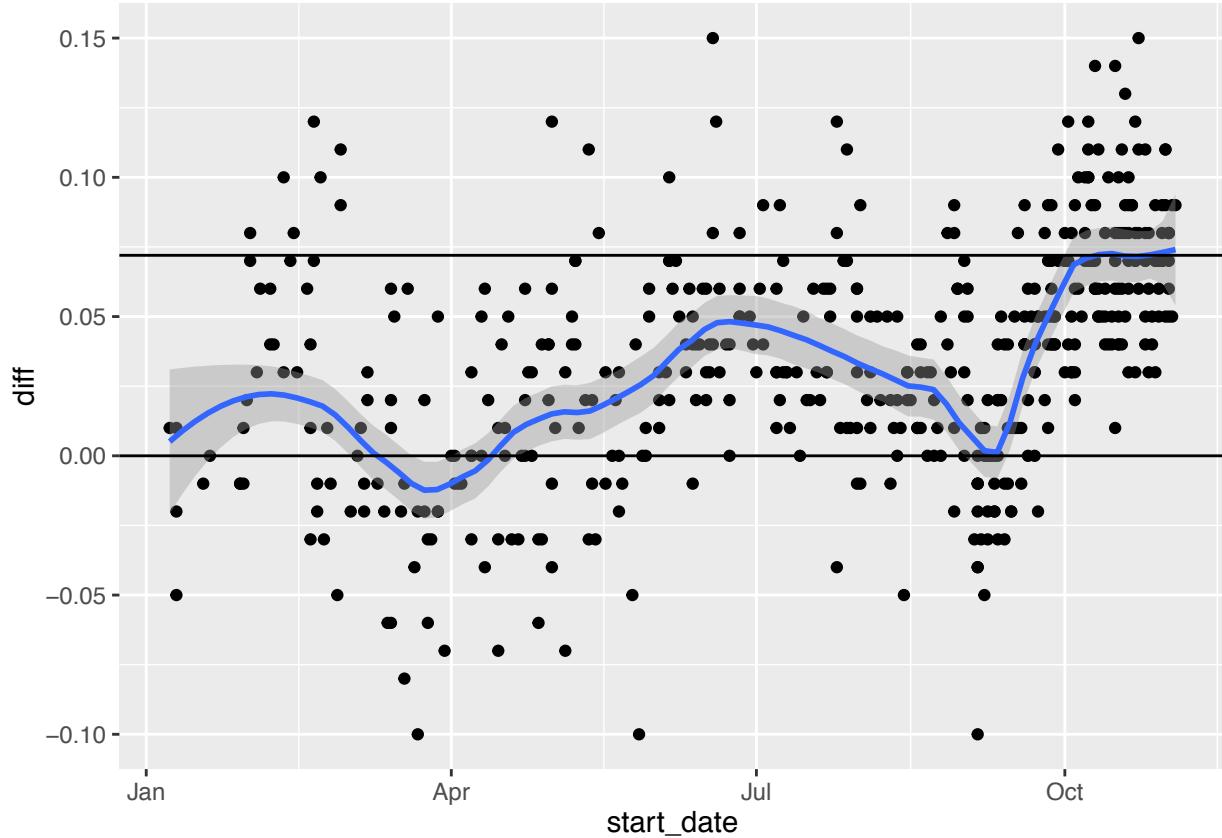


We saw earlier that as the elections got closer many more plots are conducted. But the plot above does not show the standard deviations decreasing. At least not at the rate of  $1/\sqrt{\text{number of polls}}$ . Why is this?

- 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 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  $\theta$ . 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  $\varepsilon$  should be about  $2\sqrt{p(1-p)} / \sqrt{2000}$ :

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

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

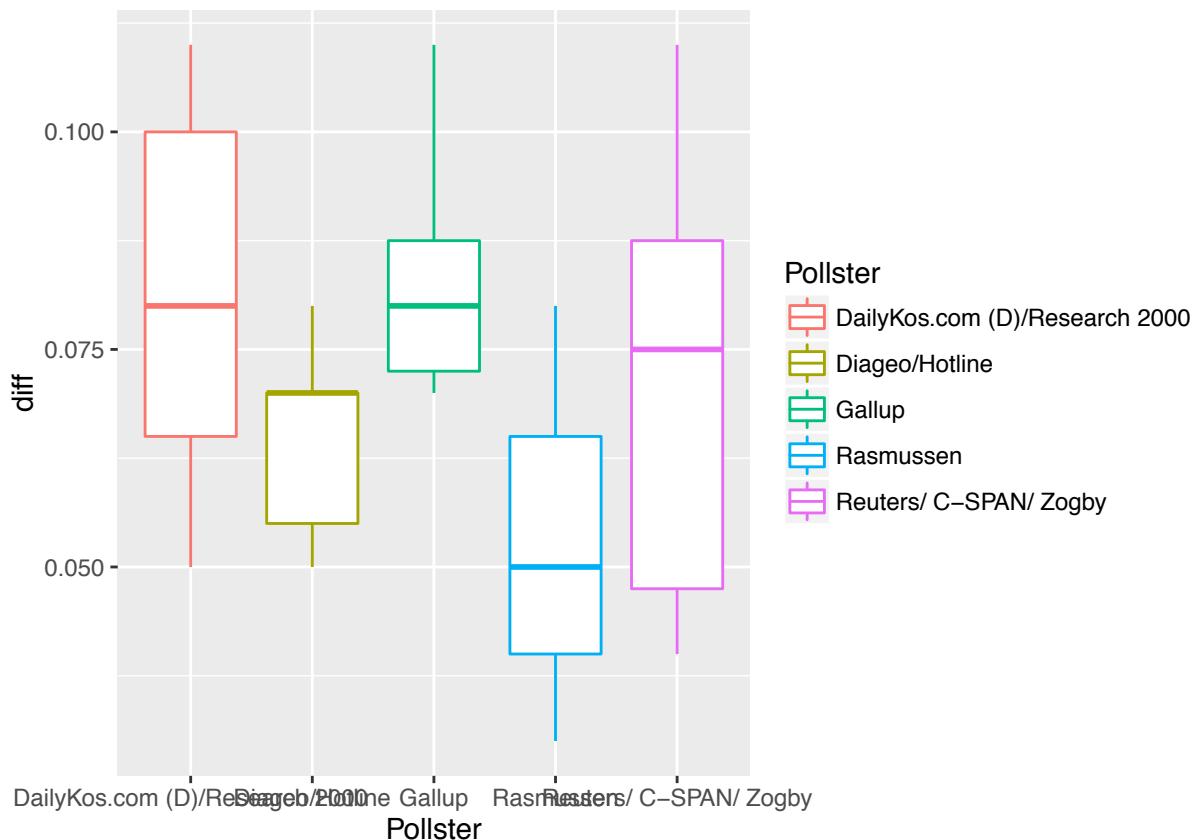
But the observed variance is slightly larger:

```
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
```

# Inference

## Inference

We are going to have a competition. There is a prize.

You have to guess the proportion of blue beads in the jar. You can take a sample (with replacement) from the jar here:

[https://dgrtwo.shinyapps.io/urn\\_drawing/](https://dgrtwo.shinyapps.io/urn_drawing/)

Each sample will cost you 10 cents (\$0.10). So if you take a sample size of 100 you will have to pay me \$10 to collect your prize.

Once you take your sample, please enter your guess, sample size, and an interval size. We will create an interval of this size with your guess in the middle. If the true percentage is not in your interval you are eliminated. The size of the interval will serve as a tie breaker (smaller wins).

<http://goo.gl/forms/5bv4cRQWKA>

The deadline to enter the competition is Monday February 22, 2016 at 9:00 AM.

## Introduction

This chapter introduces the statistical concepts necessary to understand margins of errors, p-values and confidence intervals. These terms are ubiquitous in data science. Let's use polls as an example:

```
##          pollster diff Obama Romney margin_of_error   n
## 1            Gallup -1    48     49      2.00 2551
## 2        Rasmussen -1    48     49      3.00 1500
## 3       Monmouth    0    48     48      2.60 1417
## 4             ARG    0    49     49      3.00 1200
## 5 Politico/GWU/Battleground    0    47     47      3.10 1000
## 6      Gravis Marketing    0    48     48      3.30  872
## 7            CNN    0    49     49      3.50  693
## 8           NBC/WSJ    1    48     47      2.55 1475
## 9           IBD/TIPP    1    50     49      3.70  712
## 10 DailyKos/SEIU/PPP (D)    2    50     48      2.70 1300
## 11                  PPP    2    50     48      2.80 1200
## 12                 Pew    3    48     45      2.20 2709
## 13            ABC/Post    3    50     47      2.50 2345
```

What does *margin of error* mean? The first step is define and understand random variables. To do this, we will generate our own imaginary election with 1 million voters of which proportion  $p$  are democrats and  $1 - p$  are republicans. To keep it interesting we will keep generate  $p$  at random and not peek.

```
n <- 10^6 ##number of voters
set.seed(1) ##so we all get the same answer
##pick a number between 0.45 and 0.55 (don't peek!):
p <- sample( seq(0.45,0.55,0.001),1)
x <- rep( c("D","R"), n*c( p, 1-p))
x <- sample(x) ##not necessary, but shuffle them
```

The population is now fixed. There is a true proportion  $p$  but we don't know it.

One election day we will do the following to decide who wins (don't ruin the fun by doing it now!):

```
prop.table(table(x))
```

Pollsters try to *estimate*  $p$  but asking 1 million people and actually unnecessary so instead. They take a poll. To do this they take a random sample. They pick  $N$  random voter phone numbers, call, and ask. Assuming everybody answers and every voter has a phone, we can mimic a random sample of 100 people with:

```
poll <- sample(x, 25, replace = TRUE)
```

The pollster then looks at the proportion of democrats in the sample and uses this information to predict  $p$ . In Statistics we say try to *estimate*  $p$ .

```
table(poll)
```

```
## poll
## D R
## 13 12
```

So our poll predicts a democrat win! Is this a good *estiamte*? We will see how powerful mathematical statistics is at informing us about exactly how good it is.

Notation: we use lower case  $x$  for the population of all voters and capital letters  $X$  for the random sample.

## Random Variables

Random variable are outcomes of random process. The number of democrats out of the 25 we picked at random is an example. Let's repeat the exercise above a few times. Let's run a few other polls and see what happens:

```
X <- sample(x, 25, replace = TRUE)
sum( X=="D")
```

```
## [1] 14
```

```
X <- sample(x, 25, replace = TRUE)
sum( X=="D")
```

```
## [1] 15
```

```
X <- sample(x, 25, replace = TRUE)
sum( X=="D")
```

```
## [1] 11
```

Note how the observed number varies. We refer to this as *random* or *chance* variation. How does this random variable relate to our quantity of interest  $p$ ? Statistical theory has a lot to teach about this and is the main tool used by pollsters and poll aggregators.

## Sampling Models

Many procedures studied and used by data scientist can be modeled quite well as a sum of draws from a jar. For example, we modeled the process of polling likely voters as drawing 0s (republicans) and 1s (democrats) from a jar. To be more specific we can turn the `x` into 0s and 1s:

```
x <- as.numeric(x=="D")
sample(x, 25, replace = TRUE)

## [1] 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1 0 1 0 0 1
```

Many other examples can be constructed. For example to model our winning after betting \$1 on red (on a roulette wheel) 10 times we can use:

```
color <- rep(c("Black", "Red", "Green"), c(18, 18, 2))
X <- sample(ifelse(color=="Red", 1, -1), 10, replace = TRUE)
sum(X)

## [1] 2
```

Because we know the proportions we can do this a bit quicker by simply using:

```
X <- sample(c(-1, 1), 10, replace = TRUE, prob=c(10/19, 9/19))

## [1] -2
```

## The Expected Value and Standard Error

Now that we have described sampling models, now we can describe properties of the sum of draws. The first important concept to learn is the *expected value*. The random variable will vary around this expected value. The *standard error* gives us an idea of how the size of the variation around the expected value.

The expected value of the sum of draws is the

$$\text{number of draws} \times \text{average of the numbers in the jar}$$

Using the definition of average:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

We can see that the average of values inside a jar with 0s and 1s is the number of 1s divided by  $n$ . This is the proportion of 1s. In general if a jar has two values  $a$  and  $b$  with proportions  $p$  and  $(1-p)$  respectively, the average is  $ap + b(1-p)$ .

**Assessment:** What are the averages for the roulette examples? - Betting on black - Betting on 7

In our example with the election we know that the average of the box is  $p$ , we just don't know what  $p$  is just yet.

If our draws are independent, then the standard error of our sum is

$$\sqrt{\text{number of draws}} \times \text{standard deviation of the numbers in the jar}$$

Using the definition of standard deviation

$$\sigma = \sqrt{\frac{1}{n} \sum_i^n (x_i - \mu)^2}$$

we can derive, with a bit of math, that if a jar has two values  $a$  and  $b$  with proportions  $p$  and  $(1-p)$  respectively, the standard deviation is  $\sqrt{p(1-p)}$ .

**Assessment** What are the expected value and standard errors of the number of democrats. Hint: The answers are functions of  $p$ , the proportion of democrats.

Let  $S$  be the number of democrats in our sample, or the sum of the 25 draws. Our expected value is

$$E(S) = 25p$$

and standard error is

$$SE(S) = \sqrt{25p(1-p)}$$

Two more intuitive results. If we divide  $S$  by constant, the expected value and standard error are also divided by this constant. This implies that

Now we know that the expected value of the proportion in our sample  $S/25$ , has expected value  $p$ .

$$E(S/25) = p$$

which implies that  $S/25$  will be  $p$  plus some chance error.

The SE,

$$SE(S/25) = \sqrt{p(1-p)}/\sqrt{N}$$

gives us an idea of how large the error is.

We do notice that by making  $N$  larger we can make our standard error smaller. The *Law of Large Numbers* tells us that the bigger  $N$  the closer the sample average gets to the average of the jar which is the quantity we want to estimate.

## Estimates

Because  $S/25$  is  $p$  plus or minus some chance error we use it as our *estimate* of  $p$ . We usually put hats on top of our estimates:  $\hat{p}$ . We now know that when  $N$  is large

$$\hat{p} \approx p$$

Suppose that after we take our poll we want to give a best guess for  $p$ . One way we could do this is to report the observed value and its standard error. Unfortunately, we need to know  $p$  to report an expected value. It turns out that plugging in the observed proportion, works out pretty well. So we can report:

```
p_hat <- mean(poll=="D")
cat("Our estimate of the percent of decomrads is",p_hat,
    "plus or minus", round( sqrt(p_hat*(1-p_hat)/25), 2))

## Our estimate of the percent of decomrads is 0.52 plus or minus 0.1
```

Which is not terribly useful. However, now we know that we can be more precise by taking a larger poll.

**Assessment** If  $p = 0.5$ , how large should the poll be to have a standard error of 2% or less ? Then take a poll and report expected value and standard error

$$\sqrt{0.25/N} = 0.02$$

implies

$$N = 0.25/0.02^2 = 625$$

```
N <- 625
X <- sample(x, N, replace = TRUE)
p_hat <- mean(X)
cat("Our estimate of the percent of decomrads is",p_hat,
    "plus or minus", round( sqrt(p_hat*(1-p_hat)/N), 2))
```

```
## Our estimate of the percent of decomrads is 0.4976 plus or minus 0.02
```

## Probability Distribution for Random Variables

The plus or minus statements above are not very precise. Is it possible to say more? Can we, for example, compute the probability that  $\hat{p}$  is within 1% of the actual  $p$  ? It turns out that we can. First let's start with a Monte Carlo simulation.

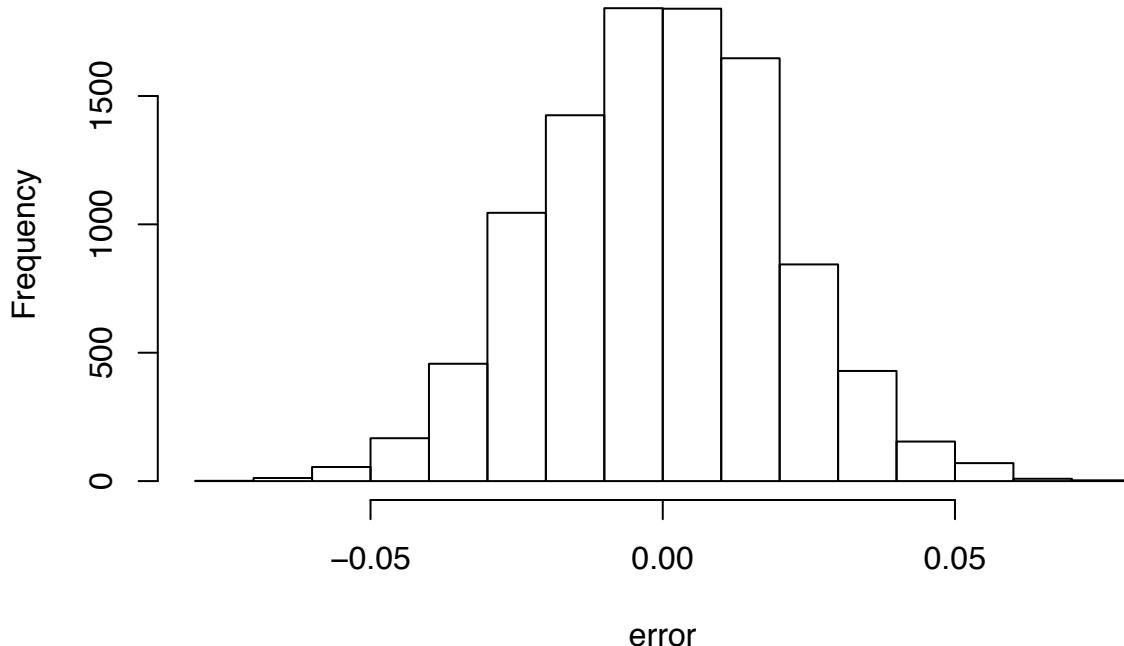
**Assessment:** Repeat the exercise of taking a poll of 625 likely voters, 10,000 times. Without peaking at  $p$  study the distribution of  $\hat{p} - p$ . Have we seen this distribution before? How often was our error smaller/larger than 0.02?

```

N <- 625
B <- 10^4
error <- replicate(B,{
  X <- sample(x, N, replace = TRUE)
  mean(X)-p
})
hist(error)

```

## Histogram of error



```
mean(abs(error) > 0.02)
```

```
## [1] 0.3246
```

The distribution we see here is the *probability distribution* of our random variable. Knowing this distribution will be extremely helpful because we can, for example, report the probability that our error is smaller than a particular quantity. In the next section we describe a powerful mathematical result that helps simplify this.

## Central Limit Theorem

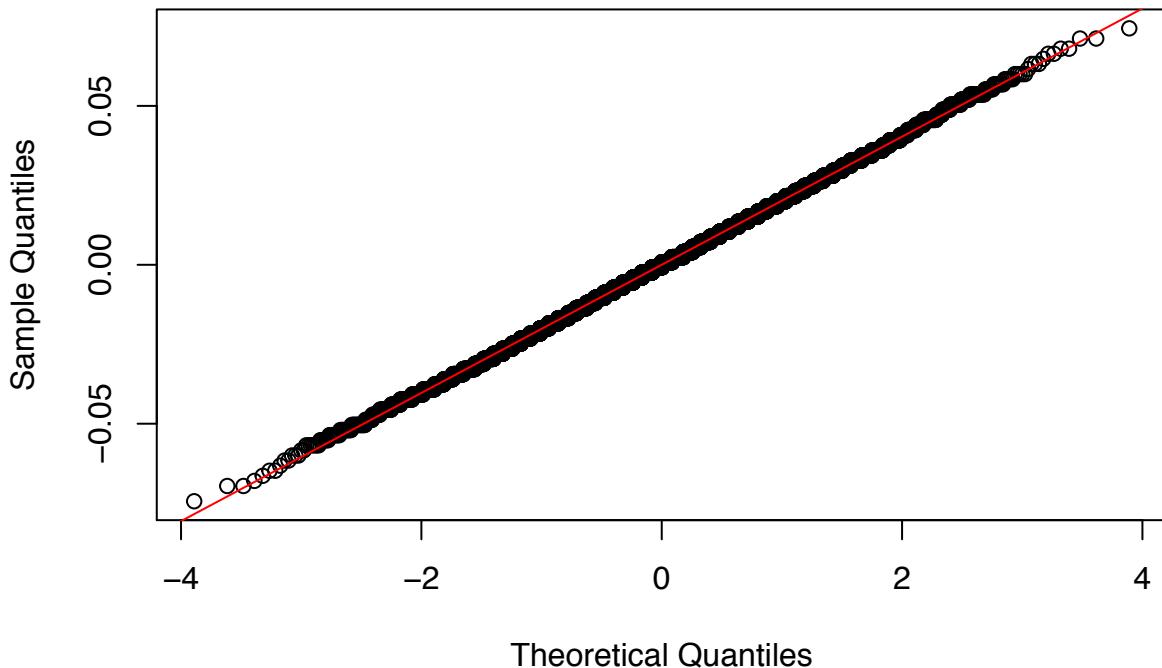
We can see that the distribution of  $\hat{p}$  is very well approximated with the normal distribution.

```

qqnorm(error)
qqline(error,col=2)

```

## Normal Q–Q Plot



### Theoretical Quantiles

The fact that the distribution is centered at 0 confirms that the expected value of  $\hat{p}$  is  $p$ . The fact that the standard deviation of the errors is

```
sqrt(mean( (error)^2))
```

```
## [1] 0.02024796
```

confirms that the standard error is in fact 0.02. Furthermore, we could have predicted that 32% of errors were larger than 0.02 using the normal approximation. What proportion of the normal curve is more than 1 SD away from average?

```
##compare  
mean(abs(error) > 0.02)
```

```
## [1] 0.3246
```

```
##to  
pnorm(-1) + (1-pnorm(1))
```

```
## [1] 0.3173105
```

Notice that the Monte Carlo simulation is like conducting 10,000 polls. This of course is practically impossible. But the mathematical theory saves us from having to do this! Without just one poll we are able to say that our estimate is  $\hat{p}$  and there is 32% chance that our error is larger than 2%.

### Assessment:

Suppose you are consulting for a casino. They want to know how many \$1 bets a person needs to make so that the probability of negative earnings is less than 1 in 1,000.

1. We know that the average of the sampling model is  $-1/19$  and the standard deviation is  $\sigma = 2\sqrt{10/19 \times 9/19}$ . Run a Monte Carlo simulation with  $N = 400$  to learn about the distribution of the errors. Remember the error is the average winning minus  $\mu = -1/19$ . Confirm that the expected value of the error is 0 and that the standard error is  $\sigma/\sqrt{N}$ .

```
N <- 400
B <- 10^5
mu <- -1/19
sigma <- 2*sqrt(9/19*10/19)
error <- replicate(B,{
  X <- sample( c(-1,1), N, replace=TRUE, prob=c(10/19,9/19) )
  mean(X) - mu
})
mean(error)

## [1] -4.132105e-05

sqrt( mean( error^2))

## [1] 0.05006588

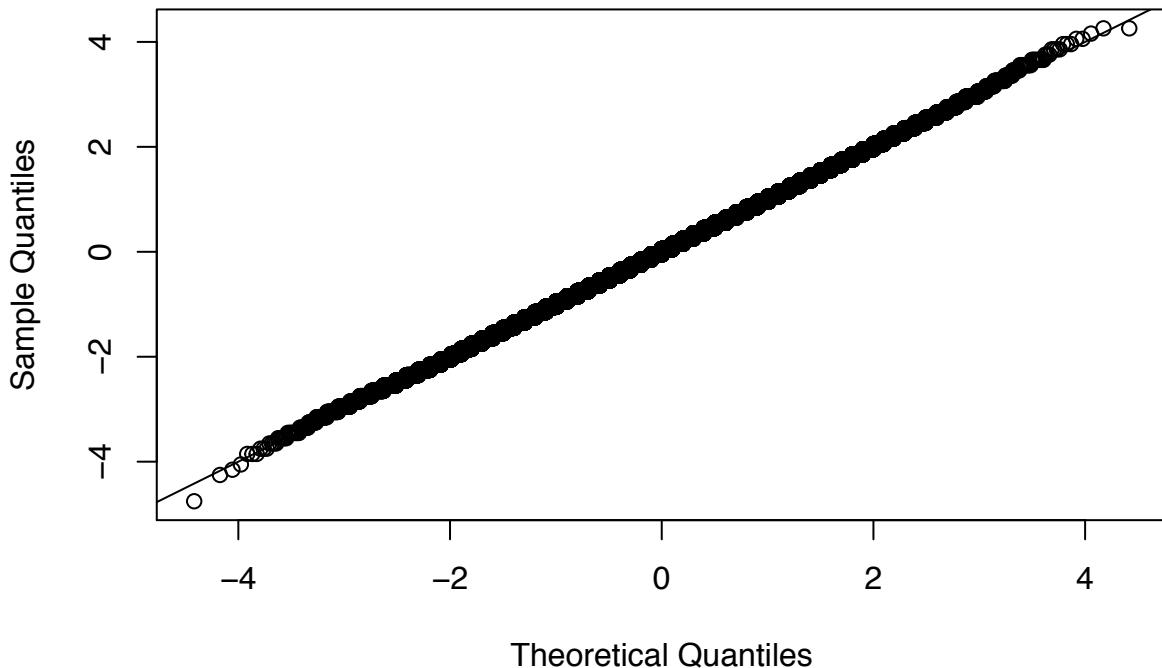
sigma/sqrt(N)

## [1] 0.0499307
```

2. Confirm that the distribution is approximately normal with mean 0 and standard deviation  $\sigma/N$

```
z <- error/ (sigma / sqrt(N) )
qqnorm(z)
abline(0,1)
```

## Normal Q–Q Plot



3. How large does the error have to be for the casino to lose money?

```
## if x is 0, the the error is:
0 - mu
```

```
## [1] 0.05263158
```

4. Use the CLT to compute the probability of a positive earnings. Confirm with the simulation results.

Let  $S$  be the sum of the draws.

$$Z = \sqrt{N} \frac{S/N - \mu}{\sigma}$$

is standard normal. We want to know

$$\Pr(S/N > 0)$$

We rewrite both sides so that  $Z$  is on the left:

$$\Pr(\sqrt{N} \frac{S/N - \mu}{\sigma} > -\sqrt{N} \frac{\mu}{\sigma})$$

which is

$$\Pr(Z > -\sqrt{N} \frac{\mu}{\sigma})$$

and we can compute

```
1 - pnorm( -sqrt(N)*mu/sigma )
```

```
## [1] 0.1459203
```

```
##and compare to  
mean(error > -mu)
```

```
## [1] 0.13458
```

5. With  $N = 400$  the probability of negative earnings is too high. What does  $N$  need to be for the probability of positive earnings is 0? Confirm with a simulation.

We know that if  $z = \Phi^{-1}(1 - 10^{-3})$  then

$$\Pr(Z > z) = 10^{-3}$$

So we need

$$-\sqrt{N} \frac{\mu}{\sigma} = \Phi^{-1}(1 - 10^{-3})$$

or

$$N = (-\Phi^{-1}(1 - 10^{-3})\sigma/\mu)^2$$

```
N <- ceiling( -qnorm(1-10^-3)*sigma/mu )^2
```

Lets' confirm:

```
N <- ceiling( -qnorm(1-10^-3)*sigma/mu )^2  
B <- 10^5  
mu <- -1/19  
sigma <- 2*sqrt(9/19*10/19)  
profit <- replicate(B,{  
  X <- sample( c(-1,1), N, replace=TRUE, prob=c(10/19,9/19)) )  
  mean(X)  
})  
mean(profit>0)
```

```
## [1] 0.00104
```

## Assessment

In R we have access to a large population of male heights. We can access them like this

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

Suppose you are demographer and want to get an estimate of the average height of this population. We can use sampling to do this as well. We will sample 25 men and measure them.

- What is the average and standard deviation of our sampling model? What is the expected value and standard deviation.

```
mu <- mean(y)
sigma <- sqrt(mean((y-mu)^2))
```

- Consider the random variable defined by taking a sample of size  $N$  and then computing the average. What are the expected value and standard error of this random variable:

```
N <- 25
mu

## [1] 68.68407

sigma/sqrt(N)

## [1] 0.5626792
```

- What is the probability that our estimate is within one inch of the population average?

Let's call the sample average

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

Then we have

$$\Pr(|\bar{Y} - \mu| \leq 1) = \Pr(\sqrt{N} |\bar{Y} - \mu| / \sigma \leq \sqrt{N} 1/\sigma)$$

```
(pnorm(sqrt(N)/sigma) - pnorm(-sqrt(N)/sigma))
```

```
## [1] 0.9244666
```

Note that if we wanted to compute this last quantity in practice we need to know  $\sigma$ . But we don't.

A common approach is to use the sample standard deviation as a stand-in. We define the sample standard deviation as

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

The function `sd` computes this quantity

```
Y <- sample(y, N, replace=TRUE)
sd(X)
```

```
## [1] 0.5003947
```

```

1 - pnorm(sqrt(N)/sd(Y))

## [1] 0.009681883

#is an approximation of
1 - pnorm(sqrt(N)/sigma)

## [1] 0.0377667

##that is completely derived from data

```

### The t-distribution (optional)

If you have heard of the t-distribution, you should know that here is where we would use it. If the original data is normal, as height data is, then there is a better approximation for

$$\sqrt{N} \frac{\bar{X} - \mu}{s}$$

than the normal distribution and it is the t-distribution. Note these are closer

```

1 - pnorm(sqrt(N)/sigma)

## [1] 0.0377667

#is an approximation of
1 - pt(sqrt(N)/sd(X), N-1)

## [1] 2.49593e-10

1 - pnorm(sqrt(N)/sd(X))

## [1] 0

```

### Confidence Intervals

If we create intervals that miss the target 32% of the time, we will not be considered good forecasters. We can always make less bold predictions, such as “the percent of democrats will be between 0% and 100%” but that will not help our reputation either. Can we find a better balance?

Because there is chance variation, it is impossible to hit the target 100% of time unless we state the obvious “between 0% and 100%”. So let’s compromise a bit. If we hit the target 95% of the time, chances are we will be considered good forecasters. Can we use the theory above to construct such interval?

What we want to do is use the result of our poll to construct an interval, say,  $[A, B]$  such that

$$\Pr(A \leq p \text{ and } B \geq p) \geq 0.95$$

we want to make this interval as small as possible. How do we choose  $A$  and  $B$ ?

We know  $\hat{p}$  follows a normal distribution with expected value  $p$  and standard error approximately equal to  $\sqrt{\hat{p}(1 - \hat{p})}/\sqrt{N}$ . This implies that the following random variable

$$Z = \sqrt{N} \frac{\hat{p} - p}{\sqrt{\hat{p}(1 - \hat{p})}}$$

is approximately normal with expected 0 and standard deviation 1.

**Assessment** What value can we make  $z$  so that  $\Pr(|Z| \leq z) = 0.95$  ?

Answer:

$$z = \Phi^{-1}(0.975)$$

Because:

```
z = qnorm(0.975)
pnorm(z) - pnorm(-z)
```

```
## [1] 0.95
```

So we have our interval. We need to actually define it in terms of  $\hat{p}$ ,  $N$ , and  $z$  because these are the only quantity we can actually construct.

**Assessment: Practice the math** How do we rewrite the event

$$|Z| \leq z$$

so that it can be written as an interval of quantities we know.

$$|Z| = \sqrt{N} \frac{|\hat{p} - p|}{\sqrt{\hat{p}(1 - \hat{p})}} \leq z$$

$$|\hat{p} - p| \leq z \sqrt{\hat{p}(1 - \hat{p})} \sqrt{N}$$

Assuming  $z$  is positive,

$$\hat{p} - z \sqrt{\hat{p}(1 - \hat{p})} / \sqrt{N} \leq p \leq \hat{p} + z \sqrt{\hat{p}(1 - \hat{p})} / \sqrt{N}$$

```
p_hat + c(-1,1)*qnorm(0.975)*sqrt(p_hat*(1-p_hat))/sqrt(N)
```

```
## [1] 0.3016059 0.6935941
```

This interval has a 95% chance of *covering* the true  $p$

Did we actually make a correct prediction? It's election night and we will find out

```
ci <- p_hat + c(-1,1)*qnorm(0.975)*sqrt(p_hat*(1-p_hat))/sqrt(N)
p
```

```
## [1] 0.476
```

```
ci[1] <= p & ci[2] >= p
```

```
## [1] TRUE
```

**Assessment** Construct a 99% confidence interval. Is it more or less specific than the 95%?

```
p_hat + c(-1,1)*qnorm(0.995)*sqrt(p_hat*(1-p_hat))/sqrt(N)
```

```
## [1] 0.24002 0.75518
```

**Assessment** You have a coin. You want to figure out if it's fair or not. To create your coin run the following code:

```
set.seed(2016)
prob_heads <- sample(c(0.5, 0.6, 0.4), 1)
```

1. Now the probability is set. The coin is either fair or not, but we don't know which. Use a Monte Carlo simulation to toss your coin 100 times. Report the proportion of heads

```
N <- 100
X <- sample(c(0,1), N, replace=TRUE,
            prob=c(1-prob_heads,prob_heads))
ph_hat <- mean(X)
ph_hat
```

```
## [1] 0.59
```

2. Is it fair? Use CLT to construct a 95% confidence based on this estimate.

```
conf_int <- ph_hat +
  c(-1,1)*qnorm(0.975)*sqrt(ph_hat*(1-ph_hat))/sqrt(N)
conf_int
```

```
## [1] 0.4936024 0.6863976
```

3. Does the interval cover the true value of `prob_head`?

```
prob_heads>=ci[1] & prob_heads<=ci[2]
```

```
## [1] TRUE
```

4. No use a Monte Carlo simulation to re run this experiment (keeping `prob_head` fixed) 10000 times. Do we confirm that we cover 95% of the time?

```

B <- 10000
cover <- replicate(B,{
  X <- sample(c(0,1), N, replace=TRUE,
              prob=c(prob_heads,1-prob_heads))
  ph_hat <- mean(X)
  ci <- ph_hat +
    c(-1,1)*qnorm(0.975)*sqrt(ph_hat*(1-ph_hat))/sqrt(N)
  prob_heads>=ci[1] & prob_heads<=ci[2]
})
mean(cover)

```

```
## [1] 0.9463
```

### Confidence Intervals: Monte Carlo Simulation

Note that Nate Silver stated that Obama had a 90% chance of winning. Was this a confidence interval?

The description we have given up to know says nothing about the probability of winnings. In fact, the statement does not even make sense because  $p$  is fixed. It is not random. We should not make probability statements.

Later we will learn about Bayesian statistics were data-based statements such as “Obama has a 90% chance of winning” make sense. Here we clarify how we interpret margins of error and confidence intervals.

The 95% confidence interval

```
p_hat + c(-1,1)*qnorm(0.975)*sqrt(p_hat*(1-p_hat))/sqrt(N)
```

```
## [1] 0.3996029 0.5955971
```

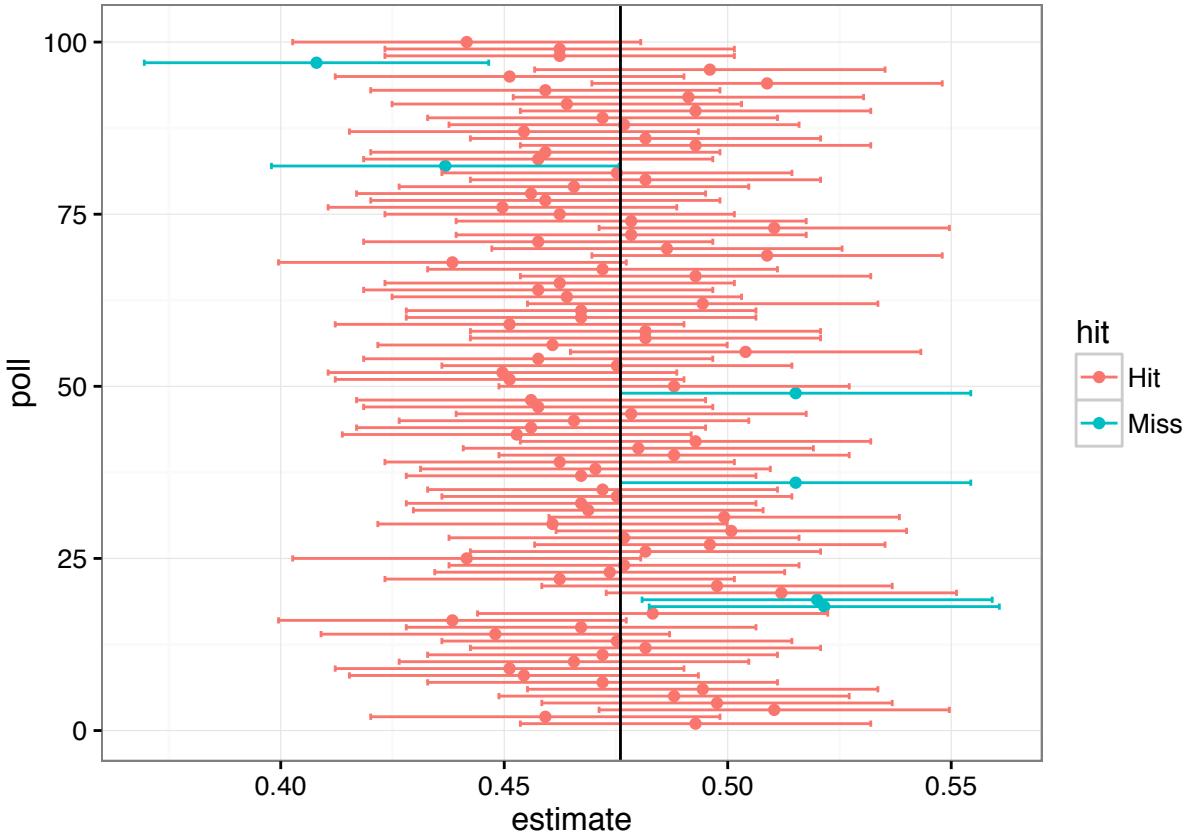
is random because  $p_{hat}$  is random. Let's use a Monte Carlo simulation to see this:

```

library(dplyr)
library(ggplot2)
N <- 625
B <- 100
tab <- replicate(B,{
  X <- sample(x, N, replace = TRUE)
  p_hat <- mean(X)
  ci <- p_hat + c(-1,1)*qnorm(0.975)*sqrt(p_hat*(1-p_hat))/sqrt(N)
  hit <- ci[1] <= p & ci[2] >= p
  c(p_hat,ci,hit)
})

tab <- data.frame(poll=1:ncol(tab), t(tab))
names(tab)<-c("poll","estimate","low","high","hit")
tab <- mutate(tab, hit=ifelse(hit, "Hit","Miss") )
ggplot(tab, aes(poll,estimate, ymin=low,ymax=high,col=hit)) +geom_point(aes())+geom_errorbar() + coord_fi

```



Notice how the interval is different after each poll: it's random. So the 95% relates to the probability that this random interval falls on top of  $p$ , not that  $p < .50$  or whatever other statements related to the probability of winning. In the figure above, that shows the interval for 100 polls, you can see that interval fall on  $p$  about 95% of the time.

### Power

Pollsters are not successful for providing correct confidence intervals but rather for predicting who will win. Unfortunately, our confidence interval included 0.5 so if forced to declare a winner, we would have to say it was a "toss-up". A problem with our poll results is that, given the sample size and the value of  $p$ , we would have to sacrifice on the probability of an incorrect call to create an interval that does not include 50. In this particular case, the reported margin of error would have had to be about 3.5%

### Assessment

What percent of confidence intervals with margins of error of 0.03 would correctly predict the election? Check with a Monte Carlo simulation.

Define

$$\delta = 0.035$$

We want to know

$$\Pr(-\delta < |p - \hat{p}| < \delta)$$

$$\Pr(|Z| < \sqrt{N}\delta/\sqrt{p(1-p)})$$

Because  $Z$  is approximately standard normal this probability is approximately

```
a = sqrt(N)*.035/sqrt(p_hat*(1-p_hat))
pnorm(a)-pnorm(-a)
```

```
## [1] 0.9198852
```

This a .91% confidence interval. Which is not bad, but it's not the 95% we set out for.

```
N <- 625
B <- 10^5
success <- replicate(B,{
  X <- sample(x, N, replace = TRUE)
  p_hat <- mean(X)
  ci <- p_hat + c(-0.035,0.035)
  ci[1] <= p & ci[2] >= p
})
mean(success)
```

```
## [1] 0.92121
```

## Assessment

If we know the percent of democrats can be as close to 50% as, say, 48%, how large does the sample size have to be so that the margin of error is about 2%.

Solution we need: We add  $z_{95}\sqrt{p(1-p)}\sqrt{N}$  and we want this to be 0.02. So

```
(qnorm(0.975)*sqrt(0.48*0.52)/0.02)^2
```

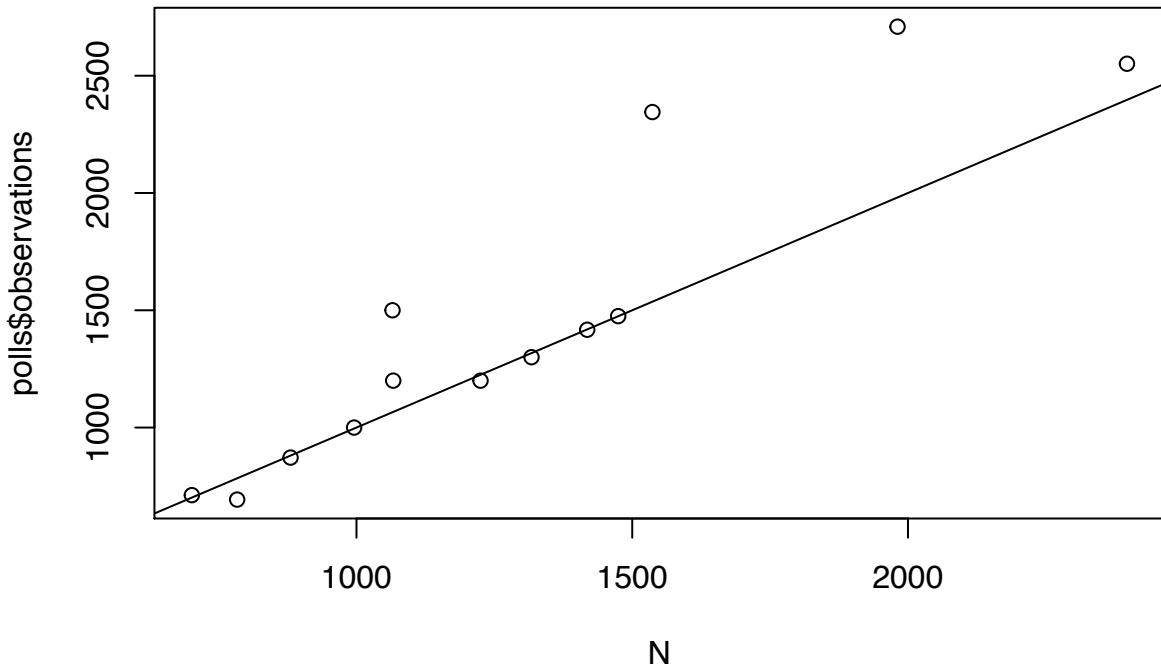
```
## [1] 2397.07
```

If you go back to the table with which we started, you notice that many polls have sample sizes between 500 and 2500.

**Assessment** From the `polls` object determine the sample size from the margin of error. The plot it against the reported sample size.

What do you see? A - A flat line B - Exactly a straight line C - Almost a straight line with some points off D - The function  $f(x) = \sqrt{x}$

```
p_hat <- polls$Obama/100
N <- ( qnorm(0.975) * sqrt(p_hat*(1-p_hat)) /
  (polls$margin_of_error/100))^2
plot(N, polls$observations)
abline(0,1)
```



### 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.

### p-values

p-values are ubiquitous in the scientific literature. They are related to confidence interval so we introduce the concept here.

Let's consider the blue and red beads. Suppose that rather than wanting an estimate of the percent of blue beads I am more interested in the question are there more blue beads or red beads.

Suppose we take a random sample of  $N = 100$  and we observe 53 blue beads. This seems to be pointing to their being more blue than red. However, as data scientists we need to be skeptical. We know there is chance involved in this process and we could get a 53 even when the proportions of red and blue are the same. We call this a *null hypothesis*. The null hypothesis is the skeptics hypothesis: the proportion of blue beads  $p$  is 0.5. We have observed a random variable  $\hat{p} = 0.53$  and the p-value is the answer to the question how likely is it to see a value this large, when the null hypothesis is true. So we write

$$\Pr(|\hat{p} - 0.5| > 0.03)$$

assuming the  $p = 0.5$ . Under the null we know that

$$\sqrt{N} \frac{\hat{p} - 0.5}{\sqrt{0.5(1 - 0.5)}}$$

is standard normal. So we can compute the probability above, which is the p-value.

$$\Pr \left( \sqrt{N} \frac{|\hat{p} - 0.5|}{\sqrt{0.5(1 - 0.5)}} > \sqrt{N} \frac{0.03}{\sqrt{0.5(1 - 0.5)}} \right)$$

```
N=100  
z <- sqrt(N)*0.03/0.5  
1 - (pnorm(0.6) - pnorm(-0.6))
```

```
## [1] 0.5485062
```

So we do in fact have reason to be a skeptics. By constructing a p-value we see that

Assessment: If  $\hat{p} = 51$  and  $N = 10000$ . What is the p-value?

We can show that if a  $x \times 100\%$  confidence interval does not include 0, then the p-value must be smaller than  $1 - x$ . So they provide related information. However, the confidence interval is always more informative as it gives information of the size of the estimate.

Assessment: Suppose you are comparing average test scores from two school districts. You have exams from a random sample of 10000. The p-value for the difference in average scores is 0.01. Should we change the curriculum of the district with highly significant statistical differences?

**Setting the random seed** Before we continue, we briefly explain the following important line of code:

```
set.seed(1)
```

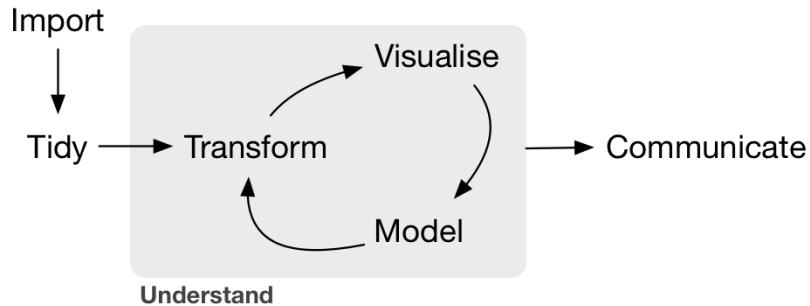
Throughout this book, we use random number generators. This implies that many of the results presented can actually change by chance, including the correct answer to problems. One way to ensure that results do not change is by setting R's random number generation seed. For more on the topic please read the help file:

```
?set.seed
```

# Data Wrangling with dplyr

Stephanie Hicks, Rafael Irizarry

Once data has been transformed into a **tidy** tabular format, an important part of “data wrangling” or “data munging” is data transformation.



We have learned about some functions in the R package **dplyr** that can transform and summarize tabular data with rows and columns. For example, when data is a tabular format, you have seen how we can use **dplyr** to **filter()** rows, **select()** columns and add new columns using **mutate()**. Now we will explore some more advanced **dplyr** functionality.

## Brief recap of dplyr

**dplyr** is a powerful R-package to transform and summarize tabular data with rows and columns.

The package contains a set of functions (or “verbs”) to perform common data manipulation operations such as filtering for rows, selecting specific columns, re-ordering rows, adding new columns and summarizing data. In addition, **dplyr** contains a useful function to perform another common task which is the “split-apply-combine” concept. We will discuss that in a little bit.

## Data

### mammals sleep

The **msleep** (mammals sleep) data set contains the sleep times and weights for a set of mammals and is available in the [data repository on GitHub](#). This data set contains 83 rows and 11 variables.

To load the **msleep** data set

```
library(readr)
library(dplyr)
library(ggplot2)

msleep <- read_csv("https://raw.githubusercontent.com/datasets/labs/master/msleep_ggplot2.csv")
msleep
```

```
## Source: local data frame [83 x 11]
##
##          name      genus   vore      order conservation
## 1     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 2     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 3     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 4     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 5     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 6     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 7     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 8     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 9     Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 10    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 11    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 12    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 13    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 14    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 15    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 16    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 17    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 18    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 19    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 20    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 21    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 22    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 23    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 24    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 25    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 26    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 27    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 28    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 29    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 30    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 31    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 32    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 33    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 34    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 35    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 36    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 37    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 38    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 39    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 40    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 41    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 42    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 43    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 44    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 45    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 46    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 47    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 48    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 49    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 50    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 51    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 52    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 53    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 54    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 55    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 56    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 57    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 58    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 59    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 60    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 61    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 62    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 63    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 64    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 65    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 66    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 67    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 68    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 69    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 70    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 71    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 72    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 73    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 74    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 75    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 76    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 77    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 78    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 79    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 80    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 81    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 82    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
## 83    Acomys cahirinus  Acomyidae Carnivore Rodentia       1
```

```

##                               (chr)      (chr) (chr)      (chr)      (chr)
## 1                  Cheetah    Acinonyx carni   Carnivora    lc
## 2          Owl monkey     Aotus omni    Primates     NA
## 3  Mountain beaver  Aplodontia herbi   Rodentia     nt
## 4 Greater short-tailed shrew   Blarina omni Soricomorpha lc
## 5                           Cow     Bos herbi Artiodactyla domesticated
## 6            Three-toed sloth  Bradypus herbi    Pilosa      NA
## 7       Northern fur seal Callorhinus carni   Carnivora    vu
## 8        Vesper mouse      Calomys    NA   Rodentia      NA
## 9           Dog          Canis carni   Carnivora domesticated
## 10        Roe deer      Capreolus herbi Artiodactyla lc
## ..          ...         ...    ...    ...
## Variables not shown: sleep_total (dbl), sleep_rem (dbl), sleep_cycle
## (dbl), awake (dbl), brainwt (dbl), bodywt (dbl)

```

The columns (in order) correspond to the following:

column name	Description
name	common name
genus	taxonomic rank
vore	carnivore, omnivore or herbivore?
order	taxonomic rank
conservation	the conservation status of the mammal
sleep_total	total amount of sleep, in hours
sleep_rem	rem sleep, in hours
sleep_cycle	length of sleep cycle, in hours
awake	amount of time spent awake, in hours
brainwt	brain weight in kilograms
bodywt	body weight in kilograms

## Important dplyr verbs to remember

dplyr verbs	Description
select()	select columns
mutate()	create new columns
filter()	filter rows
arrange()	re-order or arrange rows
summarise()	summarise values
group_by()	allows for group operations in the “split-apply-combine” concept

## dplyr verbs in action

**select() and mutate() columns; filter() rows**

The two most basic functions are `select()` and `filter()` which selects columns and filters rows, respectively. The function `mutate()` can be used to create new columns. We have already seen examples of all of these in class.

For example, to select a range of columns by name, use the “:” (colon) operator

```
msleep %>% select(name:order)

## Source: local data frame [83 x 4]
##
##          name      genus   vore     order
##          (chr)      (chr) (chr)      (chr)
## 1       Cheetah    Acinonyx carni Carnivora
## 2       Owl monkey     Aotus omni Primates
## 3   Mountain beaver   Aplodontia herbi Rodentia
## 4 Greater short-tailed shrew     Blarina omni Soricomorpha
## 5             Cow        Bos herbi Artiodactyla
## 6 Three-toed sloth     Bradypus herbi Pilosa
## 7 Northern fur seal   Callorhinus carni Carnivora
## 8      Vesper mouse     Calomys NA Rodentia
## 9             Dog        Canis carni Carnivora
## 10            Roe deer    Capreolus herbi Artiodactyla
## ..           ...       ...   ...
```

To select all columns that start with the character string “sl”, use the function `starts_with()`

```
msleep %>% select(starts_with("sl"))
```

```
## Source: local data frame [83 x 3]
##
##    sleep_total sleep_rem sleep_cycle
##    (dbl)      (dbl)      (dbl)
## 1     12.1       NA       NA
## 2     17.0      1.8       NA
## 3     14.4      2.4       NA
## 4     14.9      2.3  0.1333333
## 5      4.0      0.7  0.6666667
## 6     14.4      2.2  0.7666667
## 7      8.7      1.4  0.3833333
## 8      7.0       NA       NA
## 9     10.1      2.9  0.3333333
## 10     3.0       NA       NA
## ..       ...     ...     ...
```

Some additional options to select columns based on a specific criteria include

1. `ends_with()` = Select columns that end with a character string
2. `contains()` = Select columns that contain a character string
3. `matches()` = Select columns that match a regular expression
4. `one_of()` = Select columns names that are from a group of names

**Assessment** Select all columns except those from `genus` to `conservation` and filter the rows for mammals that sleep a total of more than 16 hours **and** have a body weight of greater than 1 kilogram

```

## Provide your code here

msleep %>%
  select(-(genus:conservation)) %>%
  filter(sleep_total >= 16, bodywt >= 1)

## Source: local data frame [3 x 7]
##
##          name sleep_total sleep_rem sleep_cycle awake brainwt
##          (chr)      (dbl)      (dbl)      (dbl)  (dbl)   (dbl)
## 1 Long-nosed armadillo     17.4       3.1    0.3833333   6.6  0.0108
## 2 North American Opossum    18.0       4.9    0.3333333   6.0  0.0063
## 3 Giant armadillo        18.1       6.1        NA    5.9  0.0810
## Variables not shown: bodywt (dbl)

```

**Assessment** Create a new column called `rem_proportion` which is the ratio of rem sleep to total amount of sleep and create boxplots of the `rem_proportion` column split and colored by the `vore` column. Create labels for the x and y axis.

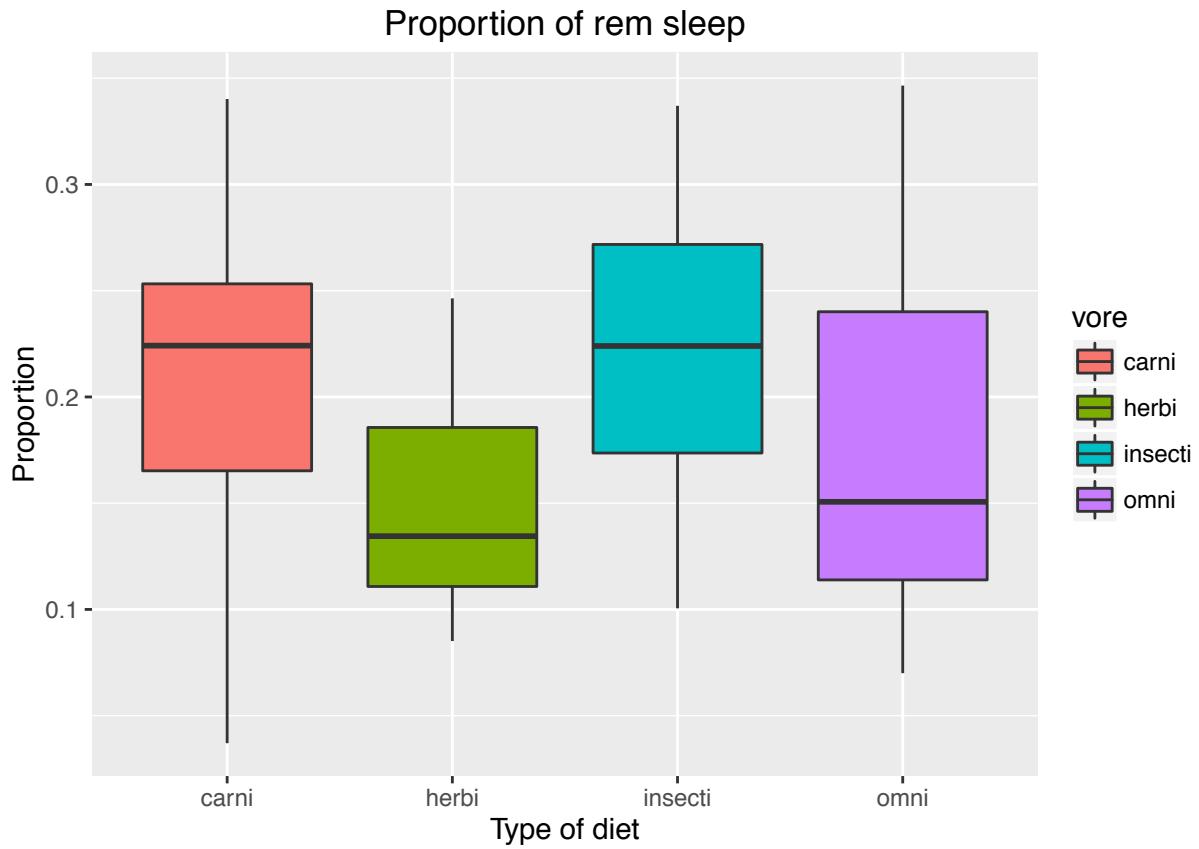
```

## Provide your code here

msleep %>%
  mutate(rem_proportion = sleep_rem / sleep_total) %>%
  ggplot(aes(x = vore, y = rem_proportion, fill = vore)) +
  geom_boxplot() + xlab("Type of diet") +
  ylab("Proportion") + labs(title = "Proportion of rem sleep")

## Warning: Removed 27 rows containing non-finite values (stat_boxplot).

```



**Assessment (optional)** Select all columns that start with the character string “sl” or ends with the character string “wt”, create a new column called `rem_proportion` which is the ratio of rem sleep to total amount of sleep, create a second column `bodywt_grams` which is the `bodywt` column in grams and filter for the rows 20 to 30 in the `msleep` data set by numerical position.

Hint: Look at the `slice()` help file to filter for rows by numerical position.

```
## Provide your code here
```

```
msleep %>%
  select(starts_with("sl"), ends_with("wt")) %>%
  mutate(rem_proportion = sleep_rem / sleep_total,
        bodywt_grams = bodywt * 1000) %>%
  slice(20:30)
```

```
## Source: local data frame [11 x 7]
##
##   sleep_total sleep_rem sleep_cycle brainwt   bodywt rem_proportion
##   (dbl)      (dbl)      (dbl)    (dbl)     (dbl)          (dbl)
## 1     18.0      4.9  0.3333333  0.0063     1.700  0.2722222
## 2      3.9       NA       NA     4.6030  2547.000        NA
## 3     19.7      3.9  0.1166667  0.0003     0.023  0.1979695
## 4      2.9       0.6  1.0000000  0.6550     521.000  0.2068966
## 5      3.1       0.4       NA     0.4190    187.000  0.1290323
## 6     10.1      3.5  0.2833333  0.0035     0.770  0.3465347
## 7     10.9      1.1       NA     0.1150    10.000  0.1009174
```

```

## 8      14.9      NA      NA      NA    0.071      NA
## 9      12.5      3.2  0.4166667  0.0256   3.300  0.2560000
## 10     9.8       1.1  0.5500000  0.0050   0.200  0.1122449
## 11     1.9       0.4      NA      NA  899.995  0.2105263
## Variables not shown: bodywt_grams (dbl)

```

### Arrange or re-order rows using `arrange()`

To arrange (or re-order) rows by a particular column such as the taxonomic order, list the name of the column you want to arrange the rows by

```

msleep %>%
  arrange(order)

```

```

## Source: local data frame [83 x 11]
##
##              name      genus  vore      order conservation
##              (chr)      (chr) (chr)      (chr)      (chr)
## 1      Tenrec      Tenrec omni Afrosoricida      NA
## 2        Cow        Bos herbi Artiodactyla domesticated
## 3  Roe deer  Capreolus herbi Artiodactyla      lc
## 4       Goat        Capri herbi Artiodactyla      lc
## 5     Giraffe      Giraffa herbi Artiodactyla      cd
## 6      Sheep       Ovis herbi Artiodactyla domesticated
## 7       Pig         Sus omni Artiodactyla domesticated
## 8    Cheetah  Acinonyx carni Carnivora      lc
## 9 Northern fur seal Callorhinus carni Carnivora      vu
## 10      Dog        Canis carni Carnivora domesticated
## ...
## Variables not shown: sleep_total (dbl), sleep_rem (dbl), sleep_cycle
## (dbl), awake (dbl), brainwt (dbl), bodywt (dbl)

```

**Assessment** Select all columns names with the characters “sleep” and arrange the rows for the `sleep_rem` in a decreasing order.

Hint: look at the `?arrange` help file for the `desc()` option.

```
## Provide your code here
```

```

msleep %>%
  select(matches("sleep")) %>%
  arrange(desc(sleep_rem))

```

```

## Source: local data frame [83 x 3]
##
##      sleep_total sleep_rem sleep_cycle
##      (dbl)      (dbl)      (dbl)
## 1      19.4      6.6      NA
## 2      18.1      6.1      NA
## 3      18.0      4.9  0.3333333
## 4      19.7      3.9  0.1166667
## 5      10.1      3.5  0.2833333

```

```

## 6      13.8      3.4  0.2166667
## 7      12.5      3.2  0.4166667
## 8      17.4      3.1  0.3833333
## 9      14.3      3.1  0.2000000
## 10     15.9      3.0    NA
## ...    ...      ...

```

**Assessment** Select three columns from `msleep` (`name`, `order`, `sleep_total`), arrange the rows in the `sleep_total` column in a descending order, and filter the rows for mammals that sleep for a total of 16 or more hours.

```
## Provide your code here
```

```

msleep %>%
  select(name, order, sleep_total) %>%
  arrange(order, desc(sleep_total)) %>%
  filter(sleep_total >= 16)

```

```

## Source: local data frame [8 x 3]
##
##           name      order sleep_total
##           (chr)      (chr)      (dbl)
## 1 Little brown bat Chiroptera     19.9
## 2 Big brown bat   Chiroptera     19.7
## 3 Giant armadillo Cingulata     18.1
## 4 Long-nosed armadillo Cingulata     17.4
## 5 Thick-tailed opossum Didelphimorphia 19.4
## 6 North American Opossum Didelphimorphia 18.0
## 7 Owl monkey       Primates      17.0
## 8 Arctic ground squirrel Rodentia     16.6

```

### Create summaries of the data frame using `summarise()`

The `summarise()` function will create summary statistics for a given column in the data frame such as finding the mean. For example, to compute the average number of hours of sleep, apply the `mean()` function to the column `sleep_total` and call the summary value `avg_sleep`.

```

msleep %>%
  summarise(avg_sleep = mean(sleep_total))

```

```

## Source: local data frame [1 x 1]
##
##   avg_sleep
##   (dbl)
## 1 10.43373

```

There are many other summary statistics you could consider such `sd()`, `min()`, `max()`, `median()`, `sum()`, `n()` (returns the length of vector), `first()` (returns first value in vector), `last()` (returns last value in vector) and `n_distinct()` (number of distinct values in vector).

**Assessment** Summarize `sleep_total` column in the `msleep` data set with the average sleep, the minimum and maximum amount of sleep, and the total number of mammals.

```
## Provide your code here

msleep %>%
  summarise(avg_sleep = mean(sleep_total),
            min_sleep = min(sleep_total),
            max_sleep = max(sleep_total),
            total = n())

## Source: local data frame [1 x 4]
##
##   avg_sleep min_sleep max_sleep total
##   (dbl)      (dbl)      (dbl) (int)
## 1 10.43373     1.9      19.9     83
```

### Group operations using `group_by()`

The `group_by()` verb is an important function in `dplyr`. As we mentioned before it's related to concept of "split-apply-combine". We literally want to split the data frame by some variable (e.g. `vore`), apply a function to the individual data frames and then combine the output.

Say we wanted to calculate the standard deviation of the body and brain weights for each of factor in the `vore` column. First, we can look at the types of factors in the `vore` column

```
table(msleep$vore)

##
##   carni    herbi insecti    omni
##   19       32      5       20
```

Then, we could use `filter()` to filter for rows that contain "carni" in the `vore` column and summarize with the mean of the brain and body weights.

```
msleep %>%
  filter(vore == "carni") %>%
  summarize("bodywt_sd" = mean(bodywt),
            "brainwt_sd" = mean(brainwt, na.rm = TRUE))

## Source: local data frame [1 x 2]
##
##   bodywt_sd brainwt_sd
##   (dbl)      (dbl)
## 1 90.75111  0.07925556
```

We could repeat this for each factor in `vore`, which is a bit tedious. Instead, we could use this using the `group_by()` function.

Let's do that: split the `msleep` data frame by the `vore` column, then calculate the mean of body weight and brain weight for each individual data frame. (hint: We expect a set of summary statistics for each level in `vore`.)

```

msleep %>%
  group_by(vore) %>%
  summarise("bodywt_sd" = mean(bodywt),
            "brainwt_sd" = mean(brainwt, na.rm = TRUE))

```

```

## Source: local data frame [5 x 3]
##
##      vore bodywt_sd brainwt_sd
##      (chr)      (dbl)      (dbl)
## 1  carni  90.75111  0.07925556
## 2  herbi 366.87725  0.62159750
## 3 insecti 12.92160  0.02155000
## 4    omni 12.71800  0.14573118
## 5      NA  0.85800  0.00762600

```

**Assessment** Split the `msleep` data frame by the taxonomic order, then for each taxonomic order summarize the `sleep_total` with the average sleep, the minimum and maximum amount of sleep, and the total number of mammals in each order.

```
## Provide your code here
```

```

msleep %>%
  group_by(order) %>%
  summarise(avg_sleep = mean(sleep_total),
            min_sleep = min(sleep_total),
            max_sleep = max(sleep_total),
            total = n())

```

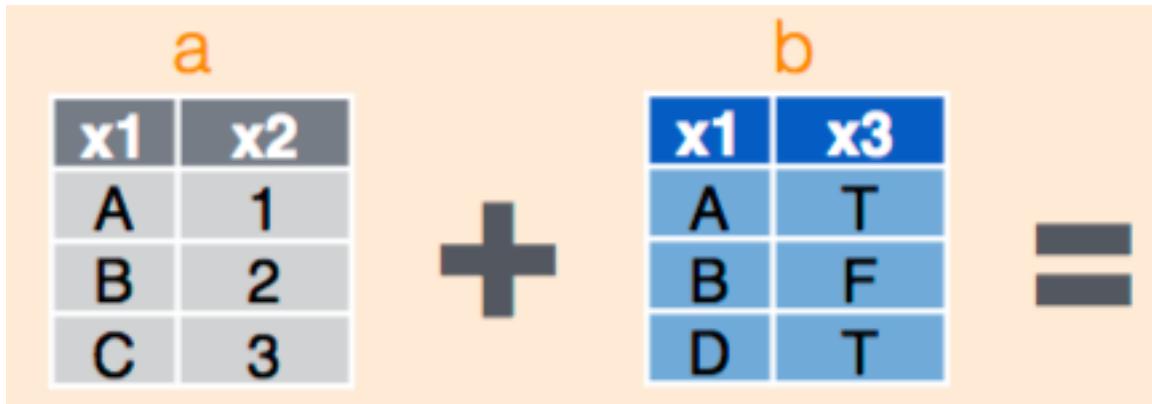
```

## Source: local data frame [19 x 5]
##
##      order avg_sleep min_sleep max_sleep total
##      (chr)      (dbl)      (dbl)      (dbl) (int)
## 1  Afrosoricida 15.600000     15.6     15.6     1
## 2   Artiodactyla  4.516667      1.9      9.1     6
## 3    Carnivora 10.116667      3.5     15.8    12
## 4     Cetacea  4.500000      2.7      5.6     3
## 5   Chiroptera 19.800000     19.7     19.9     2
## 6    Cingulata 17.750000     17.4     18.1     2
## 7 Didelphimorphia 18.700000     18.0     19.4     2
## 8 Diprotodontia 12.400000     11.1     13.7     2
## 9  Erinaceomorpha 10.200000     10.1     10.3     2
## 10   Hyracoidea  5.666667      5.3      6.3     3
## 11   Lagomorpha  8.400000      8.4      8.4     1
## 12  Monotremata  8.600000      8.6      8.6     1
## 13 Perissodactyla  3.466667      2.9      4.4     3
## 14      Pilosa 14.400000     14.4     14.4     1
## 15     Primates 10.500000      8.0     17.0    12
## 16 Proboscidea  3.600000      3.3      3.9     2
## 17    Rodentia 12.468182      7.0     16.6    22
## 18   Scandentia  8.900000      8.9      8.9     1
## 19 Soricomorpha 11.100000      8.4     14.9     5

```

## joining two data frames in dplyr

The last part of `dplyr` that we will discuss are a set of `dplyr` verbs that allow you to join two data sets. The following are cartoons extracted from [Data Wrangling with dplyr and tidyr Cheatsheet](#) from RStudio to illustrate the different ways to join data frames using `dplyr`.



Two types of functions to join together data frames

### Mutating Joins

<b>x1</b>	<b>x2</b>	<b>x3</b>
A	1	T
B	2	F
C	3	NA

**dplyr::left\_join(a, b, by = "x1")**

Join matching rows from b to a.

<b>x1</b>	<b>x3</b>	<b>x2</b>
A	T	1
B	F	2
D	T	NA

**dplyr::right\_join(a, b, by = "x1")**

Join matching rows from a to b.

<b>x1</b>	<b>x2</b>	<b>x3</b>
A	1	T
B	2	F

**dplyr::inner\_join(a, b, by = "x1")**

Join data. Retain only rows in both sets.

<b>x1</b>	<b>x2</b>	<b>x3</b>
A	1	T
B	2	F
C	3	NA
D	NA	T

**dplyr::full\_join(a, b, by = "x1")**

Join data. Retain all values, all rows.

## Filtering Joins

x1	x2
A	1
B	2

**dplyr::semi\_join(a, b, by = "x1")**

All rows in a that have a match in b.

x1	x2
C	3

**dplyr::anti\_join(a, b, by = "x1")**

All rows in a that do not have a match in b.

## Data

For this section, we will work with two small data sets related to the 2016 Oscars Nominations. The two data sets are `oscars` and `movies`. The first data set contains information about the name of the actor/actress, the name of the movie and the category for nomination. The second data set contains a list of movies and the length of the movie in minutes.

We will explore the `dplyr` verbs to join the two tables. First let's load the data.

```
library(readr)

oscars <- "
name,movie,category
Adam McKay,The Big Short,Best Director
Alejandro González Iñárritu,The Revenant,Best Director
Lenny Abrahamson,Room,Best Director
Tom McCarthy,Spotlight,Best Director
George Miller,Mad Max: Fury Road,Best Director
Bryan Cranston,Trumbo,Best Actor
Matt Damon,The Martian,Best Actor
Michael Fassbender,Steve Jobs,Best Actor
Leonardo DiCaprio,The Revenant,Best Actor
Eddie Redmayne,The Danish Girl,Best Actor
Cate Blanchett,Carol,Best Actress
Brie Larson,Room,Best Actress
Jennifer Lawrence,Joy,Best Actress
Charlotte Rampling,45 Years,Best Actress
Saoirse Ronan,Brooklyn,Best Actress
"
oscars <- read_csv(oscars, trim_ws = TRUE, skip = 1)
oscars

## Source: local data frame [15 x 3]
##
##           name      movie   category
##           (chr)    (chr)     (chr)
## 1       Adam McKay The Big Short Best Director
## 2 Alejandro González Iñárritu The Revenant Best Director
## 3        Lenny Abrahamson          Room Best Director
## 4      Tom McCarthy      Spotlight Best Director
## 5      George Miller Mad Max: Fury Road Best Director
```

```

## 6          Bryan Cranston      Trumbo      Best Actor
## 7          Matt Damon        The Martian  Best Actor
## 8          Michael Fassbender Steve Jobs    Best Actor
## 9          Leonardo DiCaprio The Revenant Best Actor
## 10         Eddie Redmayne   The Danish Girl Best Actor
## 11         Cate Blanchett    Carol        Best Actress
## 12         Brie Larson       Room        Best Actress
## 13         Jennifer Lawrence Joy          Best Actress
## 14         Charlotte Rampling 45 Years    Best Actress
## 15         Saoirse Ronan     Brooklyn   Best Actress

```

```

movies <- "
movie,length_mins
The Big Short,130
Star Wars: The Force Awakens,135
Brooklyn,111
Mad Max: Fury Road,120
Room,118
The Martian,144
The Revenant,156
Spotlight,128
"
movies <- read_csv(movies, trim_ws = TRUE, skip = 1)
movies

```

```

## Source: local data frame [8 x 2]
##
##               movie length_mins
##               (chr)      (int)
## 1          The Big Short      130
## 2 Star Wars: The Force Awakens 135
## 3           Brooklyn      111
## 4          Mad Max: Fury Road 120
## 5             Room      118
## 6          The Martian      144
## 7          The Revenant      156
## 8           Spotlight      128

```

```
inner_join(x,y)
```

This function joins all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.

```
inner_join(oscars, movies, by = "movie")
```

```

## Source: local data frame [9 x 4]
##
##               name      movie      category length_mins
##               (chr)      (chr)      (chr)      (int)
## 1          Adam McKay The Big Short Best Director      130
## 2 Alejandro González Iñárritu The Revenant Best Director      156
## 3          Lenny Abrahamson           Room Best Director      118

```

```

## 4           Tom McCarthy      Spotlight Best Director    128
## 5           George Miller Mad Max: Fury Road Best Director 120
## 6           Matt Damon       The Martian   Best Actor    144
## 7           Leonardo DiCaprio The Revenant   Best Actor    156
## 8           Brie Larson      Room     Best Actress   118
## 9           Saoirse Ronan   Brooklyn   Best Actress   111

```

### `semi_join(x,y)`

This function returns all rows from x where there are matching values in y, keeping just columns from x. A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, where a semi join will never duplicate rows of x.

```
semi_join(oscars, movies, by = "movie")
```

```

## Source: local data frame [9 x 3]
##
##           name          movie      category
##           (chr)        (chr)      (chr)
## 1       Adam McKay The Big Short Best Director
## 2   Saoirse Ronan    Brooklyn Best Actress
## 3   George Miller Mad Max: Fury Road Best Director
## 4 Lenny Abrahamson        Room Best Director
## 5       Brie Larson        Room Best Actress
## 6       Matt Damon       The Martian Best Actor
## 7 Alejandro González Iñárritu The Revenant Best Director
## 8   Leonardo DiCaprio The Revenant   Best Actor
## 9       Tom McCarthy      Spotlight Best Director

```

**Assessment** Try applying the `semi_join()` function with `x=movies` and `y=oscars`. What is the difference?

```
## Provide your code here
```

```
semi_join(movies, oscars, by = "movie")
```

```

## Source: local data frame [7 x 2]
##
##           movie length_mins
##           (chr)      (int)
## 1       The Big Short      130
## 2       The Revenant      156
## 3           Room         118
## 4       Spotlight        128
## 5 Mad Max: Fury Road     120
## 6       The Martian       144
## 7       Brooklyn        111

```

### **Assessment**

Using the `dplyr` join functions, combine the columns from the `oscars` and `movies` data sets and return all rows from the `oscars` data set and all columns in both the `oscars` and `movies` columns.

Hint: read the help file for `left_join()` or `right_join()`.

```
## Provide your code here

left_join(oscars, movies, by = "movie")

## Source: local data frame [15 x 4]
##
##          name      movie category
##          (chr)    (chr)   (chr)
## 1      Adam McKay  The Big Short Best Director
## 2 Alejandro González Iñárritu  The Revenant Best Director
## 3      Lenny Abrahamson      Room Best Director
## 4      Tom McCarthy  Spotlight Best Director
## 5     George Miller Mad Max: Fury Road Best Director
## 6    Bryan Cranston      Trumbo Best Actor
## 7      Matt Damon  The Martian Best Actor
## 8 Michael Fassbender  Steve Jobs Best Actor
## 9 Leonardo DiCaprio  The Revenant Best Actor
## 10     Eddie Redmayne  The Danish Girl Best Actor
## 11     Cate Blanchett      Carol Best Actress
## 12      Brie Larson      Room Best Actress
## 13  Jennifer Lawrence       Joy Best Actress
## 14  Charlotte Rampling  45 Years Best Actress
## 15  Saoirse Ronan  Brooklyn Best Actress
## Variables not shown: length_mins (int)

## or
right_join(movies, oscars, by = "movie")

## Source: local data frame [15 x 4]
##
##      movie length_mins      name
##      (chr)      (int)    (chr)
## 1  The Big Short      130 Adam McKay
## 2  The Revenant      156 Alejandro González Iñárritu
## 3      Room         118 Lenny Abrahamson
## 4  Spotlight        128 Tom McCarthy
## 5 Mad Max: Fury Road      120 George Miller
## 6      Trumbo        NA Bryan Cranston
## 7  The Martian        144 Matt Damon
## 8  Steve Jobs        NA Michael Fassbender
## 9  The Revenant      156 Leonardo DiCaprio
## 10 The Danish Girl        NA Eddie Redmayne
## 11      Carol        NA Cate Blanchett
## 12      Room         118 Brie Larson
## 13       Joy        NA Jennifer Lawrence
## 14  45 Years        NA Charlotte Rampling
## 15  Brooklyn        111 Saoirse Ronan
## Variables not shown: category (chr)
```

Why are there NAs?

## Assessment

Using the `dplyr` join functions, combine the columns from the `oscars` and `movies` data sets and return all rows from the `movies` data set and all columns in both the `oscars` and `movies` columns.

```
## Provide your code here
```

```
right_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [10 x 4]
##
##          name      movie category
##          (chr)    (chr)   (chr)
## 1      Adam McKay  The Big Short Best Director
## 2      NA Star Wars: The Force Awakens NA
## 3     Saoirse Ronan Brooklyn Best Actress
## 4     George Miller Mad Max: Fury Road Best Director
## 5  Lenny Abrahamson Room Best Director
## 6      Brie Larson Room Best Actress
## 7      Matt Damon The Martian Best Actor
## 8 Alejandro González Iñárritu The Revenant Best Director
## 9      Leonardo DiCaprio The Revenant Best Actor
## 10     Tom McCarthy Spotlight Best Director
## Variables not shown: length_mins (int)
```

```
## or
```

```
left_join(movies, oscars, by = "movie")
```

```
## Source: local data frame [10 x 4]
##
```

```
##          movie length_mins      name
##          (chr)      (int)    (chr)
## 1      The Big Short       130 Adam McKay
## 2  Star Wars: The Force Awakens       135 NA
## 3      Brooklyn        111 Saoirse Ronan
## 4      Mad Max: Fury Road       120 George Miller
## 5          Room        118 Lenny Abrahamson
## 6          Room        118 Brie Larson
## 7      The Martian       144 Matt Damon
## 8      The Revenant      156 Alejandro González Iñárritu
## 9      The Revenant      156 Leonardo DiCaprio
## 10     Spotlight        128 Tom McCarthy
## Variables not shown: category (chr)
```

```
full_join(x,y)
```

This function returns all rows and all columns from both x and y. When there are not matching values, it will return NA for the one missing.

```
full_join(oscars, movies, by = "movie")
```

```

## Source: local data frame [16 x 4]
##
##          name      movie   category
##          (chr)     (chr)      (chr)
## 1      Adam McKay The Big Short Best Director
## 2 Alejandro González Iñárritu The Revenant Best Director
## 3      Lenny Abrahamson Room Best Director
## 4      Tom McCarthy Spotlight Best Director
## 5      George Miller Mad Max: Fury Road Best Director
## 6      Bryan Cranston Trumbo Best Actor
## 7      Matt Damon The Martian Best Actor
## 8 Michael Fassbender Steve Jobs Best Actor
## 9      Leonardo DiCaprio The Revenant Best Actor
## 10     Eddie Redmayne The Danish Girl Best Actor
## 11     Cate Blanchett Carol Best Actress
## 12     Brie Larson Room Best Actress
## 13 Jennifer Lawrence Joy Best Actress
## 14 Charlotte Rampling 45 Years Best Actress
## 15 Saoirse Ronan Brooklyn Best Actress
## 16 NA Star Wars: The Force Awakens NA
## Variables not shown: length_mins (int)

```

```
full_join(movies, oscars, by = "movie")
```

```

## Source: local data frame [16 x 4]
##
##          movie length_mins      name
##          (chr)      (int)      (chr)
## 1      The Big Short      130 Adam McKay
## 2 Star Wars: The Force Awakens      135 NA
## 3      Brooklyn      111 Saoirse Ronan
## 4      Mad Max: Fury Road      120 George Miller
## 5      Room      118 Lenny Abrahamson
## 6      Room      118 Brie Larson
## 7      The Martian      144 Matt Damon
## 8      The Revenant     156 Alejandro González Iñárritu
## 9      The Revenant     156 Leonardo DiCaprio
## 10     Spotlight      128 Tom McCarthy
## 11     Trumbo       NA Bryan Cranston
## 12     Steve Jobs       NA Michael Fassbender
## 13 The Danish Girl       NA Eddie Redmayne
## 14      Carol       NA Cate Blanchett
## 15      Joy       NA Jennifer Lawrence
## 16     45 Years       NA Charlotte Rampling
## Variables not shown: category (chr)

```

## Assessment

Using the `dplyr` join functions, return all rows from `oscars` data set where there are not matching values in `movies`, only keeping the columns from the `oscars` data set.

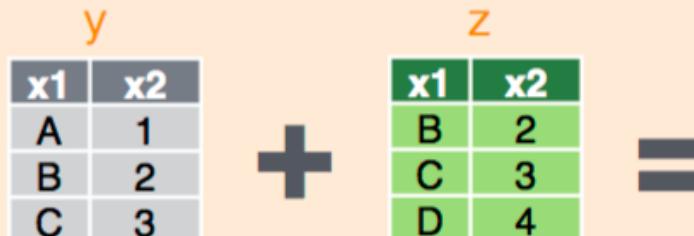
Hint: Read the help file for `anti_join()`.

```
## Provide your code here

anti_join(oscars, movies, by = "movie")
```

```
## Source: local data frame [6 x 3]
##
##          name      movie   category
##          (chr)    (chr)     (chr)
## 1 Charlotte Rampling      45 Years Best Actress
## 2 Jennifer Lawrence        Joy Best Actress
## 3 Cate Blanchett         Carol Best Actress
## 4 Eddie Redmayne The Danish Girl Best Actor
## 5 Michael Fassbender    Steve Jobs Best Actor
## 6 Bryan Cranston        Trumbo Best Actor
```

Other functions in dplyr to join together data frames



## Set Operations

x1	x2
B	2
C	3

### dplyr::intersect(y, z)

Rows that appear in both y and z.

x1	x2
A	1
B	2
C	3
D	4

### dplyr::union(y, z)

Rows that appear in either or both y and z.

x1	x2
A	1

### dplyr::setdiff(y, z)

Rows that appear in y but not z.

## Binding

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

**dplyr::bind\_rows(y, z)**

Append z to y as new rows.

**dplyr::bind\_cols(y, z)**

Append z to y as new columns.

Caution: matches rows by position.

## Cheatsheets

- Data Wrangling with dplyr and tidyr from RStudio

# Data Wrangling

## Data Wrangling

In the real world, data science projects rarely involve data that can be easily imported ready for analysis. According to Wikipedia:

Data munging or data wrangling is loosely the process of manually converting or mapping data from one “raw” form into another format that allows for more convenient consumption of the data with the help of semi-automated tools.

Our example dataset provides an example:

```
url <- "https://raw.githubusercontent.com/datasets/bio260-heights.csv"
dat <- read.csv(url)
```

First note how we make assignments in R: we use `<-`. We can also use the equal sign `=` although here we try to stick to `<-` to make it very clear it is an assignment and not logical statement.

We also note that we have put the content of what comes out of `read.csv` into an *object*. We picked the object name `dat`.

So what is `dat` exactly? We can get a quick summary of what an object is with the function `str` (stands for structure)

```
str(dat)
```

```
## 'data.frame':    148 obs. of  3 variables:
##   $ Timestamp           : Factor w/ 143 levels "1/25/2016 14:39:56",...: 9 10 11 12 13 13 ...
##   $ What.is.your.gender.: Factor w/ 3 levels "Female","I prefer not to disclose",...: 1 1 1 ...
##   $ What.is.your.height..in.inches.: Factor w/ 67 levels "167","168","172",...: 38 35 53 50 58 64 52 65 ...
```

Here we see that this object is a `data.frame`. These are one of the most widely used data types in R. They are particularly useful for storing tables.

To see more of this object we can type it

Now we want to describe the heights. We could simply report the list of numbers. But there is a problem. Take a look at the entries:

```
View(dat)
```

Notice these not all entries are numbers. Furthermore, they are not all in inches. So what to do? We need to wrangle

**Extracting columns** To extract columns from the `data.frame` we use the `$` character like this:

```
dat$Timestamp
```

This now gives us a vector. We can access elements of the vector using the `[` symbol:

```
dat$Timestamp[2]
```

```
## [1] 1/25/2016 8:15:21  
## 143 Levels: 1/25/2016 14:39:56 1/25/2016 16:32:52 ... 1/27/2016 9:34:14
```

**Quick Review of Vectors** Vector are a sequence of data elements of the same type. Many of the operations used to analyze data are applied to vectors. In R vectors can be numeric, characters or logical.

The most basic way to creat a vector is with the function `c`

```
x <- c(1,2,3,4,5)
```

Two very common ways of generating vectors are using `:` or the `seq` function:

```
x <- 1:5  
x <- seq(1,5)
```

Vecotrs can have names

```
names(x) <- letters[1:5]  
x
```

```
## a b c d e  
## 1 2 3 4 5
```

**Coercion** Vectors need to be homogenous. But when R is instructed to create a vector of different types, it does not give an error. Instead it tries to *coerce* values to be the same. Here is an example:

```
height <- c(60, 59, 55, "5'5", 70)  
height
```

```
## [1] "60"  "59"  "55"  "5'5" "70"
```

Note that no warning or error was given. It simply changed everything to a character. This is important to know because sometimes we make a mistake in entering data and receive no error message.

## Data Manipulation wiht `dplyr`

R provides incredibly powerful and flexible language for data manipulation. However, the syntax is somewhat hard to get used to. We will therefore introducing a package that makes the syntax much more like the English language. This package is `dplyr` which you should install if you have not done so already.

```
library(dplyr)
```

When using `dplyr` we recommend reading in data with the functions in the `readr` package:

```
library(readr)  
dat <- read_csv("https://raw.githubusercontent.com/datasets/country-data/master/country-data.csv")
```

This object is now a special type of `data.frame` called `tbl_df` that has a nicer printing method. We can now simply evaluate an expression with just the object and see a meaningful summary instead of everything.

```
dat
```

```
## Source: local data frame [148 x 3]
##
##           Timestamp What is your gender? What is your height (in inches)?
##           (chr)          (chr)          (chr)
## 1  1/25/2016 8:15:15      Female          63
## 2  1/25/2016 8:15:21      Female          62
## 3  1/25/2016 8:15:25       Male          69
## 4  1/25/2016 8:15:29      Female          68
## 5  1/25/2016 8:15:37       Male         71.65
## 6  1/25/2016 8:15:37       Male          75
## 7  1/25/2016 8:15:39       Male        68.8976
## 8  1/25/2016 8:15:40       Male          74
## 9  1/25/2016 8:15:41      Female          65
## 10 1/25/2016 8:15:44      Female        5'4
## ...             ...          ...        ...
```

**Selecting columns** Right, we are interested in looking at heights. We can select just that column using:

```
select(dat, contains("height"))
```

```
## Source: local data frame [148 x 1]
##
##           What is your height (in inches)?
##           (chr)
## 1                  63
## 2                  62
## 3                  69
## 4                  68
## 5                 71.65
## 6                  75
## 7                68.8976
## 8                  74
## 9                  65
## 10                 5'4
## ...             ...
```

We have a problem: this is a `character`. We want numbers.

## Renaming columns

Before we continue it will be convenient to change the names of our columns to something more convenient.

```
names(dat) <- c("time", "gender", "height")
```

## Vectorization

```
height <- c(60, 59, 55, "5'5", 70)
height[3]
```

```
## [1] "55"
```

```
as.numeric(height[3])
```

```
## [1] 55
```

One powerful feature of R is that we can *vectorize* most operation

```
as.numeric(height)
```

```
## Warning: NAs introduced by coercion
```

```
## [1] 60 59 55 NA 70
```

Note now we do receive an warning. This is because R has no idea how to convert “5’5” to a number.

## Missing values

Note in the the `NA` value in the object above.

These are missing values. We can find out which values are missing using the function

```
?is.na
```

## Adding columns

```
dat <- mutate(dat, numeric_height=as.numeric(height),
              original=height)
```

```
## Warning in eval(substitute(expr), envir, enclos): NAs introduced by
## coercion
```

## Subsetting Observations

To see all the row in which we have problems:

```
filter(dat, is.na(numeric_height))
```

```

## Source: local data frame [21 x 5]
##
##           time             gender height numeric_height
##           (chr)          (chr)   (dbl)
## 1  1/25/2016 8:15:44 Female    5'4        NA
## 2  1/25/2016 8:15:45 Female    5'8"       NA
## 3  1/25/2016 8:15:45 Female    5'5        NA
## 4  1/25/2016 8:29:00 Male     5'7        NA
## 5  1/25/2016 14:39:56 Female    5'6        NA
## 6  1/25/2016 22:02:03 Male     5'11"      NA
## 7  1/26/2016 8:36:33 I prefer not to disclose 5'10"
## 8  1/26/2016 9:49:15 Male     5'7"       NA
## 9  1/26/2016 9:49:19 Male     5'7        NA
## 10 1/26/2016 9:51:19 Female    5'8        NA
## .. ...
## Variables not shown: original (chr)

```

## The Pipe

```
filter(dat, is.na(numeric_height)) %>% select(height)
```

```

## Source: local data frame [21 x 1]
##
##       height
##       (chr)
## 1      5'4
## 2      5'8"
## 3      5'5
## 4      5'7
## 5      5'6
## 6      5'11"
## 7      5'10"
## 8      5'7"
## 9      5'7
## 10     5'8
## ...
## ...

```

Let's see more

```
filter(dat, is.na(numeric_height)) %>% select(height) %>% print(n=21)
```

```

## Source: local data frame [21 x 1]
##
##       height
##       (chr)
## 1      5'4
## 2      5'8"
## 3      5'5
## 4      5'7
## 5      5'6
## 6      5'11"

```

```

## 7      5'10"
## 8      5'7"
## 9      5'7
## 10     5'8
## 11     5' 11"
## 12     6'1"
## 13     69"
## 14     5' 7"
## 15     5'10"
## 16     5'10
## 17     5'10
## 18 5ft 9 inches
## 19 5 ft 9 inches
## 20     5'2
## 21     5'11

```

**gsub** One of the most useful functions for data wrangling is **gsub**. It lets us search for characters and substitutes it for others. More generally it searches for regular expression. We will learn about those later.

Here is an example:

```

x <- dat$height[109:116]
x

## [1] "5'10"          "70"           "67.7"          "62"
## [5] "5ft 9 inches" "5 ft 9 inches" "5'2"           "74"

```

Note that we are using both ' and ft as the same thing. To simplify the problem we want to substitute one for the other. **gsub** does the trick:

```

x <- gsub("ft", "", x)
x

## [1] "5'10"          "70"           "67.7"          "62"
## [5] "5' 9 inches"  "5 ' 9 inches" "5'2"           "74"

```

The word inches is not doing anything here so we might as well remove it.

```

x <- gsub("inches","",x)
x

## [1] "5'10"      "70"        "67.7"      "62"       "5' 9 "    "5 ' 9 "  "5'2"      "74"

```

We are now ready to start fixing the height data:

```

dat <- mutate(dat, height= gsub("ft","",height) ) %>%
  mutate(height= gsub("\\"|inches|\\"","" ,height) )

```

## Functions

Up to now we have used prebuilt functions. However, many times we have to construct our own. We can do this in R using the `function`:

```
avg <- function(x){  
  return( sum(x) / length(x) )  
}  
avg( 1:5 )  
  
## [1] 3
```

Assessment: Construct a function that computes the variance defined as follows for a vector  $x_1, \dots, x_n$ :

$$\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \text{ with } \mu = \frac{1}{n} \sum_{i=1}^n x_i$$

What is the variance of 1:5 ?

Assessment: Write a function `convert` that takes two character arguments, feet and inches as characters, and returns inches

Here we construct a more complicated function that changes 5'4 to 5\*12+4

```
fixheight <- function(x){  
  y <- strsplit(x, "'")  
  ret <- sapply(y, function(z){  
    ifelse( length(z)>1, as.numeric(z[1])*12 + as.numeric(z[2]) ,  
           as.numeric(z[1]))  
  })  
  return(ret)  
}
```

We can now test the function

```
fixheight( "70")  
  
## [1] 70  
  
fixheight( "5'10")  
  
## [1] 70  
  
fixheight( c("5'9","70","5'11"))  
  
## [1] 69 70 71
```

Finally we can mutate our data:

```
dat <- mutate(dat, height=fixheight(height)) %>% select(-numeric_height)
```

The last call to select removes the now unnecessary column `numeric_height`. Let's see the result:

```
filter(dat, is.na(height)) %>% select(height)
```

```
## Source: local data frame [0 x 1]
##
## Variables not shown: height (dbl)
```

We have removed all the NAs

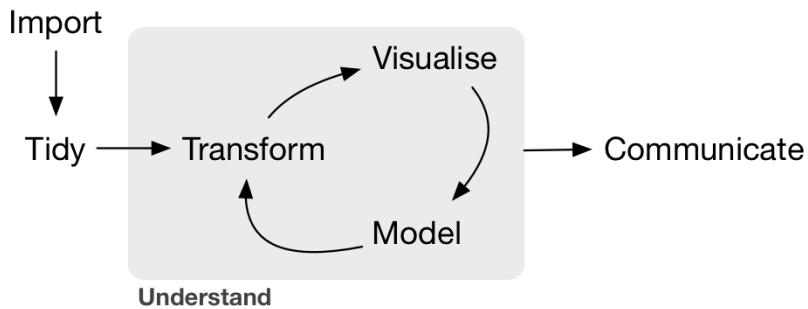
# Data Wrangling with `tidyverse`

Stephanie Hicks, Rafael Irizarry

The data analysis process can be thought about in four parts

1. Data cleaning
2. Data transformation
3. Data visualization
4. Modeling

where we each of these steps need their own tools and software to complete.



As we have seen in class, one of the most time-consuming aspects of the data analysis process is “data wrangling”. This is also known as “data munging”, which is a trendy term for *cleaning up a messy data set*. This refers to the first two steps in the data analysis process:

1. Data cleaning (or tidying data)
2. Data transformation

It can take a long time to clean and transform messy data into a format that is useful for data visualization and modeling, but there are tools that can help turn messy data into clean data.

## Defining data structures

There are many ways to define the structure of a data set. Most data frames are made up of **rows** and **columns** where the columns are almost always labeled and the rows are *sometimes* labeled.

For example, a data set could be structured in the following way:

- each row represents one company (row names are companies)
- each column represent one time point
- the stock prices are defined for each row/column pair

> `stocks`

	2016-01-01	2016-01-02	2016-01-03	2016-01-04	2016-01-05
Google	99.43952	99.76982	101.55871	100.07051	100.12929
Facebook	103.43013	100.92183	97.46988	98.62629	99.10868
Twitter	104.89633	101.43926	101.60309	100.44273	97.77664

Alternatively, a data set can be structured in the following way:

- each row represents one time point (but no row names)
- the first column defines the time variable and the last three columns contain the stock prices for three companies

## > stocks

	time	Google	Facebook	Twitter
1	2016-01-01	99.43952	103.43013	104.89633
2	2016-01-02	99.76982	100.92183	101.43926
3	2016-01-03	101.55871	97.46988	101.60309
4	2016-01-04	100.07051	98.62629	100.44273
5	2016-01-05	100.12929	99.10868	97.77664

In both cases, the data is the same, but the structure is different. This can be *frustrating* to deal with because the meaning of the values (rows and columns) in the two data sets are different. Providing a standardized way of organizing values within a data set would alleviate a major portion of this frustration.

## Defining tidy data

Now, we will introduce the concept of **tidy** data. Tidy data is a standard way of mapping the meaning of a dataset to its structure. The properties of a tidy data set are based on:

- Each column is a variable
- Each rows is an observation

Working with tidy data is useful because it creates a structured way of organizing data values within a data set. This makes the data analysis process more efficient and simplifies the development of data analysis tools that work together. In this way, you can focus on the problem you are investigating, rather than the uninteresting logistics of data.

## What is **tidyR**?

**tidyR** is an R package that transforms data sets to a tidy format.

There are two main functions in **tidyR**:

- **gather()** = takes multiple columns, and gathers them into key-value pairs (it makes “wide” data longer)
- **separate()** = turns a single character column into multiple columns (it makes “long” data wider)

We’ll explore what it means to go between a “wide” and “long” data format using **gather()** and **separate()** next.

## How do I get **tidyR**?

To install **tidyR**

```
install.packages("tidyverse")
```

To load `tidyverse` and we'll need `dplyr`

```
library(tidyr)  
library(dplyr)
```

For motivation, a tidy version of the stock data we looked at above looks like this: (we'll learn how the functions work in just a moment)

```
> stocks %>%
+   gather(company, price, Google:Twitter)
#> # Source: local dataset
#> # Rowwise: time
#>
#> # # A tibble: 15 x 4
#> #   time     company    price
#> #   <date>   <chr>      <dbl>
#> # 1 2016-01-01 Google 99.43952
#> # 2 2016-01-02 Google 99.76982
#> # 3 2016-01-03 Google 101.55871
#> # 4 2016-01-04 Google 100.07051
#> # 5 2016-01-05 Google 100.12929
#> # 6 2016-01-01 Facebook 103.43013
#> # 7 2016-01-02 Facebook 100.92183
#> # 8 2016-01-03 Facebook 97.46988
#> # 9 2016-01-04 Facebook 98.62629
#> # 10 2016-01-05 Facebook 99.10868
#> # 11 2016-01-01 Twitter 104.89633
#> # 12 2016-01-02 Twitter 101.43926
#> # 13 2016-01-03 Twitter 101.60309
#> # 14 2016-01-04 Twitter 100.44273
#> # 15 2016-01-05 Twitter 97.77664
```

In this “tidy” data set, we have three columns representing three variables (time, company name and stock price). Every row represents contains one stock price from a particular time and for a specific company.

Pipe operator: %>%

We have introduced the operator: `%>%`. dplyr imports this operator from another package ([magrittr see help file here](#)). This operator allows you to pipe the output from one function to the input of another function. Instead of nesting functions (reading from the inside to the outside), the idea of piping is to read the functions from left to right.

Now in this case, we pipe the `stocks` data frame to the function that will gather multiple columns into key-value pairs.

Data

## 2016 Iowa Presidential Caucus

We will explore public poll data from HuffPost Pollster from the 2016 Iowa Democratic and Republican Presidential Caucus.

First we will read in the data:

```
library(readr)  
dem_polls = read_csv("http://elections.huffingtonpost.com/pollster/2016-iowa-presidential-democratic-canvass.csv")  
rep_polls = read_csv("http://elections.huffingtonpost.com/pollster/2016-iowa-presidential-republican-canvass.csv")
```

Let's take a look at data

`View(dem_polls)`

```
View(rep_polls)
```

```
glimpse(dem_polls)
```

```
glimpse(rep_polls)
```

We see there is a lot of information in each data frame. First let's use `dplyr` to select a subset of the columns.

```
dem_polls <- dem_polls %>%
  select(Pollster, `End Date`, Clinton:Undecided)

rep_polls <- rep_polls %>%
  select(Pollster, `End Date`, Trump:Walker)
```

In the democratic and republican polling data sets, there is one column representing the polling percentages for each candidate, similar to the stock price data set with multiple columns representing different companies. To **tidy** it, we need to *gather* these columns into a two-column *key-value* pair. This is often described as transforming a *wide* data set into a *long* data set.

## gather()

This function gathers multiple columns and collapses them into new *key-value* pairs. This transform data from *wide* format into a *long* format.

- The **key** is the name of the *new* column that you are creating which contains the values of the column headings that you are gathering
  - The **value** is the name of the *new* column that will contain the values themselves
  - The third argument defines the columns to gather

```
dem_polls %>%  
  gather(key = candidate, value = percentage, Clinton:Undecided)
```

```
## Source: local data frame [648 x 4]
##
##                                     Pollster   End Date candidate percentage
```

```

##                               (chr)      (date)      (chr)      (int)
## 1      Emerson College Polling Society 2016-01-31 Clinton     51
## 2                                Quinnipiac 2016-01-31 Clinton     46
## 3 Des Moines Register/Bloomberg/Selzer 2016-01-29 Clinton     45
## 4      Gravis Marketing/One America News 2016-01-27 Clinton     53
## 5                  PPP (D-Progress Iowa) 2016-01-27 Clinton     48
## 6                                NBC/WSJ/Marist 2016-01-26 Clinton     48
## 7      Monmouth University 2016-01-26 Clinton     47
## 8                                ARG 2016-01-24 Clinton     45
## 9                                Quinnipiac 2016-01-24 Clinton     45
## 10     Iowa State/WHO-HD 2016-01-22 Clinton     47
## ...
## ...           ...       ...       ...       ...

```

To select a range of columns by name, use the “:” (colon) operator

**Assessment** Using the democratic poll data, apply the `gather()` function to tidy the poll data by *excluding* the Pollster and End Date columns, rather than directly providing the column names to gather.

Hint: Look at the `gather()` help file on how to exclude column names.

```

## Provide your code here

dem_polls %>%
  gather(key = candidate, value = percentage, -c(Pollster, `End Date`))

```

```

## Source: local data frame [648 x 4]
##
##                               Pollster      End Date candidate percentage
##                               (chr)      (date)      (chr)      (int)
## 1      Emerson College Polling Society 2016-01-31 Clinton     51
## 2                                Quinnipiac 2016-01-31 Clinton     46
## 3 Des Moines Register/Bloomberg/Selzer 2016-01-29 Clinton     45
## 4      Gravis Marketing/One America News 2016-01-27 Clinton     53
## 5                  PPP (D-Progress Iowa) 2016-01-27 Clinton     48
## 6                                NBC/WSJ/Marist 2016-01-26 Clinton     48
## 7      Monmouth University 2016-01-26 Clinton     47
## 8                                ARG 2016-01-24 Clinton     45
## 9                                Quinnipiac 2016-01-24 Clinton     45
## 10     Iowa State/WHO-HD 2016-01-22 Clinton     47
## ...
## ...           ...       ...       ...       ...

```

## To select all the columns \*except\* a specific column,  
## use the “-” (subtraction) operator (also known as negative indexing)

**Assessment** Using the “tidy” democratic poll data, use dplyr to filter for only the following candidates (Clinton, Sanders, O’Malley) and for polls only ending after May 1, 2015.

```

## Provide your code here

dem_polls %>%
  gather(key = candidate, value = percentage, Clinton:Undecided) %>%
  filter(candidate %in% c("Clinton", "Sanders", "O'Malley") &
    `End Date` >= "2015-05-01")

```

```

## Source: local data frame [195 x 4]
##
##           Pollster   End Date candidate percentage
##           (chr)      (date)     (chr)      (int)
## 1 Emerson College Polling Society 2016-01-31 Clinton      51
## 2 Quinnipiac 2016-01-31 Clinton      46
## 3 Des Moines Register/Bloomberg/Selzer 2016-01-29 Clinton      45
## 4 Gravis Marketing/One America News 2016-01-27 Clinton      53
## 5 PPP (D-Progress Iowa) 2016-01-27 Clinton      48
## 6 NBC/WSJ/Marist 2016-01-26 Clinton      48
## 7 Monmouth University 2016-01-26 Clinton      47
## 8 ARG 2016-01-24 Clinton      45
## 9 Quinnipiac 2016-01-24 Clinton      45
## 10 Iowa State/WHO-HD 2016-01-22 Clinton      47
## .. ...

```

**Assessment (optional)** Using the tidy and filtered democratic poll data set, use `ggplot2` to plot the results from each poll (percentage) for each of the candidates. Color the lines by the candidate.

```

## Provide your code here

library(ggplot2)

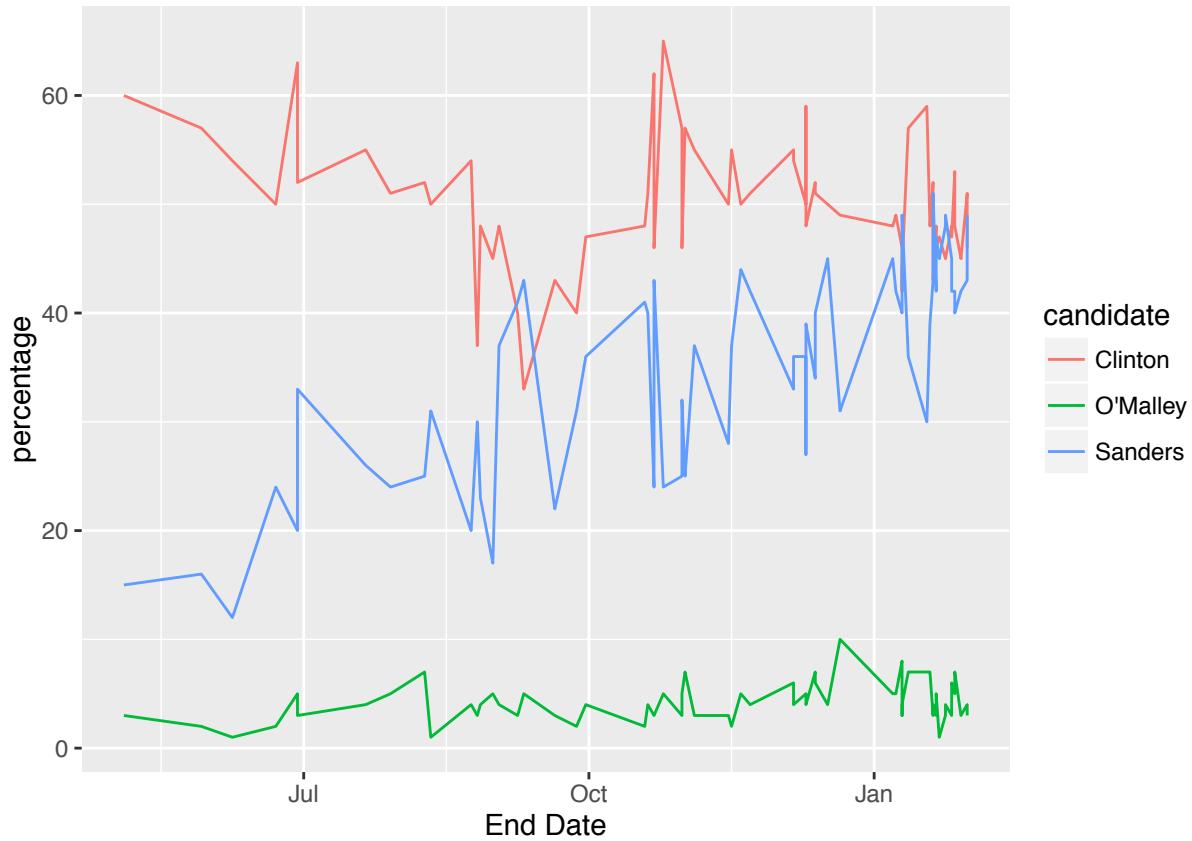
```

```

## Warning: package 'ggplot2' was built under R version 3.2.4

dem_polls %>%
  gather(key = candidate, value = percentage, Clinton:Undecided) %>%
  filter(candidate %in% c("Clinton", "Sanders", "O'Malley") &
    `End Date` >= "2015-05-01") %>%
  ggplot(aes(x=`End Date`, y = percentage, color = candidate)) +
  geom_line()

```

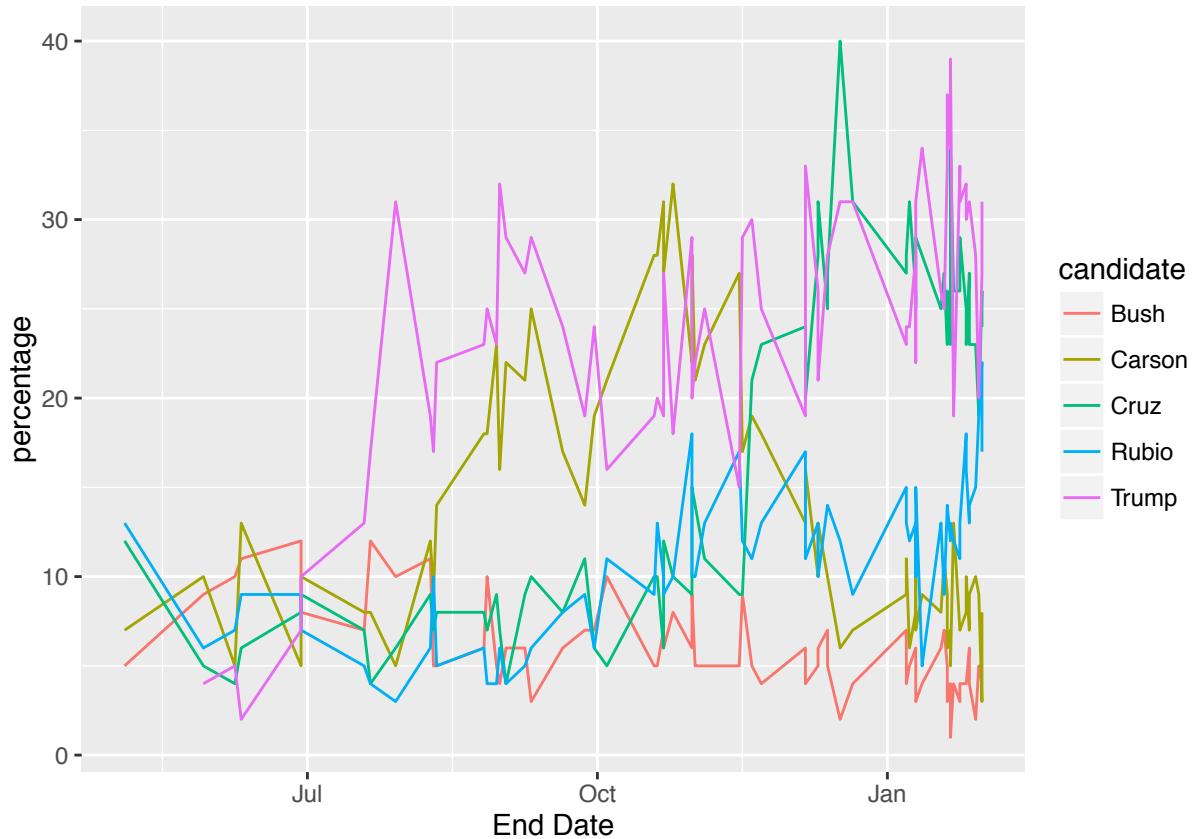


**Assessment (optional)** Repeat this analysis using the republican poll data. Filter for candidates (Trump, Cruz, Rubio, Carson, Bush) and for polls only after May 1, 2015. Color the lines by candidates.

```
## Provide your code here

rep_polls %>%
  gather(key = candidate, value = percentage, Trump:Walker) %>%
  filter(candidate %in% c("Trump", "Cruz", "Rubio", "Carson", "Bush") &
    `End Date` >= "2015-05-01") %>%
  ggplot(aes(x = `End Date`, y = percentage, color = candidate)) +
  geom_line()
```

## Warning: Removed 1 rows containing missing values (geom\_path).



## spread()

In contrast to *gathering* multiple columns into key-value pairs, we can *spread* a key-value pair across multiple columns.

The function `spread()` does just that. It transforms data from a *long* format into a *wide* format.

- The `key` is the name of the column in your data set that contains the values of the column headings that you are spreading across multiple columns
- The `value` is the name of the column that contains the values for the multiple columns

```
dem_polls_gathered <- dem_polls %>%
  gather(key = candidate, value = percentage,
         Clinton:Undecided)

## Source: local data frame [648 x 4]
## 
##           Pollster   End Date candidate percentage
##           (chr)     (date)    (chr)      (int)
## 1 Emerson College Polling Society 2016-01-31 Clinton      51
## 2 Quinnipiac 2016-01-31 Clinton      46
## 3 Des Moines Register/Bloomberg/Selzer 2016-01-29 Clinton      45
## 4 Gravis Marketing/One America News 2016-01-27 Clinton      53
```

```

## 5          PPP (D-Progress Iowa) 2016-01-27 Clinton      48
## 6          NBC/WSJ/Marist 2016-01-26 Clinton      48
## 7 Monmouth University 2016-01-26 Clinton      47
## 8          ARG 2016-01-24 Clinton      45
## 9          Quinnipiac 2016-01-24 Clinton      45
## 10         Iowa State/WHO-HD 2016-01-22 Clinton      47
## ...
## ...       ...     ...     ...     ...     ...

dem_polls_gathered %>%
  spread(key = candidate, value = percentage)

## Source: local data frame [81 x 10]
##
## #> #> Pollster End Date Biden Chafee Clinton Lessig O'Malley Sanders
## #> #> (chr)     (date) (int) (int) (int) (int) (int) (int)
## #> 1 ARG 2016-01-10 NA NA 44 NA 3 47
## #> 2 ARG 2016-01-24 NA NA 45 NA 3 48
## #> 3 CBS/YouGov 2015-09-10 10 1 33 NA 5 43
## #> 4 CBS/YouGov 2015-10-22 NA 1 46 0 3 43
## #> 5 CBS/YouGov 2015-11-19 NA NA 50 NA 5 44
## #> 6 CBS/YouGov 2015-12-17 NA NA 50 NA 4 45
## #> 7 CBS/YouGov 2016-01-21 NA NA 46 NA 5 47
## #> 8 CNN 2014-09-10 15 NA 53 NA 2 5
## #> 9 CNN 2015-08-11 12 0 50 NA 1 31
## #> 10 CNN 2015-11-04 NA NA 55 NA 3 37
## ...
## ...     ...     ...     ...     ...     ...     ...     ...     ...

## Variables not shown: Undecided (int), Webb (int)

```

## Other supporting functions in tidyverse

- `separate()` = separate one column into multiple columns
- `unite()` = unite multiple columns into one

```

dem_polls_separate <- dem_polls %>%
  separate(col = `End Date`, into = c("y", "m", "d"))
dem_polls_separate

```

```

## Source: local data frame [81 x 12]
##
## #> #> Pollster y m d Clinton Sanders
## #> #> (chr) (chr) (chr) (chr) (int) (int)
## #> 1 Emerson College Polling Society 2016 01 31 51 43
## #> 2 Quinnipiac 2016 01 31 46 49
## #> 3 Des Moines Register/Bloomberg/Selzer 2016 01 29 45 42
## #> 4 Gravis Marketing/One America News 2016 01 27 53 42
## #> 5 PPP (D-Progress Iowa) 2016 01 27 48 40
## #> 6 NBC/WSJ/Marist 2016 01 26 48 45
## #> 7 Monmouth University 2016 01 26 47 42
## #> 8 ARG 2016 01 24 45 48
## #> 9 Quinnipiac 2016 01 24 45 49
## #> 10 Iowa State/WHO-HD 2016 01 22 47 45
## ...
## ...     ...     ...     ...     ...     ...

```

```
## Variables not shown: O'Malley (int), Biden (int), Chafee (int), Lessig  
## (int), Webb (int), Undecided (int)
```

**Assessment** Use the `unite()` function to create a new column titled “`end_date`” that combines the columns `y`, `m` and `d` together into a single column separated by the “`/`” character.

```
## Provide your code here
```

```
dem_polls_separate %>%  
  unite(col = end_date, y, m, d, sep = "/")
```

```
## Source: local data frame [81 x 10]  
##  
##           Pollster   end_date Clinton Sanders  
##           (chr)      (chr)    (int)    (int)  
## 1 Emerson College Polling Society 2016/01/31      51      43  
## 2 Quinnipiac 2016/01/31      46      49  
## 3 Des Moines Register/Bloomberg/Selzer 2016/01/29      45      42  
## 4 Gravis Marketing/One America News 2016/01/27      53      42  
## 5 PPP (D-Progress Iowa) 2016/01/27      48      40  
## 6 NBC/WSJ/Marist 2016/01/26      48      45  
## 7 Monmouth University 2016/01/26      47      42  
## 8 ARG 2016/01/24      45      48  
## 9 Quinnipiac 2016/01/24      45      49  
## 10 Iowa State/WHO-HD 2016/01/22      47      45  
## .. ... ... ... ...  
## Variables not shown: O'Malley (int), Biden (int), Chafee (int), Lessig  
## (int), Webb (int), Undecided (int)
```

## Cheatsheets

- Data Wrangling with `dplyr` and `tidyverse` from RStudio

# Distributions and basic exploratory data analysis

Assessment: Read in the data [here](#). Use the `filter` function to determine how many males and females we have.

Assessment: Use filter and select to obtain a `data.frame` with just male heights. How many NAs do we have when we convert them to numeric?

## Exploratory Data Analysis

An indispensable part of data wrangling is exploratory data analysis. In particular with large dataset, it is practically impossible to examine the data from potential errors by looking at tables. Here we describe some simple visualization techniques that are quite powerful for summarizing data and identifying potential errors.

## Distributions

The most basic statistical summary of a list of numbers is its distribution. The simplest way to think of a *distribution* is as a compact description of many numbers. For example, we have measured the heights of all students in a course. Imagine you need to describe these numbers to someone that has no idea what these heights are, such as an alien that has never visited Earth.

One approach to summarizing these numbers is to simply list them all out for the alien to see. Here are 10 randomly selected heights :

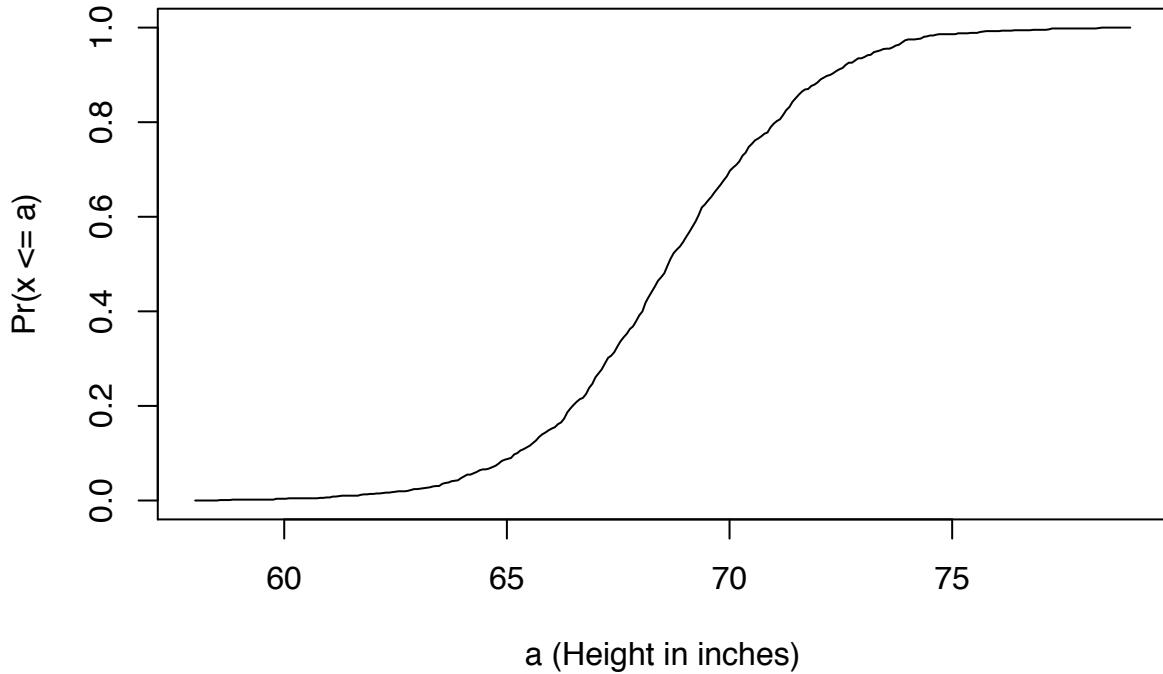
```
select(dat, height) %>% print(n=10)
```

```
## Source: local data frame [148 x 1]
##
##      height
##      (dbl)
## 1   63.0000
## 2   62.0000
## 3   69.0000
## 4   68.0000
## 5   71.6500
## 6   75.0000
## 7   68.8976
## 8   74.0000
## 9   65.0000
## 10  64.0000
## ...  ...
```

**Cumulative Distribution Function** Scanning through these numbers, we start to get a rough idea of what the entire list looks like, but it is certainly inefficient. We can quickly improve on this approach by defining and visualizing a *distribution*. To define a distribution we compute, for all possible values of  $a$ , the proportion of numbers in our list that are below  $a$ . We use the following notation:

$$F(a) \equiv \Pr(x \leq a)$$

This is called the cumulative distribution function (CDF). When the CDF is derived from data, as opposed to theoretically, we also call it the empirical CDF (ECDF). The ECDF for the adult male height data looks like this:



## Histograms

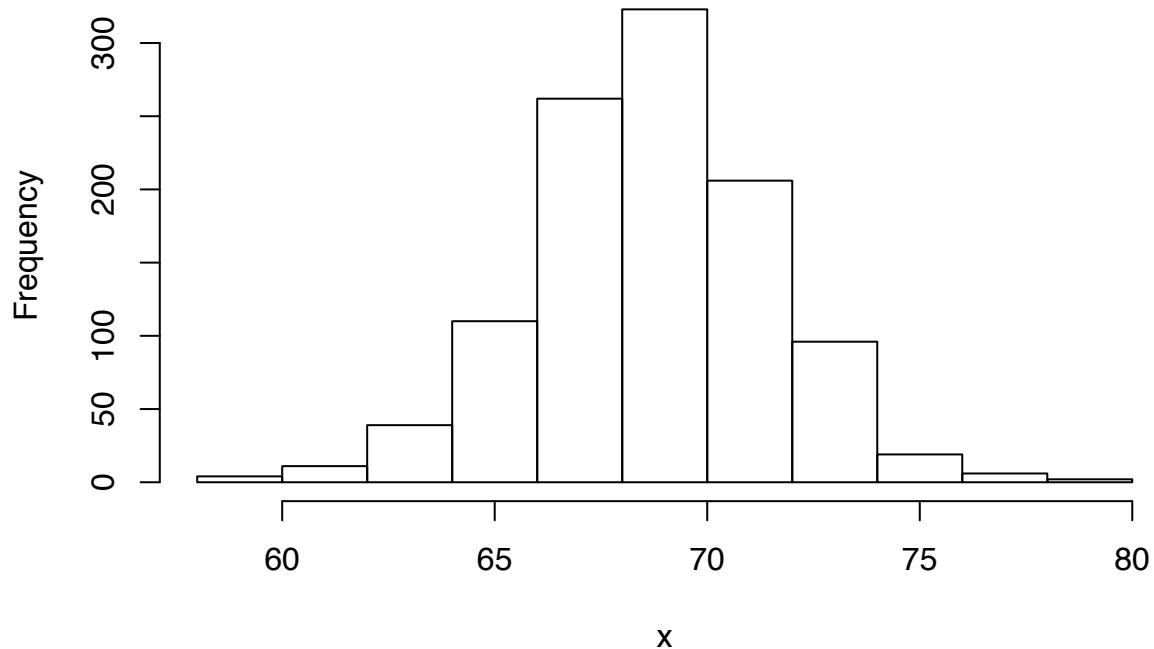
Although the empirical CDF concept is widely discussed in statistics textbooks, the plot is actually not very popular in practice. The reason is that histograms give us the same information and are easier to interpret. Histograms show us the proportion of values in intervals:

$$\Pr(a \leq x \leq b) = F(b) - F(a)$$

Plotting these heights as bars is what we call a *histogram*. It is a more useful plot because we are usually more interested in intervals, such and such percent are between 70 inches and 71 inches, etc., rather than the percent less than a particular height. It is also easier to distinguish different types (families) of distributions by looking at histograms. Here is a histogram for the general population:

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

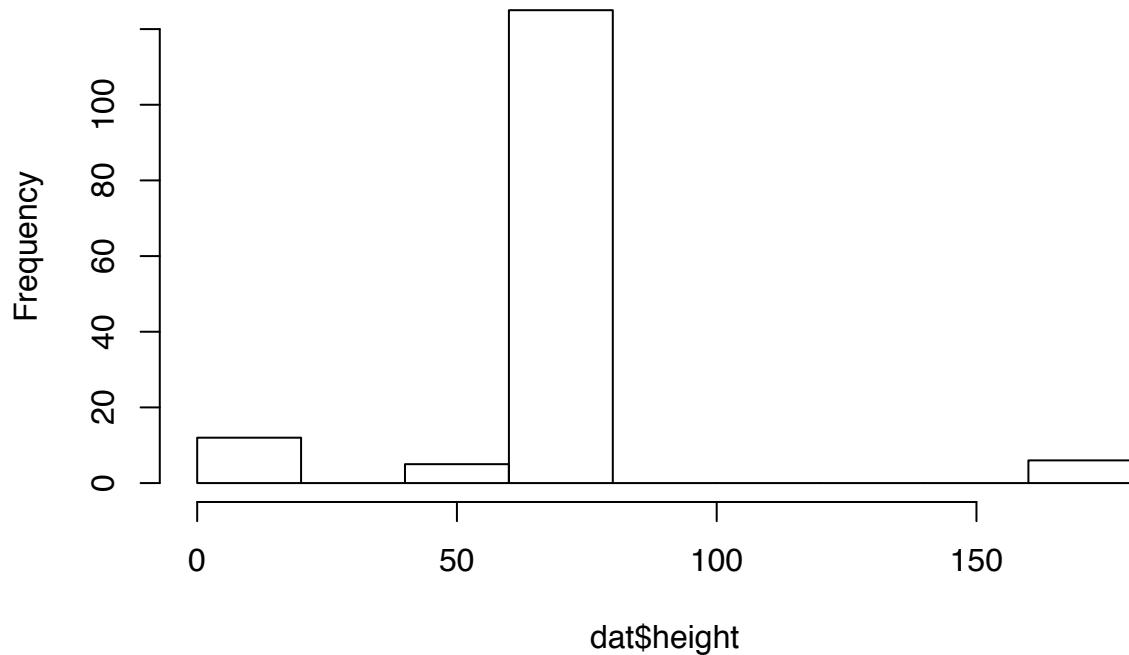
### Histogram of x



Here is a histogram for our class:

```
hist(dat$height)
```

### Histogram of dat\$height



## Outliers

The plot has revealed another problem. We have a number hieghts that are larger than 96 inches and shorter than 12 inches. Let's view the data for which this is the case. To do this we introduce the or logical operator |

```
filter(dat, height>96 | height < 12)

## Source: local data frame [18 x 3]
##
##           time gender height
##   (chr)   (chr)   (dbl)
## 1 1/25/2016  Male    5.80
## 2 1/25/2016  Male    5.10
## 3 1/25/2016  Male    5.11
## 4 1/25/2016 Female   5.70
## 5 1/26/2016  Male  172.00
## 6 1/26/2016  Male  180.00
## 7 1/26/2016 Female   5.00
## 8 1/26/2016  Male  180.00
## 9 1/26/2016  Male  180.00
## 10 1/26/2016  Male    5.90
## 11 1/26/2016 Female   5.20
## 12 1/26/2016  Male    5.50
## 13 1/26/2016 Female   5.51
## 14 1/26/2016  Male    5.80
## 15 1/26/2016 Female   5.70
## 16 1/26/2016  Male  167.00
## 17 1/27/2016  Male  168.00
## 18 1/27/2016  Male    6.00
```

```
# %>% select(original)
```

We see several heights that appear to be in centimeters. We will go ahead and assume this is the case and make the conversion:

```
dat <- mutate(dat, height = ifelse(height > 96, height / 2.54, height))
```

Now let's see what outliers remain:

```
filter(dat, height > 96 | height < 12) %>% select(height)
```

```
## Source: local data frame [12 x 1]
##
##       height
##   (dbl)
## 1    5.80
## 2    5.10
## 3    5.11
## 4    5.70
## 5    5.00
## 6    5.90
```

```

## 7    5.20
## 8    5.50
## 9    5.51
## 10   5.80
## 11   5.70
## 12   6.00

```

These values appear to use the format  $x.y$  with  $x$  feet and  $y$  inches. Note that these are not numbers. In particular note that 5.11 is larger than 5.5. We also note a particularly strange entry which we will just assume is 5.5:

```
dat <- mutate(dat, height=ifelse(height==5.51, 65, height))
```

For the rest we convert to inches using the functions `floor`. This approach does not work for 5.11 so we treat that differently:

```
dat <- mutate(dat, height=ifelse(height==5.11, 71, height))
dat <- mutate(dat, height=ifelse(height>12, height, floor(height)*12+(height-floor(height))*10))
```

Let's confirm that we have removed all the outliers:

```
filter(dat, height>96 | height < 12) %>% select(height)
```

```

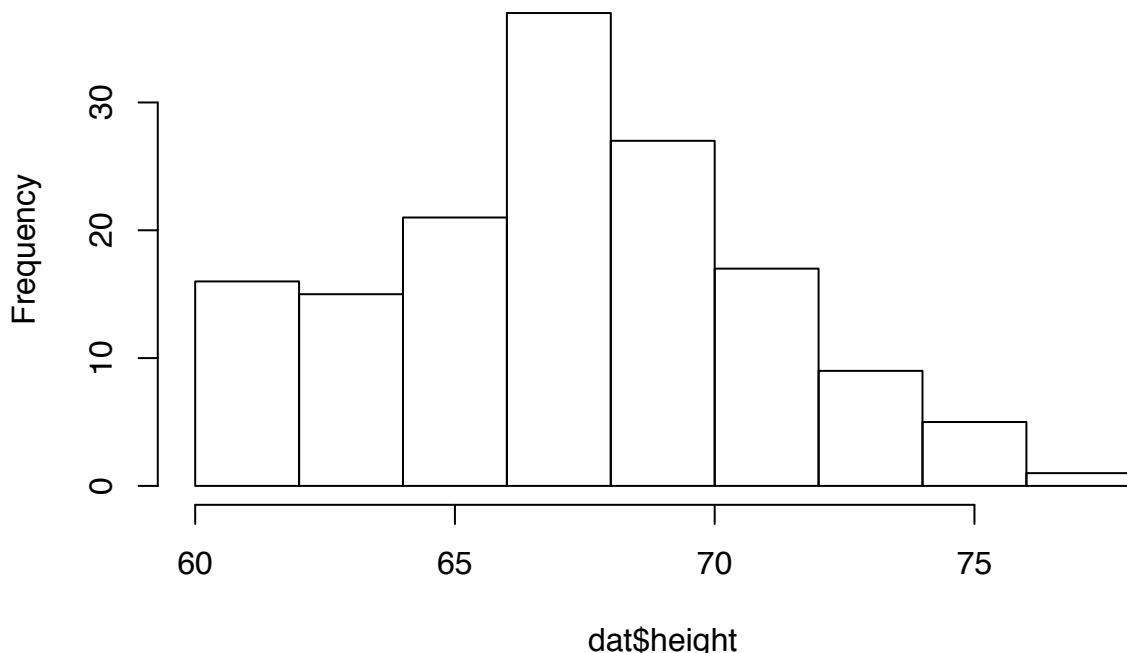
## Source: local data frame [0 x 1]
##
## Variables not shown: height (dbl)

```

The histogram looks more like the general population now

```
hist(dat$height)
```

### Histogram of dat\$height



Although not quite the same.

Assessment. Wrangle the data as we did in class. You can obtain the function to fix heights from [here](#)

What is the height of the 113th entry ?

## Normal Distribution

The histogram provides an excellent summary plot of a distribution. Can we summarize even further? We often see the average and standard deviation used as summary statistics. To understand why these are so widely used we need to understand the normal distribution.

The bell curve, also known as the normal distribution or Gaussian distribution is commonly found in nature. There are reasons for this which we will explain later.

When the histogram of a list of numbers is said to be approximated by the normal distribution, it means we can use a convenient mathematical formula to approximate the proportion of values or outcomes in any given interval:

$$\Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) dx$$

While the formula may look intimidating, don't worry, you will never actually have to type it out, as it is stored in a more convenient form (as `pnorm` in R which sets  $a$  to  $-\infty$ , and takes  $b$  as an argument).

Note that if this distribution approximates our data, then we only need to know the average  $\mu$  and the standard deviation  $\sigma$  to describe the entire population.

If we denote the values in our list as  $x_1, \dots, x_n$ . The mean:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

The variance:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_X)^2$$

with the standard deviation being the square root of the variance.

If we have a vector in R we can use the following functions:

```
x <- father.son$sheight
mu <- mean(x)
sigma <- sqrt( mean( (x - mu)^2 ) )
```

If in fact, the normal distribution is a good approximation we should be able to obtain approximations of the proportion in any range. For example, how many men are taller than six feet?

We can obtain the exact answer like this:

```
sum( x > 6*12 ) / length(x)
```

```
## [1] 0.1141002
```

```
##which is equivalent to  
mean( x>6*12)
```

```
## [1] 0.1141002
```

How good is the normal approximation?

```
1 - pnorm( 6*12, mu, sigma )
```

```
## [1] 0.1192743
```

We can try other values and see that we get very good approximations. Once we know the US adult men have an average height of 69 inches and standard deviation of 3 inches, we know everything: two numbers are all we need to describe the distribution.

Assessment: If a list of numbers has a distribution that is well approximated by the normal distribution, what proportion of these numbers are within one standard deviation away from the list's average?

### Standard units

Once we know that a list of numbers follow the normal distribution, a convenient way to describe a value is the number of standard deviations away from the average. We say these values are in *standard units*:

```
z <- (x - mu)/sigma
```

So, for example, a male seven footer is 5 SDs away from the average.

If the original distribution is approximately normal, then these values will have a *standard normal* distribution: average 0 and standard deviation 1. Notice that about 95% of the values are within two standard deviation of the average:

```
pnorm(2)-pnorm(-2)
```

```
## [1] 0.9544997
```

and most values are within 3

```
pnorm(3)-pnorm(-3)
```

```
## [1] 0.9973002
```

Assessment: Use the normal approximation to determine what proportion of US men are as tall or taller as Michael Jordan?

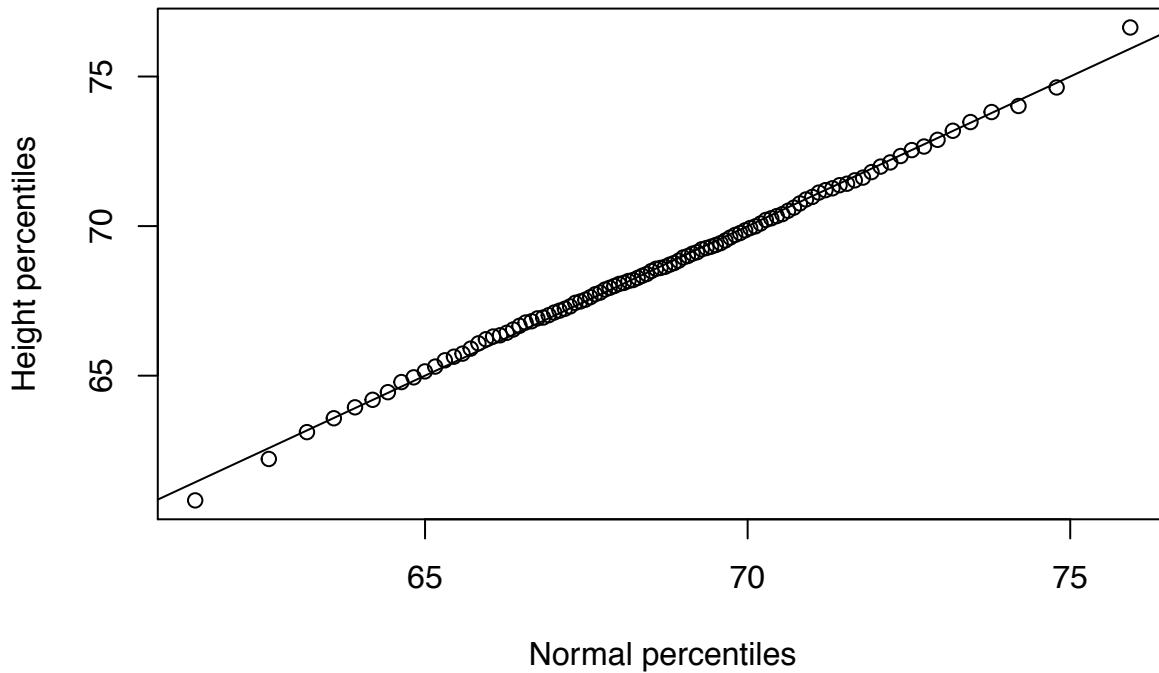
Assessment: In the US, how many more 6'6 or taller men are there than 7 foot or taller? Compute a proportion.

Assessment: What proportion of 7 footers (in the world) are in the NBA? Hint: There are 36 7 footers in the NBA. Hint 2: The average height of men 18-35 is 68.5 and SD 3 inches and is approximately normal?

## Quantile Quantile Plots

To corroborate that a theoretical distribution, for example the normal distribution, is in fact a good approximation, we can use quantile-quantile plots (qq-plots). Quantiles are best understood by considering the special case of percentiles. The p-th percentile of a list of a distribution is defined as the number q that is bigger than p% of numbers (so the inverse of the cumulative distribution function we defined earlier). For example, the median 50-th percentile is the median. We can compute the percentiles for male heights and for the normal distribution:

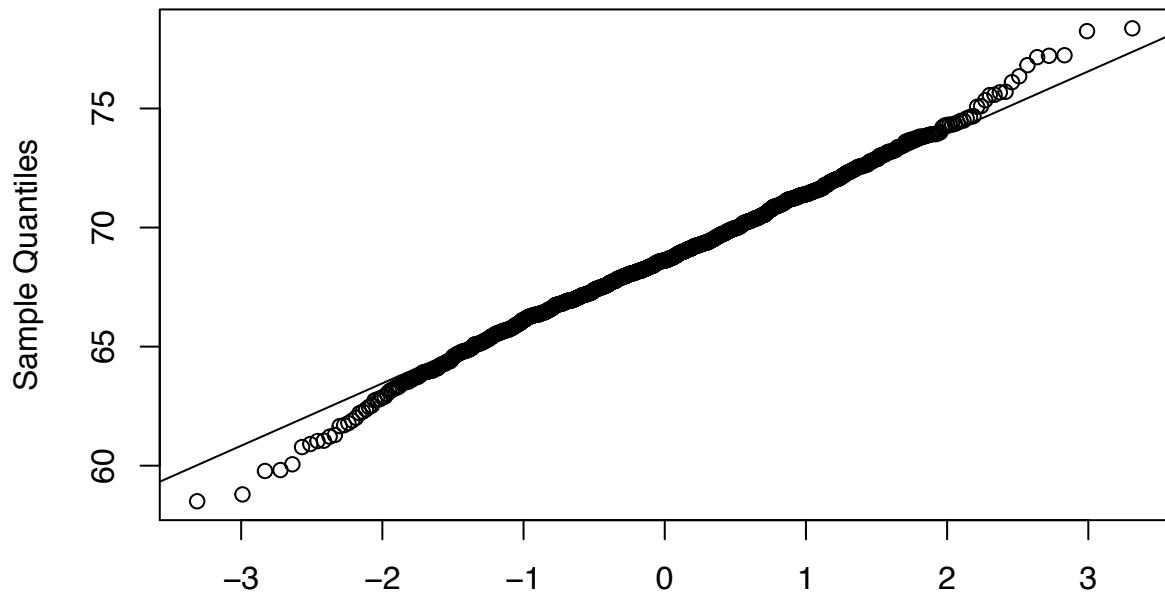
```
ps <- ( seq(0,99) + 0.5 )/100
qs <- quantile(x, ps)
normalqs <- qnorm(ps, mu, sigma)
plot(normalqs,qs,xlab="Normal percentiles",ylab="Height percentiles")
abline(0,1) ##identity line
```



Note how close these values are. Also, note that we can see these qq-plots with less code (this plot has more points than the one we constructed manually, and so tail-behavior can be seen more clearly).

```
qqnorm(x)
qqline(x)
```

## Normal Q–Q Plot

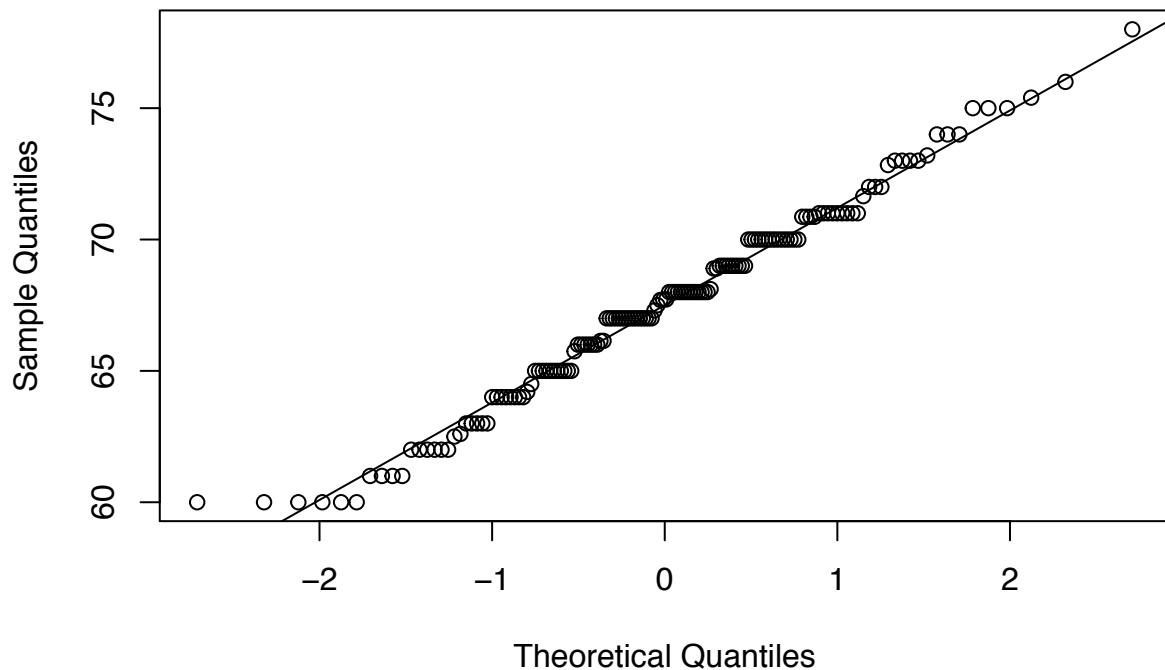


## Theoretical Quantiles

However, the `qqnorm` function plots against a standard normal distribution. This is why the line has slope  $\sigma$  and intercept  $\mu$ .

```
qqnorm(dat$height)
qqline(dat$height)
```

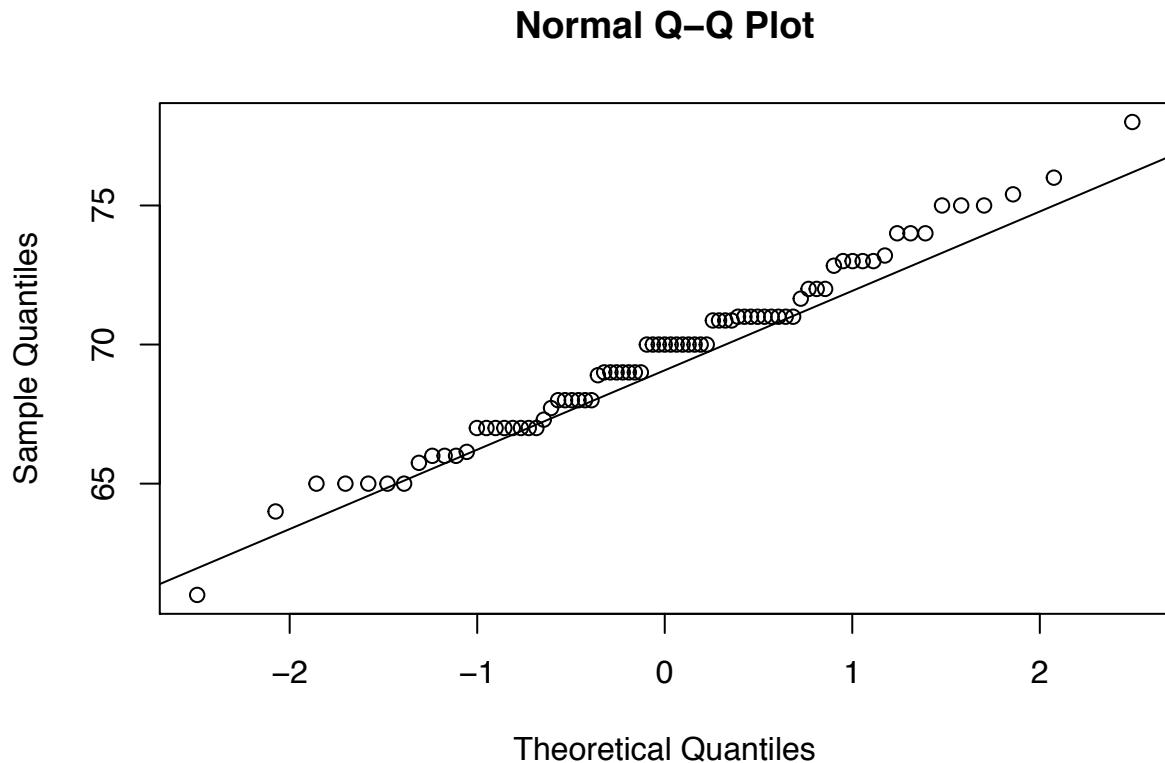
## Normal Q–Q Plot



## Theoretical Quantiles

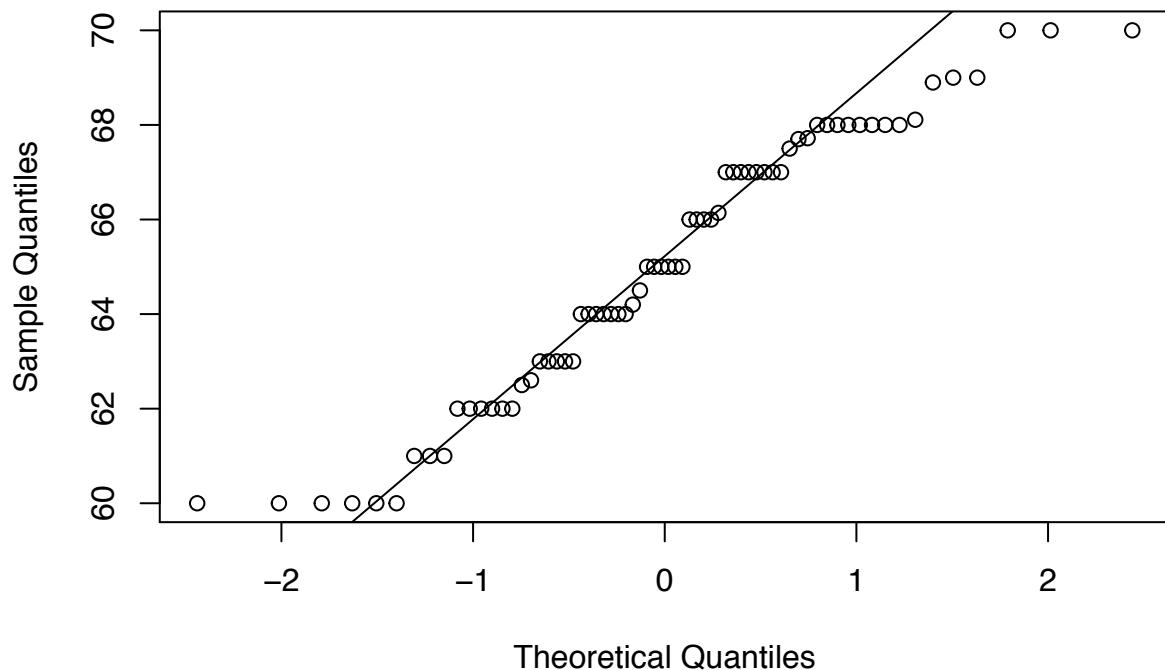
We can split it into the two genders and see slightly better fits (except for the tails)

```
men <- filter(dat, gender=="Male")
qqnorm(men$height)
qqline(men$height)
```



```
women <- filter(dat, gender=="Female")
qqnorm(women$height)
qqline(women$height)
```

## Normal Q-Q Plot



## Theoretical Quantiles

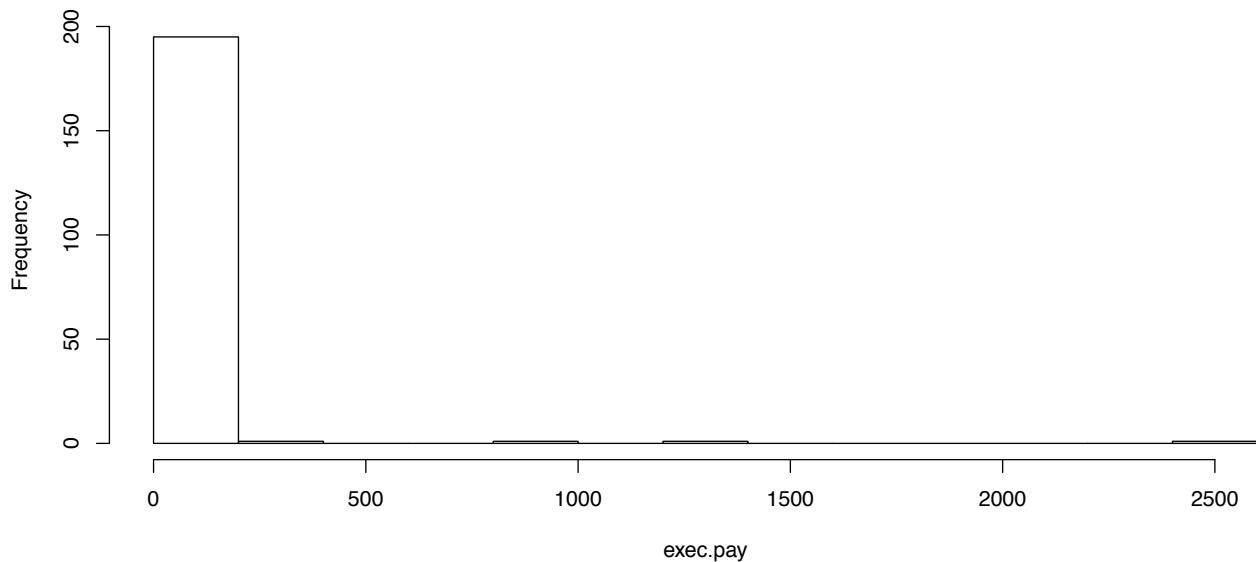
Assessment In our empirical distribution of heights, how many men are taller than Michael Jordan? Why the discrepancy with our answer above?

## Boxplots

Data is not always normally distributed. Income is a widely cited example. In these cases, the average and standard deviation are not necessarily informative since one can't infer the distribution from just these two numbers. The properties described above are specific to the normal. For example, the normal distribution does not seem to be a good approximation for the direct compensation for 199 United States CEOs in the year 2000.

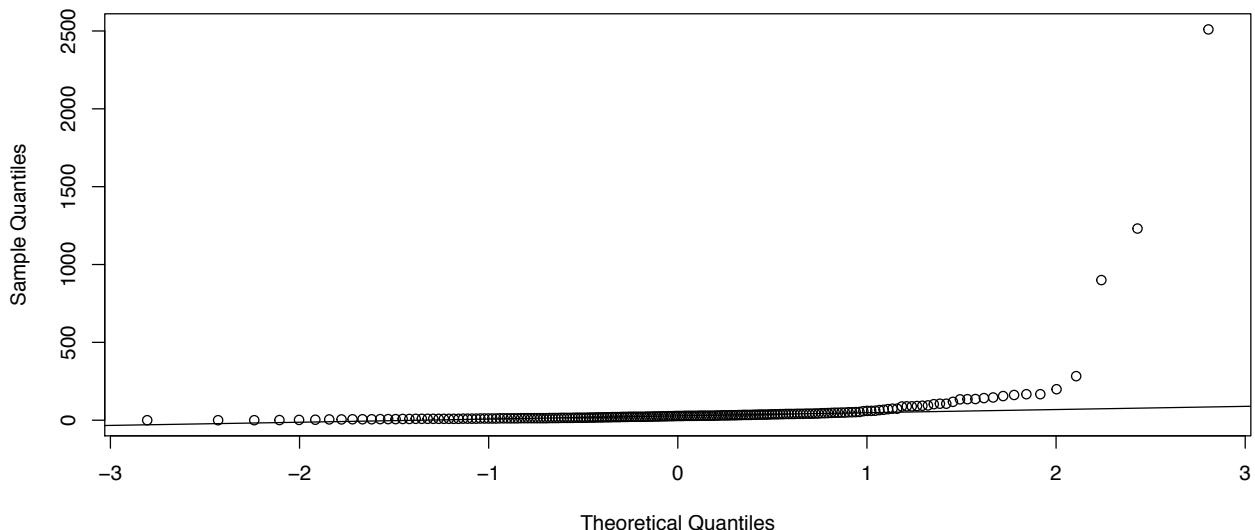
```
data(exec.pay, package="UsingR")
hist(exec.pay)
```

**Histogram of exec.pay**



```
qqnorm(exec.pay)  
qqline(exec.pay)
```

**Normal Q–Q Plot**



In addition to qq-plots, a practical summary of data is to compute 3 percentiles: 25-th, 50-th (the median) and the 75-th. A boxplot shows these 3 values along with a range of the points within  $\text{median} \pm 1.5$  (75-th percentile - 25th-percentile). Values outside this range are shown as points and sometimes referred to as *outliers*.

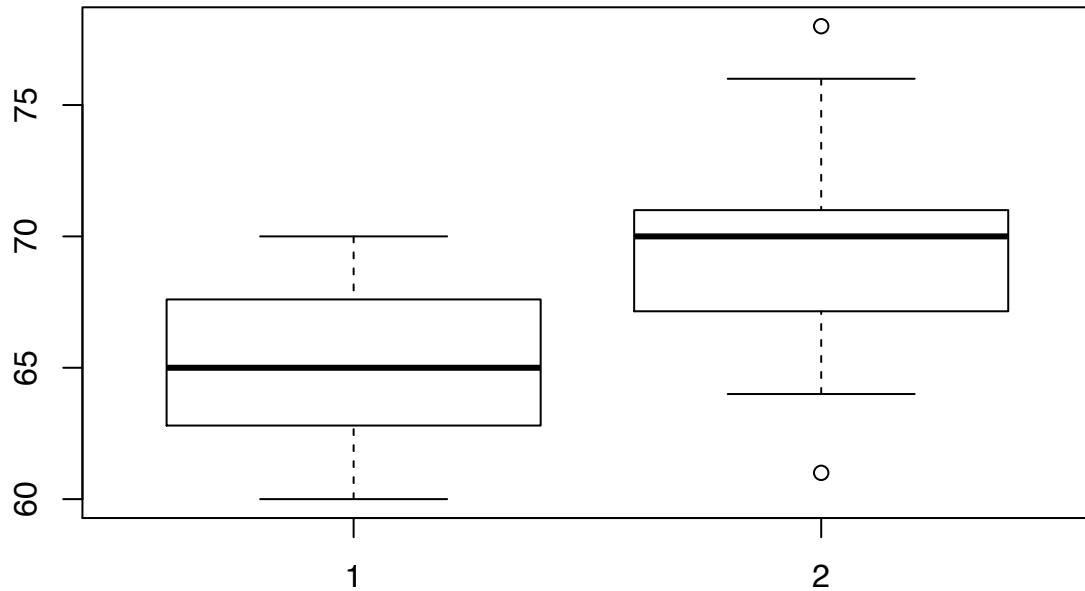
```
boxplot(exec.pay)
```



Here we show just one boxplot. However, one of the great benefits of boxplots is that we could easily show many distributions in one plot, by lining them up, side by side.

Here are the distribution of men and women heights from our class

```
boxplot( women$height , men$height )
```



# Introduction to Data Visualization

*Stephanie Hicks, Rafael Irizarry*

## Data Visualization

### Introduction

“The greatest value of a picture is when it forces us to notice what we never expected to see.” -John W. Tukey



Mistakes, biases, systematic errors and unexpected variability are commonly found in data regardless of applications. Failure to discover these problems often leads to flawed analyses and false discoveries. As an example, consider that measurement devices sometimes fail and not all summarization procedures, such as the `mean` function in R, are designed to detect these. Yet, these functions will still give you an answer. Furthermore, it may be hard or impossible to notice an error was made just from the reported summaries.

Data visualization is a powerful approach to detecting these problems. We refer to this particular task as *exploratory data analysis* (EDA). Many important methodological contributions to existing techniques in data analysis were initiated by discoveries made via EDA.

On a more positive note, data visualization can also lead to discoveries which would otherwise be missed if we simply subject the data to a battery of statistical summaries or procedures. Through this course, we make use of exploratory plots to motivate the analyses we choose.

Finally, data visualization can provide a powerful way to communicate a data-driven finding. In some cases, the visualization is so convincing, that no follow-up is required. Many examples of this have appeared in

the New York Times. The following three examples were provided by Amanda Cox, a data scientist that is responsible for most of the [infographics](#) we see in The New York Times. Both examples are basically histograms.

**NYC Regents Exam** In New York City you need a 65 to pass the Regents exam. Data on these scores are collected for several reasons. A histogram of the test scores of grades from NYC Regents Exam tells an interesting story. The histogram of these test scores *forces us to notice* something somewhat problematic:

**Voting patterns** The following histogram helps us understand voting patterns. The original [infographic](#) is actually interactive. The following is a screenshot of one of the many histograms one can view:

**Tax rates by company** [Here](#) is another example of an advanced visualization based on the histogram idea.

We have already introduced some EDA approaches for *univariate* data, namely the histograms and qq-plots. Here we provide more details and introduce some tools and summary statistics for paired data. We do this using the `ggplot2` package and much of the focus of this part is introducing this powerful tool.

## Motivating Example: Global Health and Economic Data

Throughout this section we will be analyzing global health and economic data. We will be exploring two questions:

1. Is there a relationship between life expectancy and GDP per capita?
2. How has the GDP per capita distribution changed across time?

We will use the data that has been organized and delivered in the `gapminder` package available in an [R package](#) from [Jenny Bryan](#).

To install the `gapminder` data set

```
library(devtools)
install_github("jennybc/gapminder")
```

## Loading data into R

We start by loading the data. We will be using `dplyr` to manipulate the data, so we load that as well.

```
library(dplyr)
library(gapminder)
```

We can take a quick peak at the loaded `gapminder` object:

```
gapminder          ## returns the first 10 rows because it is a `tbl_df`  
  
## # Source: local data frame [1,704 x 6]  
## #  
##   country continent year lifeExp      pop gdpPercap  
##   (fctr)     (fctr)  (int)    (dbl)    (int)      (dbl)
```

```

## 1 Afghanistan      Asia 1952 28.801 8425333 779.4453
## 2 Afghanistan      Asia 1957 30.332 9240934 820.8530
## 3 Afghanistan      Asia 1962 31.997 10267083 853.1007
## 4 Afghanistan      Asia 1967 34.020 11537966 836.1971
## 5 Afghanistan      Asia 1972 36.088 13079460 739.9811
## 6 Afghanistan      Asia 1977 38.438 14880372 786.1134
## 7 Afghanistan      Asia 1982 39.854 12881816 978.0114
## 8 Afghanistan      Asia 1987 40.822 13867957 852.3959
## 9 Afghanistan      Asia 1992 41.674 16317921 649.3414
## 10 Afghanistan     Asia 1997 41.763 22227415 635.3414
## ...

```

```
str(gapminder)
```

```

## Classes 'tbl_df', 'tbl' and 'data.frame': 1704 obs. of 6 variables:
##   $ country : Factor w/ 142 levels "Afghanistan",...: 1 1 1 1 1 1 1 1 1 ...
##   $ continent: Factor w/ 5 levels "Africa","Americas",...: 3 3 3 3 3 3 3 3 3 ...
##   $ year    : int 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##   $ lifeExp : num 28.8 30.3 32 34 36.1 ...
##   $ pop     : int 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 22
##   $ gdpPercap: num 779 821 853 836 740 ...

```

This package provides a subset of the data provided by [Gapminder](#). The data comes as a data frame (or `tbl_df` if you're using `dplyr`). We can visit [Gapminder.org documentation page](#) to obtain information about the columns:

variable	meaning
country	
continent	
year	
lifeExp	life expectancy at birth
pop	total population
gdpPercap	per-capita GDP

## ggplot2

`ggplot2` is a powerful data exploration and visualization package that can create graphics in R. It was created by [Hadley Wickham](#) when he was a graduate student at Iowa State University and is based on the principals defined in the [Grammar of Graphics](#).

**Why is it useful?** The idea of the Grammar of Graphics is to break the graph into components and handle the components of a graph separately. The `ggplot2` package contains a set of functions that allow us to build the features of the graph in a series of layers for versatility and control.

There are two main plotting functions in `ggplot2`:

- `qplot()` = a quick plot, similar to the `plot()` function in base R
- `ggplot()` = a “grammar of graphics” (gg) plot which is made up two components (a `ggplot()` object and at least one `geom` layer)

First we will install and load `ggplot2`:

```
#install.packages("ggplot2", dependencies = TRUE)
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.4
```

then we will explore the differences between `qplot()` and `ggplot()`.

## Life expectancy and GDP per capita

To try and answer our first question we will make a *scatterplot*. We will do this for the latest entry in our database which is:

```
max( gapminder$year )
```

```
## [1] 2007
```

We will first make a quick plot of these two variables.

**qplot** The `qplot` function allows us to produce simple plots, similar to `plot` in base R. The main input of `qplot` is an `x` and `y` argument, but it also accepts a data set (`data`) where the `x` and `y` variables are located.

To create a scatter-plot of life expectancy versus GDP per capita for 2007, we start by creating a subset of the data with only the 2007 year.

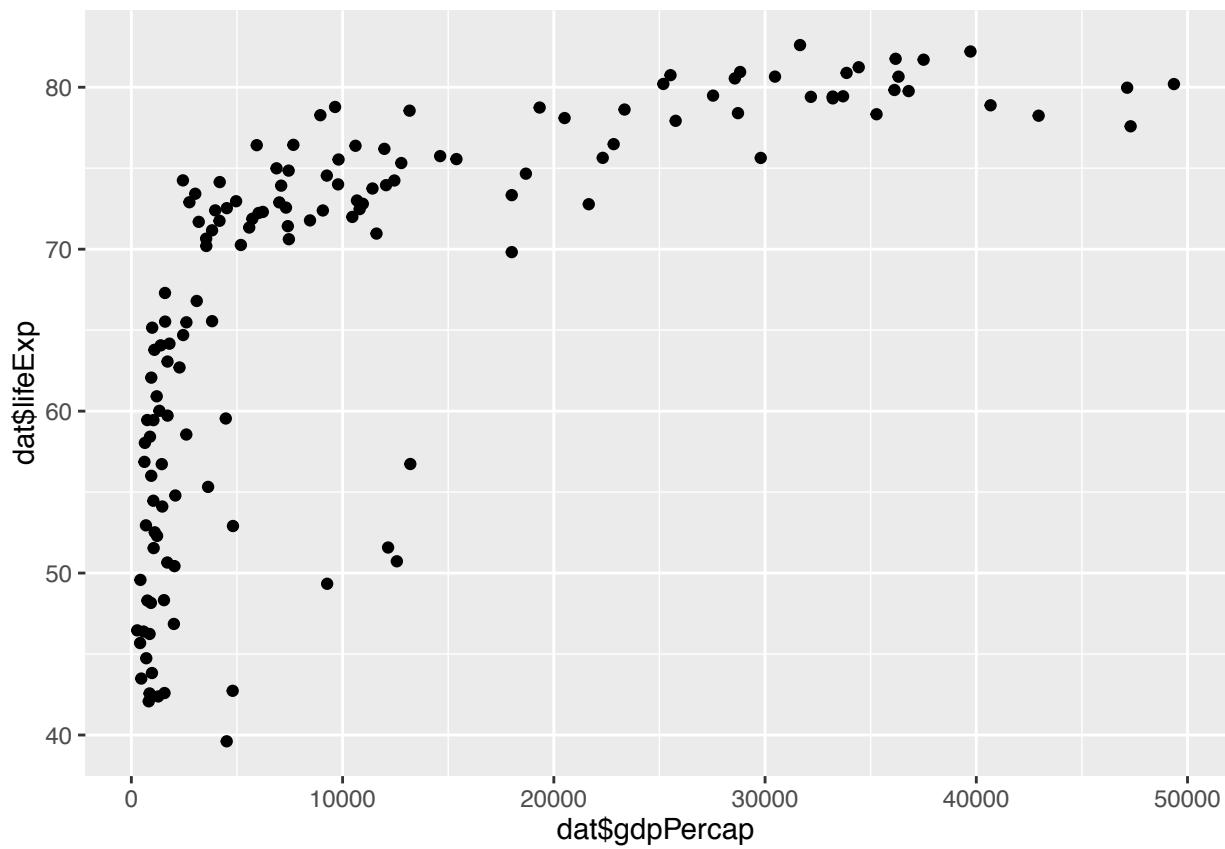
```
dat <- filter(gapminder, year==2007)
```

Why does this not work?

```
qplot(x = gdpPercap, y = lifeExp)
```

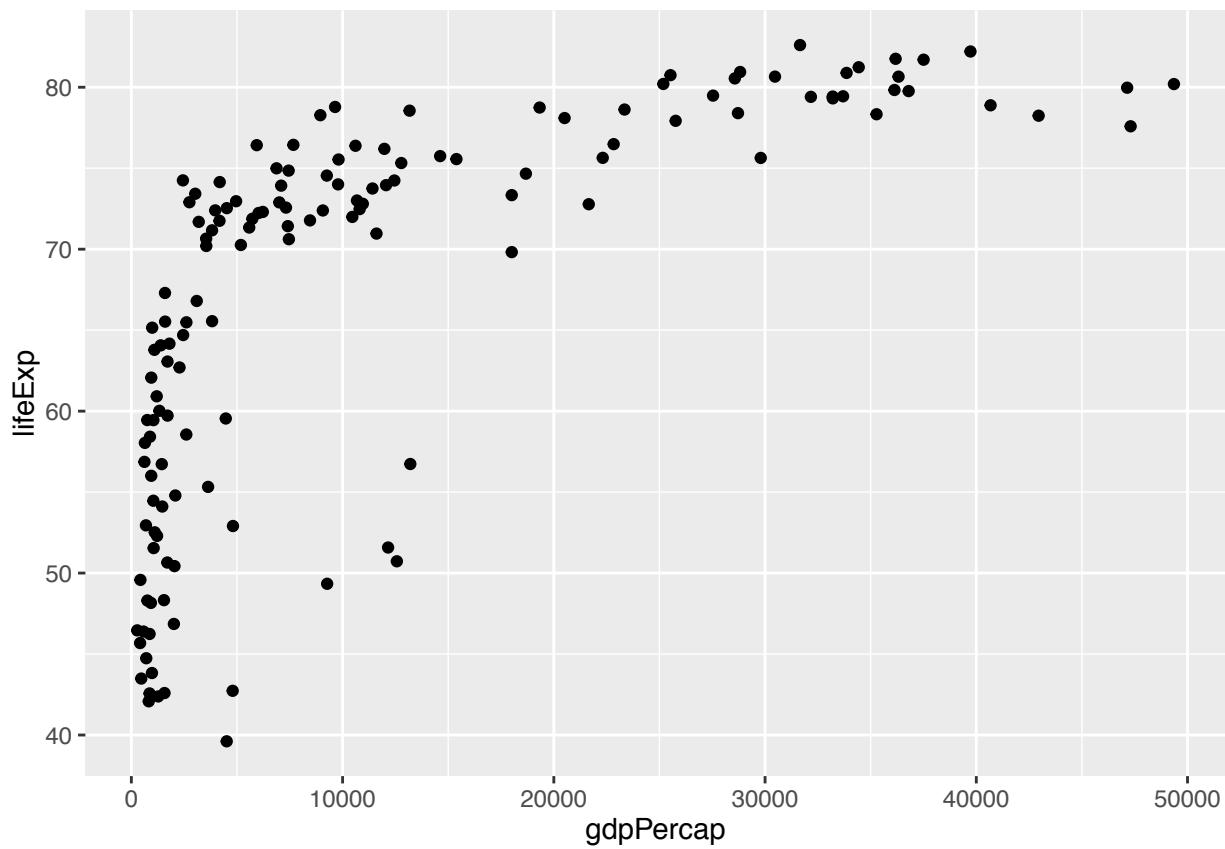
We need to specify where to get these values from. Instead, we can use the vectors of `dat` directly

```
## need to specify where variables are located
qplot(x = dat$gdpPercap, y = dat$lifeExp)
```



However, a cleaner way to do this does not define `dat` and instead uses the `data` argument:

```
## instead of having to specify name of data frame each time, just list it once
qplot(x = gdpPercap, y = lifeExp, data = filter(gapminder, year==2007))
```



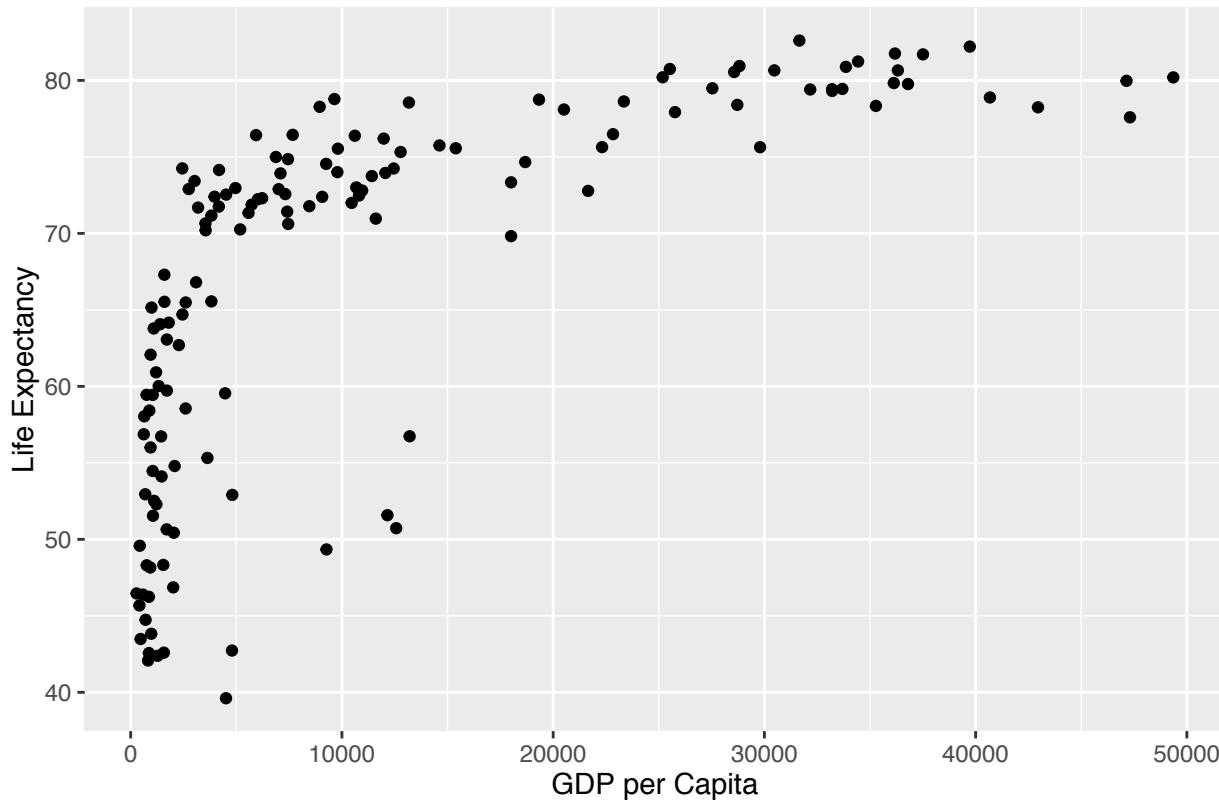
Assessment: Read the help file for `qplot`. What is the argument used to add a label to the x-axis?

Assessment: What is the argument used to add a label to the y-axis?

Assessment: What is the argument used to add a label to the title?

```
qplot(x = gdpPercap, y = lifeExp,
      data = filter(gapminder, year==2007),
      xlab = "GDP per Capita",
      ylab = "Life Expectancy",
      main = "Data from Gapminder")
```

## Data from Gapminder



From this plot, we see that there is a wide variability in life expectancy for the lower income countries and then somewhat of a positive trend. However, there are many countries with incomes below 5,000 dollars per person and it is hard to see differences between these. Next we will now learn about the grammar of graphics and create a series of visualizations.

## ggplot

This function is the implementation of the “Grammar of Graphics” that allows us to build layers of graphical elements to produce plots. As explained by Hadley Wickham

the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinates system.

## Terminology

- **ggplot** - the main function where you specify the data set and variables to plot (this is where we define the **x** and **y** variable names)
- **geoms** - geometric objects
  - e.g. `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **aes** - aesthetics
  - shape, transparency, color, fill, linetype
- **scales** - define how your data will be plotted

- continuous, discrete, log, etc

There are three ways to initialize a `ggplot()` object.

An empty ggplot object

```
p <- ggplot()
```

A ggplot object associated with a dataset

```
p <- ggplot(filter(gapminder, year==2007))
```

or a ggplot object with a dataset and x and y defined

```
p <- ggplot(filter(gapminder, year==2007), aes(x=gdpPercap, y = lifeExp))
```

```
p
```

**Creating your first ggplot()** We just used the function `aes()` which is an aesthetic mapping function inside the `ggplot()` object. We use this function to specify plot attributes (e.g. x and y variable names) that will not change as we add more layers.

Anything that goes in the `ggplot()` object becomes a global setting. From there, we use the `geom` objects to add more layers to the base `ggplot()` object. These will define what we are interested in illustrating using the data.

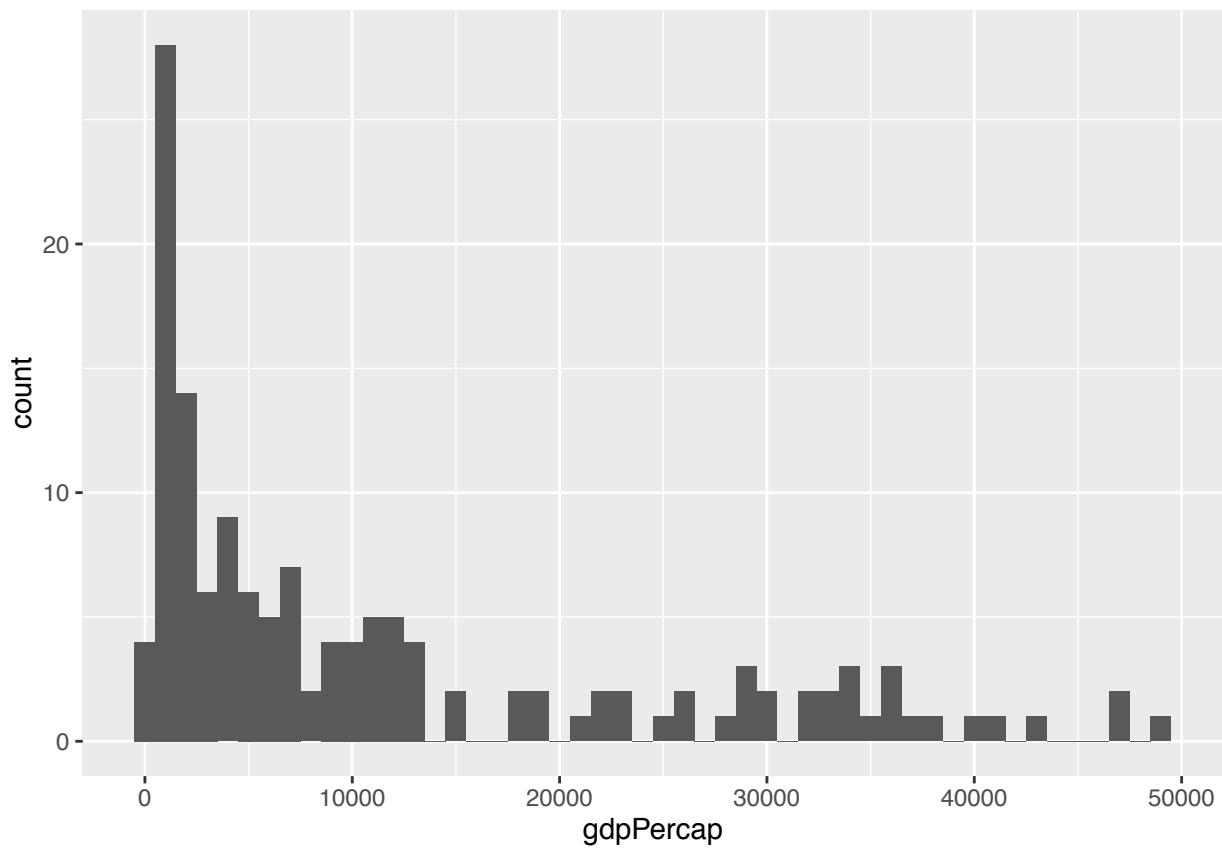
Earlier we noticed that there are many countries with incomes below 5,000 dollars per person and it is hard to see differences between these. We can examine just this variable with a histogram. Here we would use a different geometry:

```
p <- ggplot(filter(gapminder, year==2007),
            aes(gdpPercap))
p2 <- p + geom_histogram()
```

Here we get a message telling us that the number of bins was chose to be 30. How can we change it?

Assessment: Read the help file for `geom_histogram` and make a histogram with smaller bins:

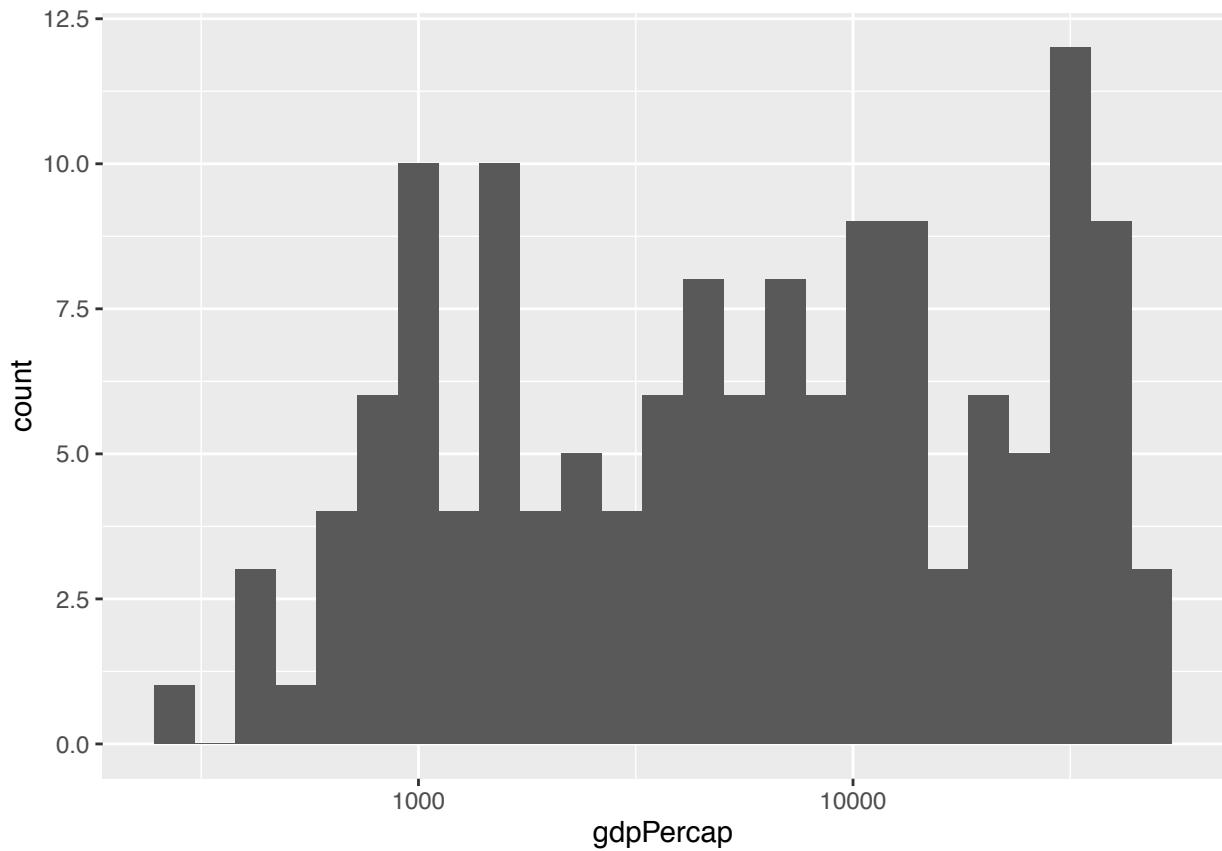
```
p + geom_histogram(binwidth=1000)
```



The histogram shows very large tails. We may do better by transforming the data. For data like this, the log transformation seems to work well. It also has a nice economic interpretation related to percent growth: in  $\log_{10}$  a change of 1 means the country is 10 times richer.

So how do we make the x-axis in the log scale? It is convenient to have [this](#) cheat sheet around when using ggplot2. From there we see that `scale_x_log10` does what we want.

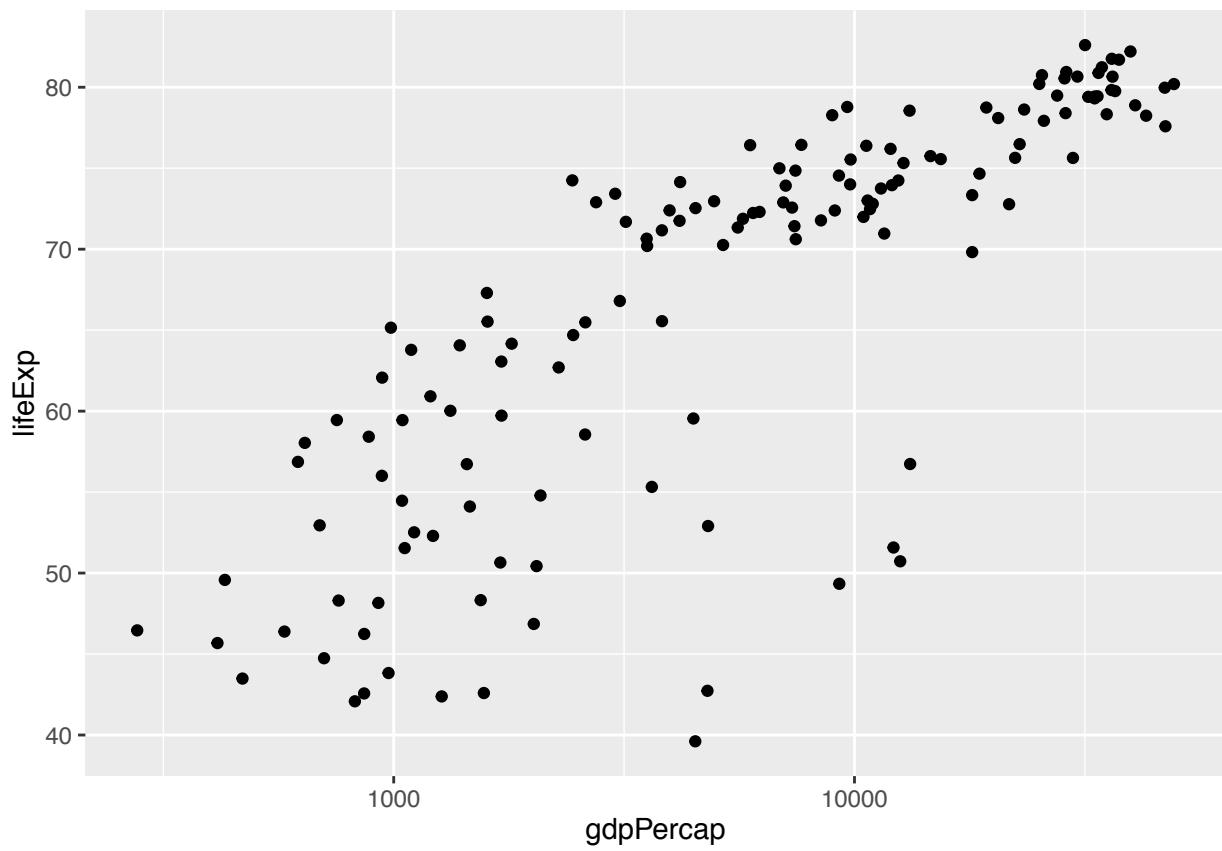
```
p <- ggplot(filter(gapminder,year==2007), aes(gdpPercap)) + geom_histogram(bins=25) + scale_x_log10()
```



We no longer see extreme tails. The scatter plot now looks much more informative:

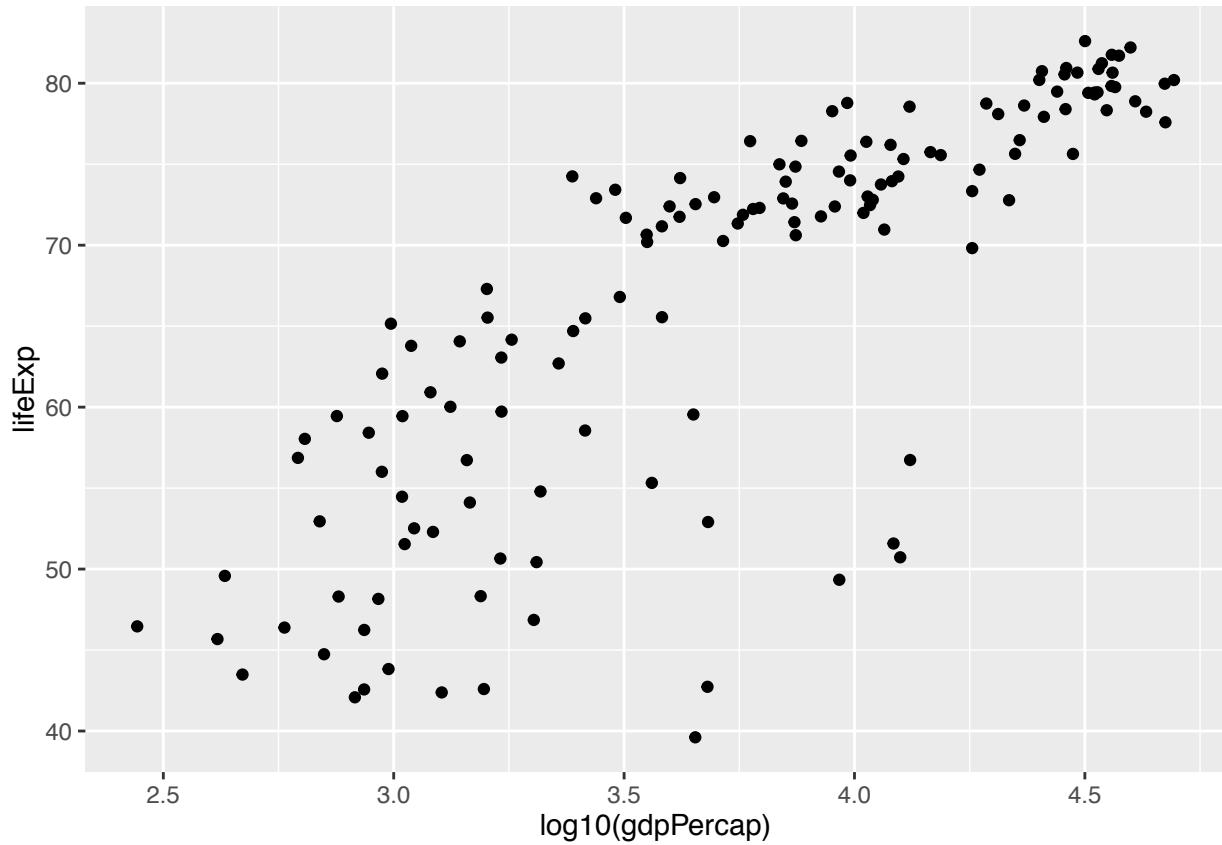
Assessment: Remake the scatter plot but now make sure the x-axis is in a log-scale

```
p <- ggplot(filter(gapminder, year==2007),  
            aes(gdpPercap, y=lifeExp))  
p + geom_point() + scale_x_log10()
```



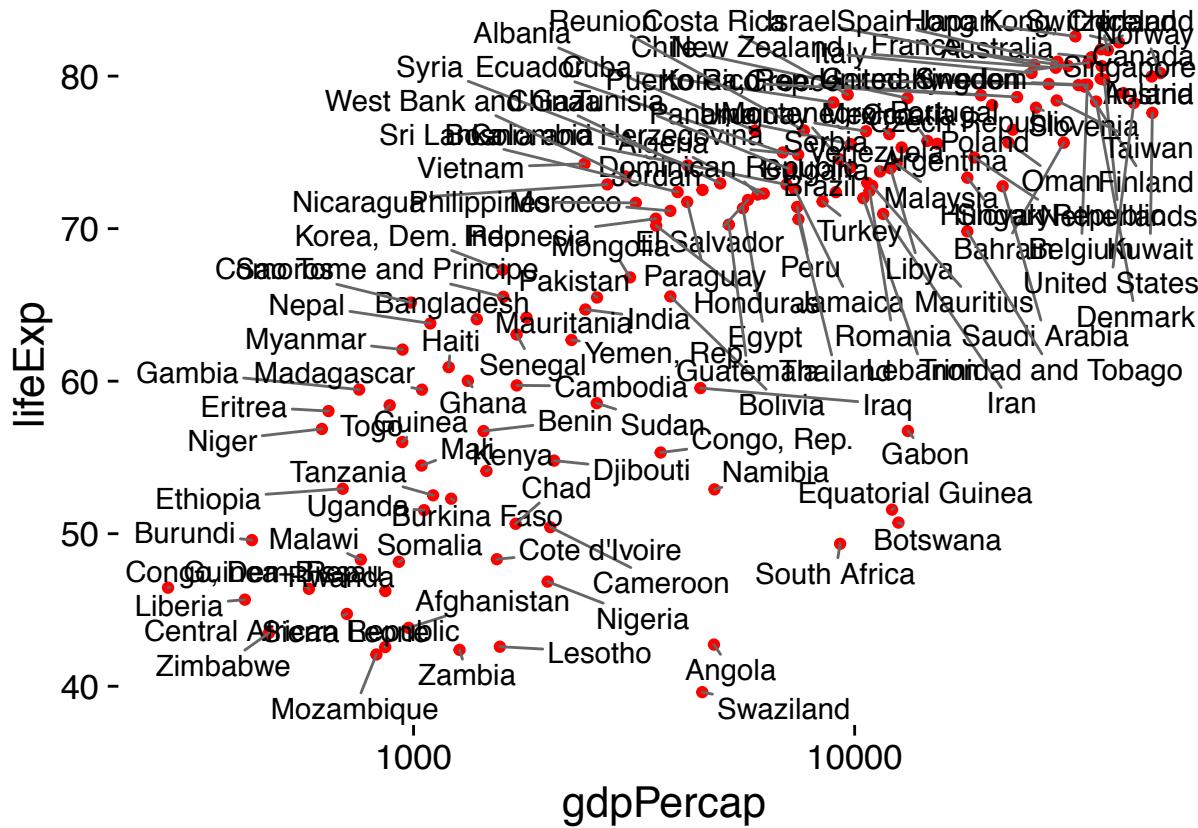
We can also do this:

```
p2 <- ggplot(filter(gapminder, year==2007),  
             aes(log10(gdpPercap), y=lifeExp))  
p2 + geom_point()
```



The first day we showed how to create a figure like this but with names. Now we can understand that code:

```
library(ggrepel)
p + geom_point(color = 'red') +
  geom_text_repel(aes(label = country)) +
  scale_x_log10() +
  theme_classic(base_size = 16)
```

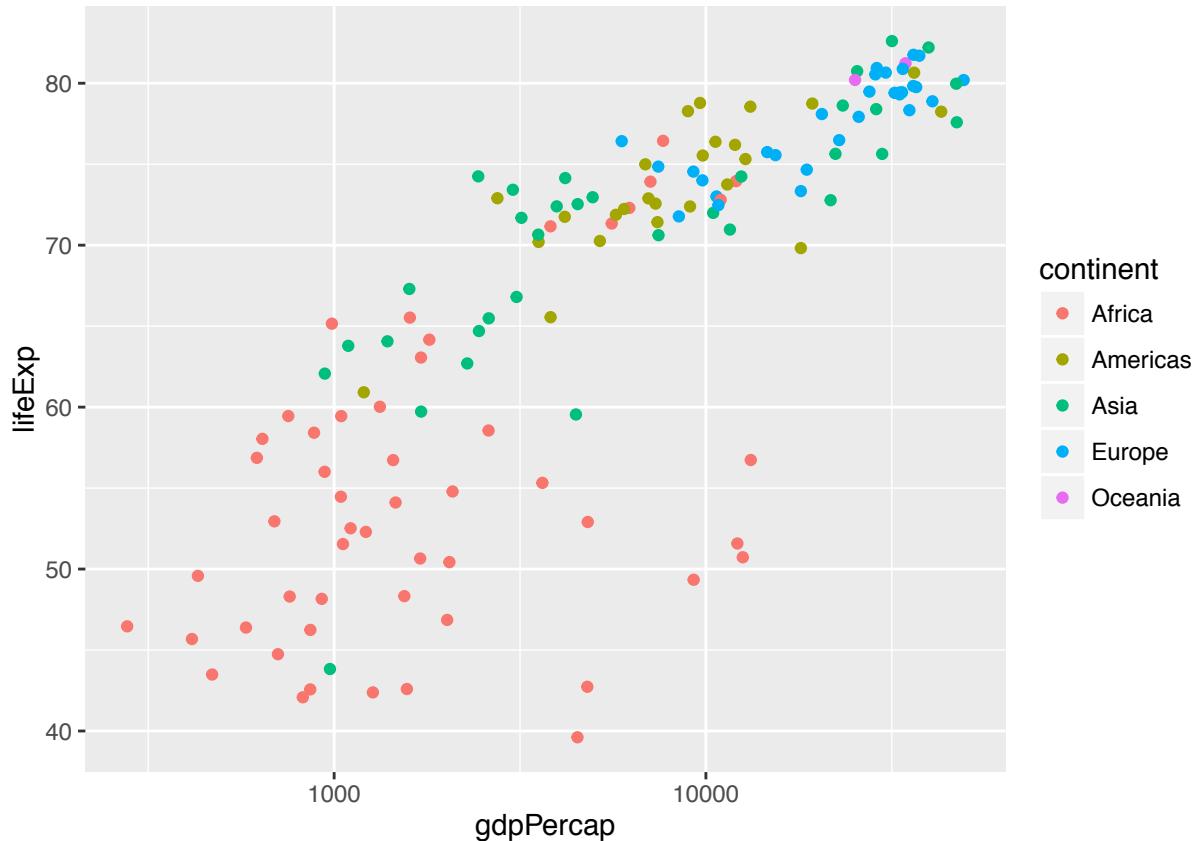


## Stratification

One of the most powerful operation we perform in EDA relates to stratifying the data.

When we are examining the relationship between two variables, it is possible that another variable is clouding the relationship. For example, for the data we are examining here, it is possible that what drives the apparent relationship is continent. For example, that OECD countries are both rich and healthy, but that within continent there is no relationship between income and life expectancy. By stratifying the data and re-examine the plot we can check for this possibility. We will learn other ways to do this, but a simple way we can stratify is by using different colors. Here is the same plot with color representing continent.

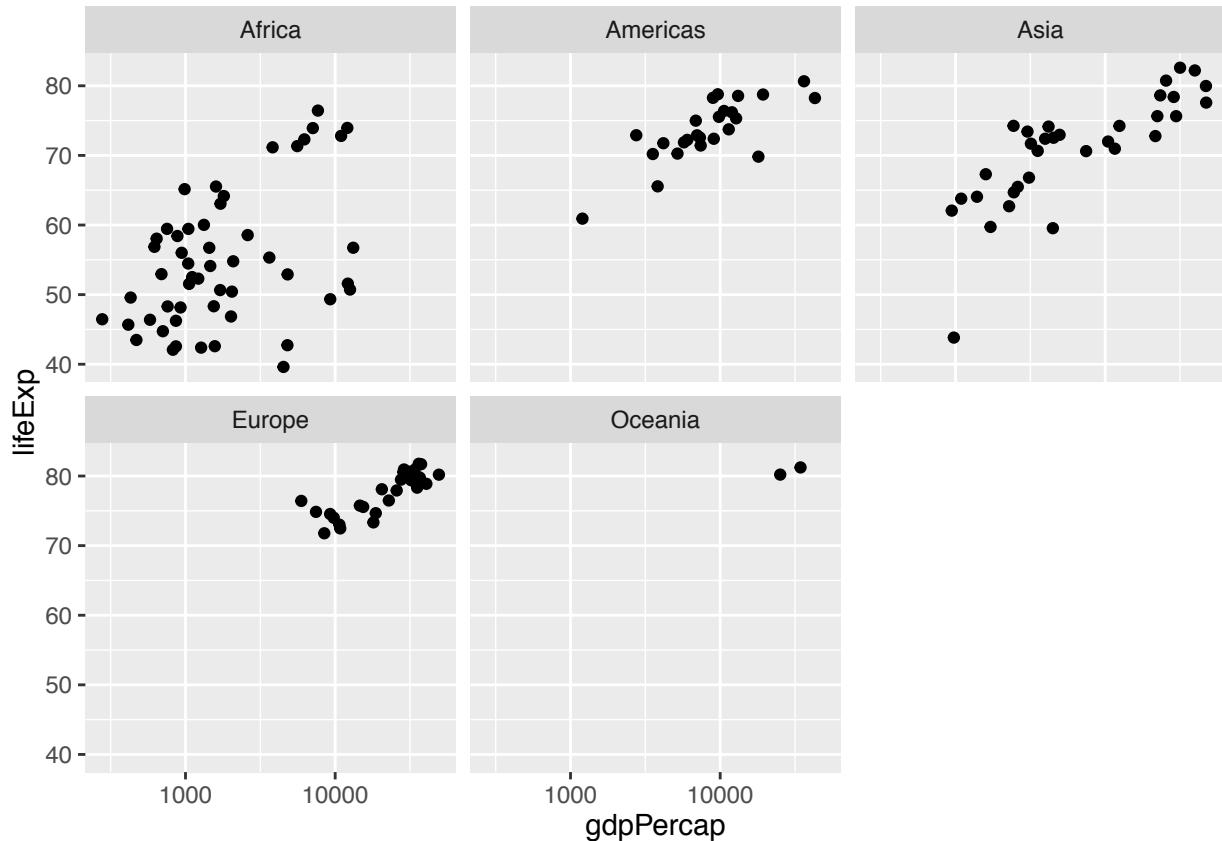
```
p <- ggplot(filter(gapminder, year==2007),
            aes(gdpPercap, y=lifeExp))
p + geom_point(aes(color=continent)) + scale_x_log10()
```



Note that this plot reveals that the relationship is indeed weaker within Africa. Although it seems that the trend still holds. Next we learn about faceting which helps us visualize if in fact the different continents are driving this.

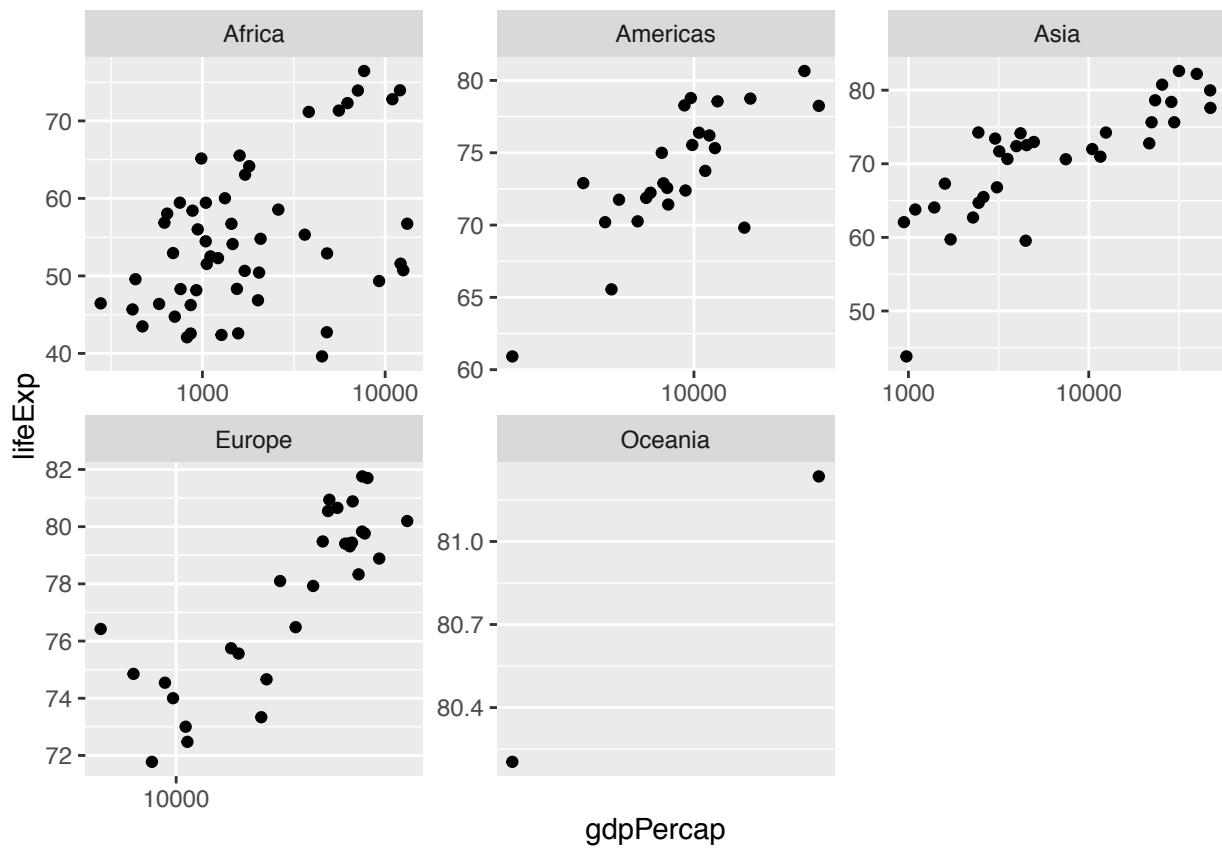
**Facets** We can examine different strata by faceting. The idea is simply to split up your data by one or more variables and then plot the subsets of data together. The `facet_wrap` function gives us flexibility on how this happens. Here is an example

```
p + geom_point() +
  facet_wrap("continent") + scale_x_log10()
```



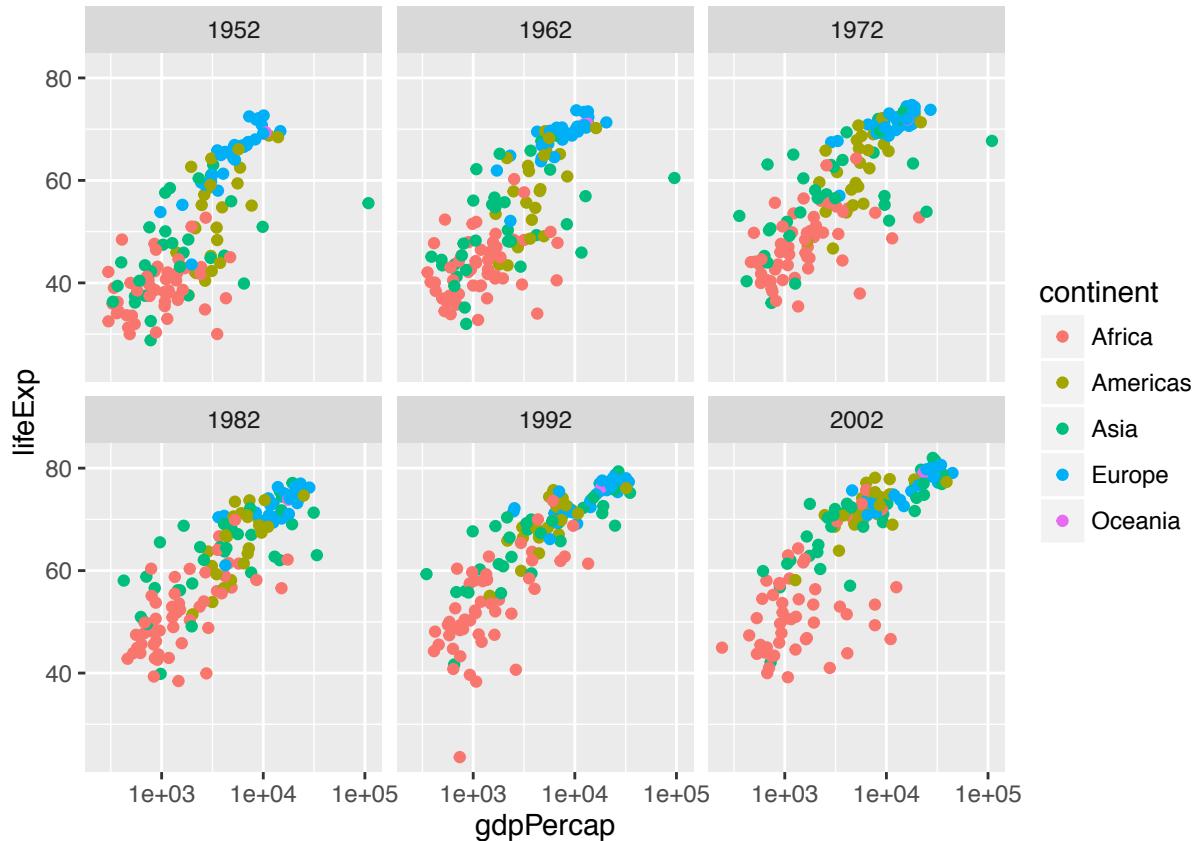
To take a closer look, letting the scale of the axis change, we can add the following argument:

```
p + geom_point() +
  facet_wrap(~continent, scale="free") + scale_x_log10()
```



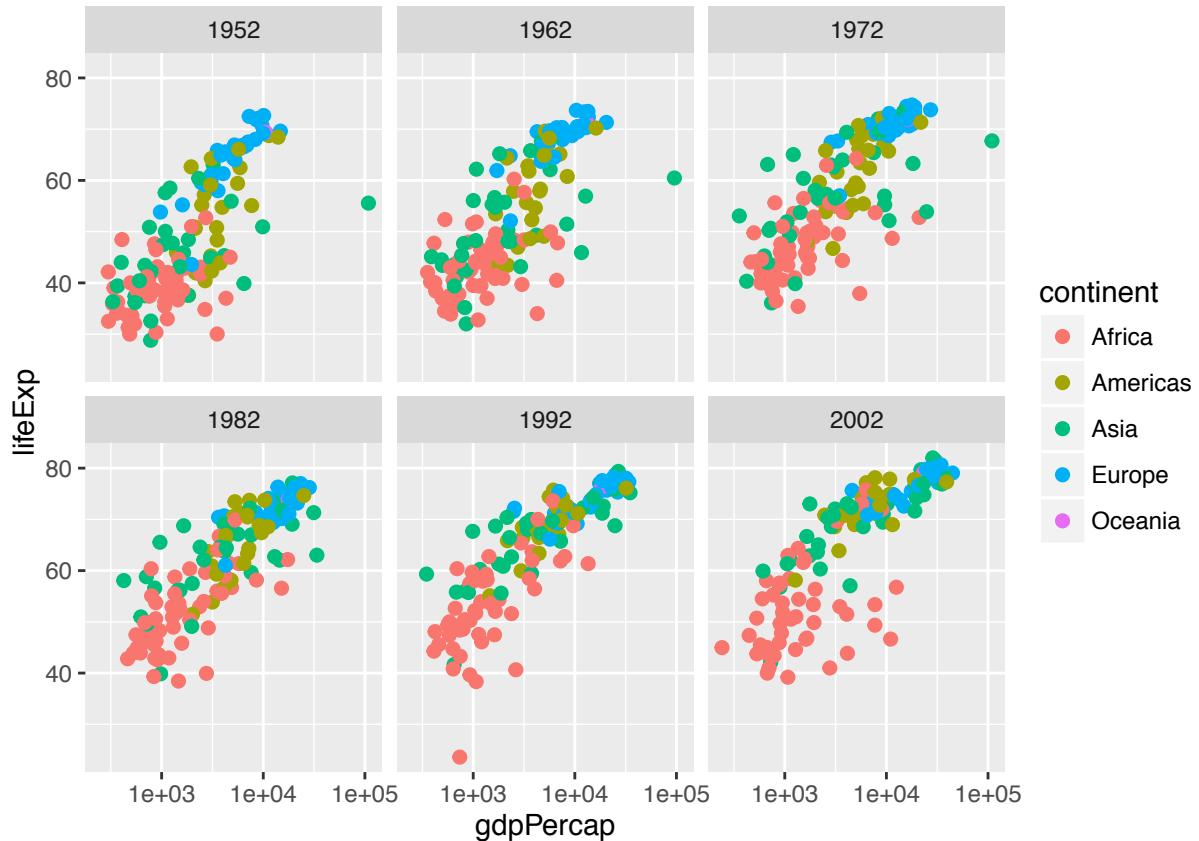
Assessment: Examine the relationship for years 1952, 1962, ..., 2002. Use color to distinguish the continents.

```
years <- seq(1952,2002,10)
p2 <- ggplot(filter(gapminder, year %in% years),
            aes(x=gdpPercap,y=lifeExp))
p2 + geom_point(aes(color=continent)) + facet_wrap(~year) + scale_x_log10()
```



Assessment. The points seem a bit small. Read the cheat sheet and learn how to increase the size of points.

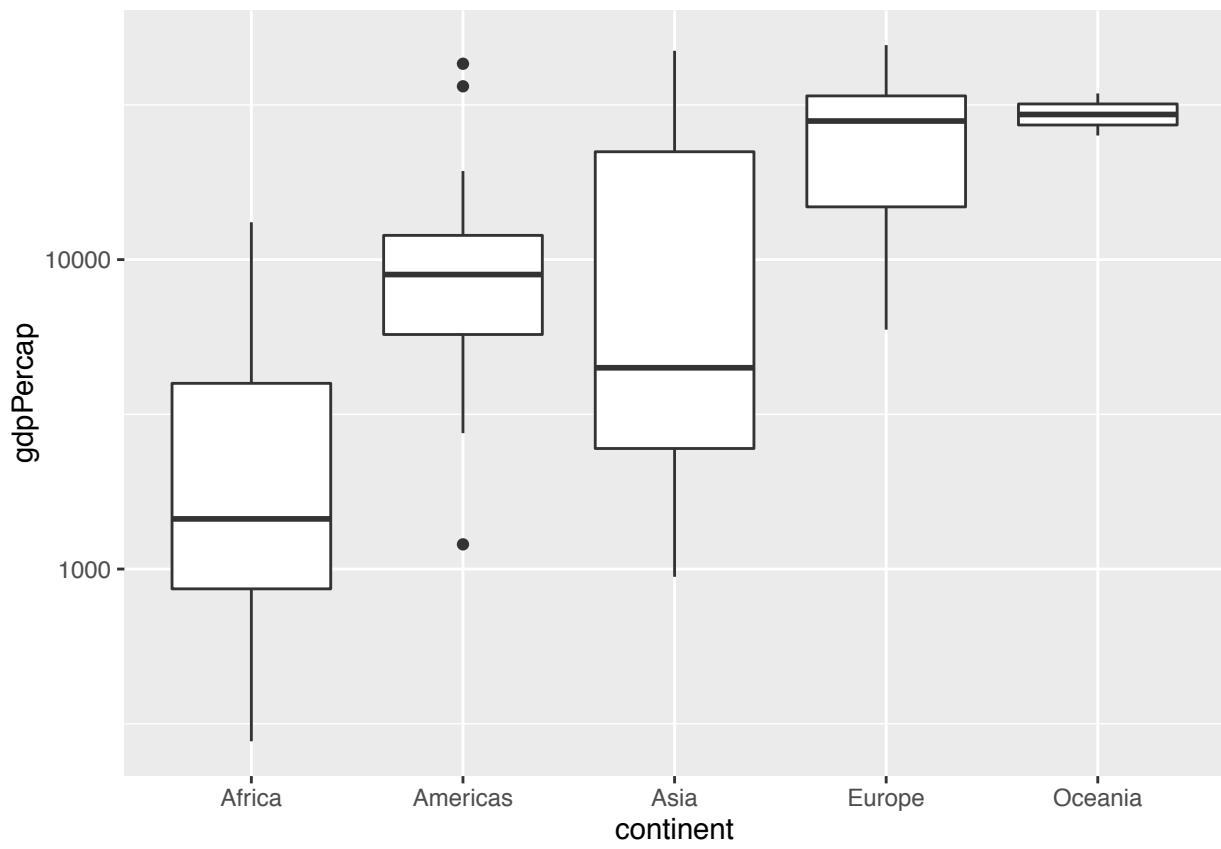
```
years <- seq(1952,2002,10)
p2 <- ggplot(filter(gapminder, year %in% years),
             aes(x=gdpPerCap,y=lifeExp))
p2 + geom_point(aes(color=continent), size=2) + facet_wrap(~year) + scale_x_log10()
```



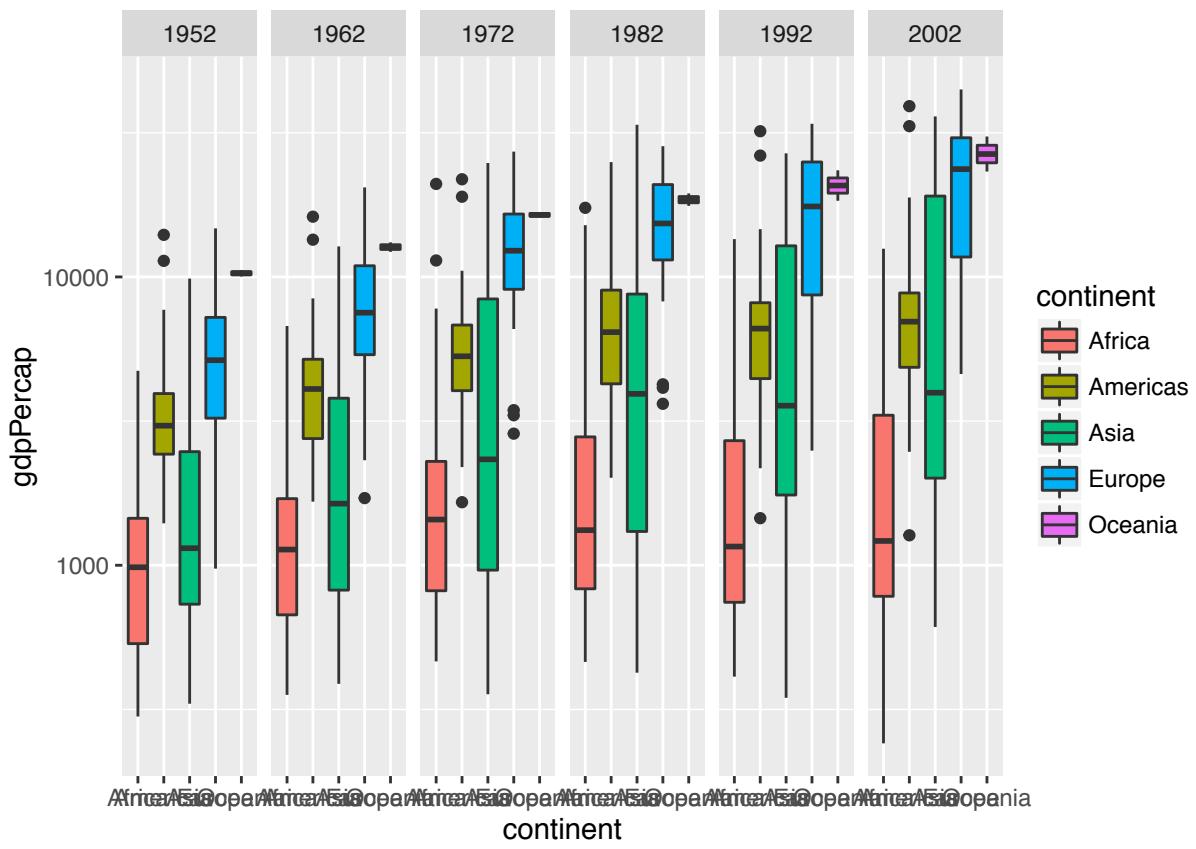
These plot clearly shows that the relationship is still present in each continent and year to year.

**Boxplot** Stratifying and the creating boxplot is one of the most useful forms of EDA. For example, we can

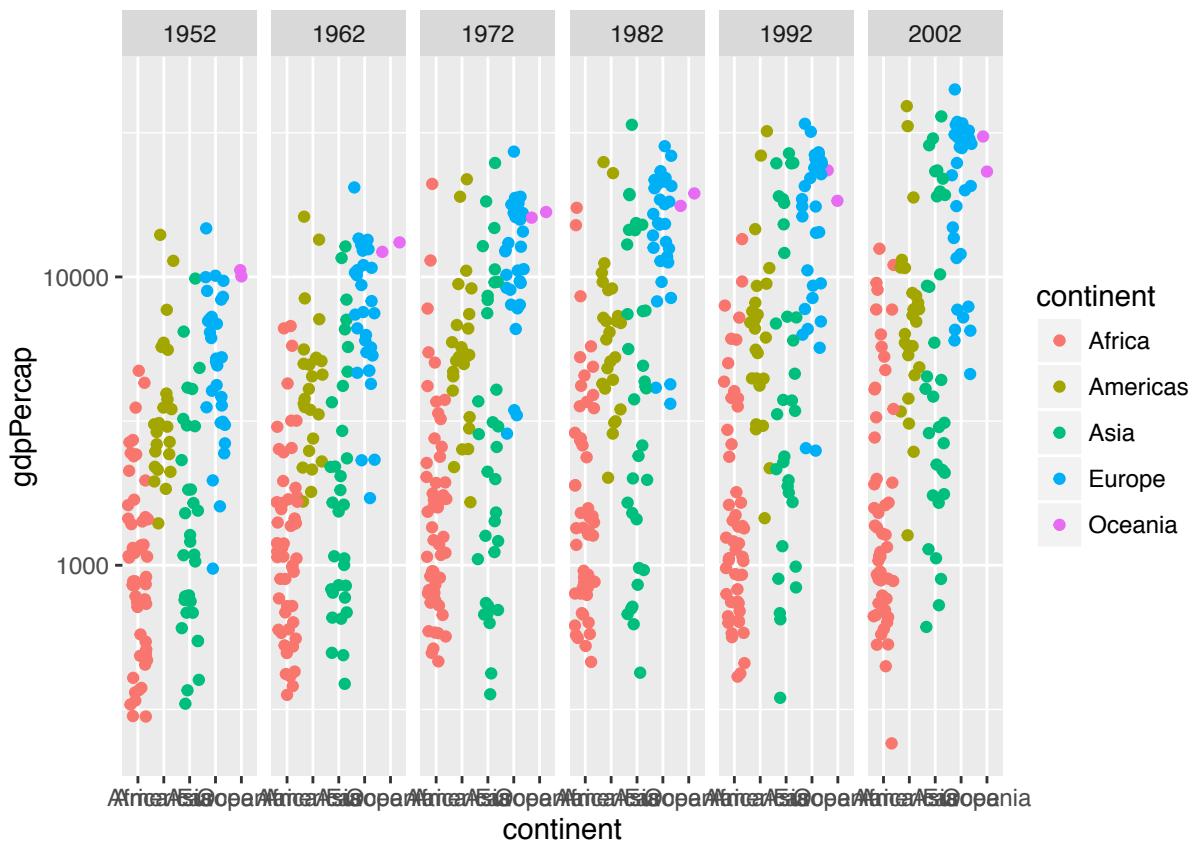
```
p2 <- ggplot(filter(gapminder, year==2007),
              aes(continent, y=gdpPerCap))
p2 + geom_boxplot() + scale_y_log10()
```



```
p2 <- ggplot(filter(gapminder, year%in%years & country!="Kuwait"),
  aes(continent, y=gdpPerCap))
p2 + geom_boxplot(aes(fill=continent)) + facet_grid(.~year)+scale_y_log10()
```



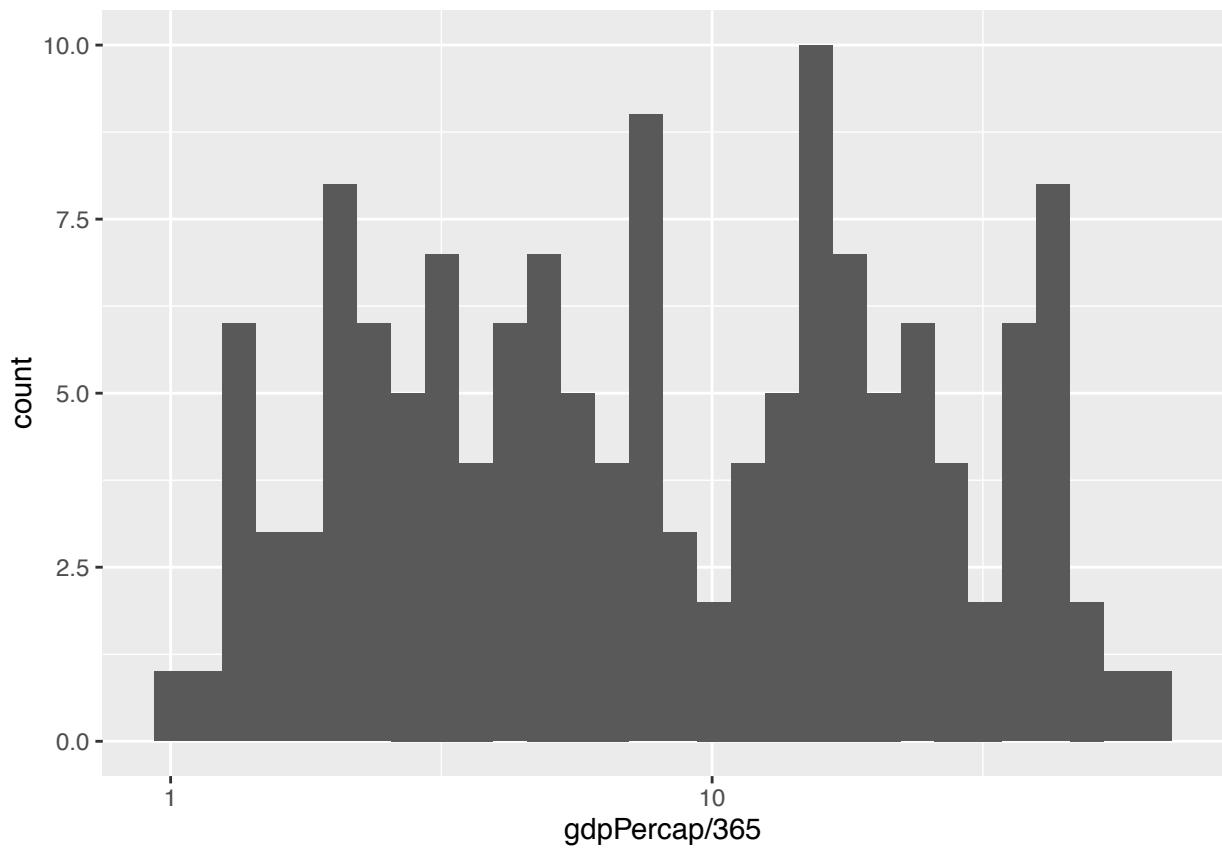
```
p2 <- ggplot(filter(gapminder, year%in%years & country!="Kuwait"),
              aes(continent, y=gdpPercap))
p2 + geom_jitter(aes(color=continent)) + facet_grid(.~year)+scale_y_log10()
```



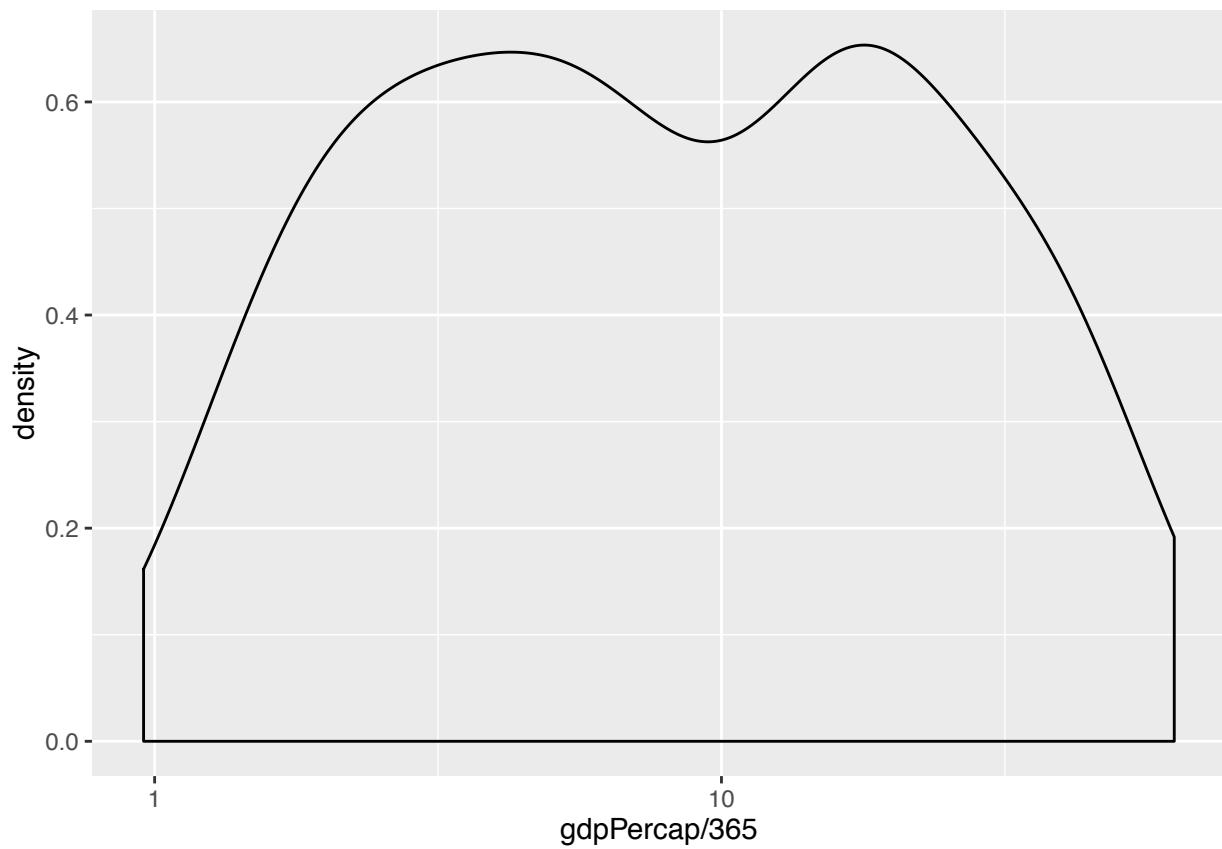
## Densities

```
p2 <- ggplot(filter(gapminder, year==1967 & country!="Kuwait"),
              aes(x=gdpPercap/365))
p2 + geom_histogram() + scale_x_log10()

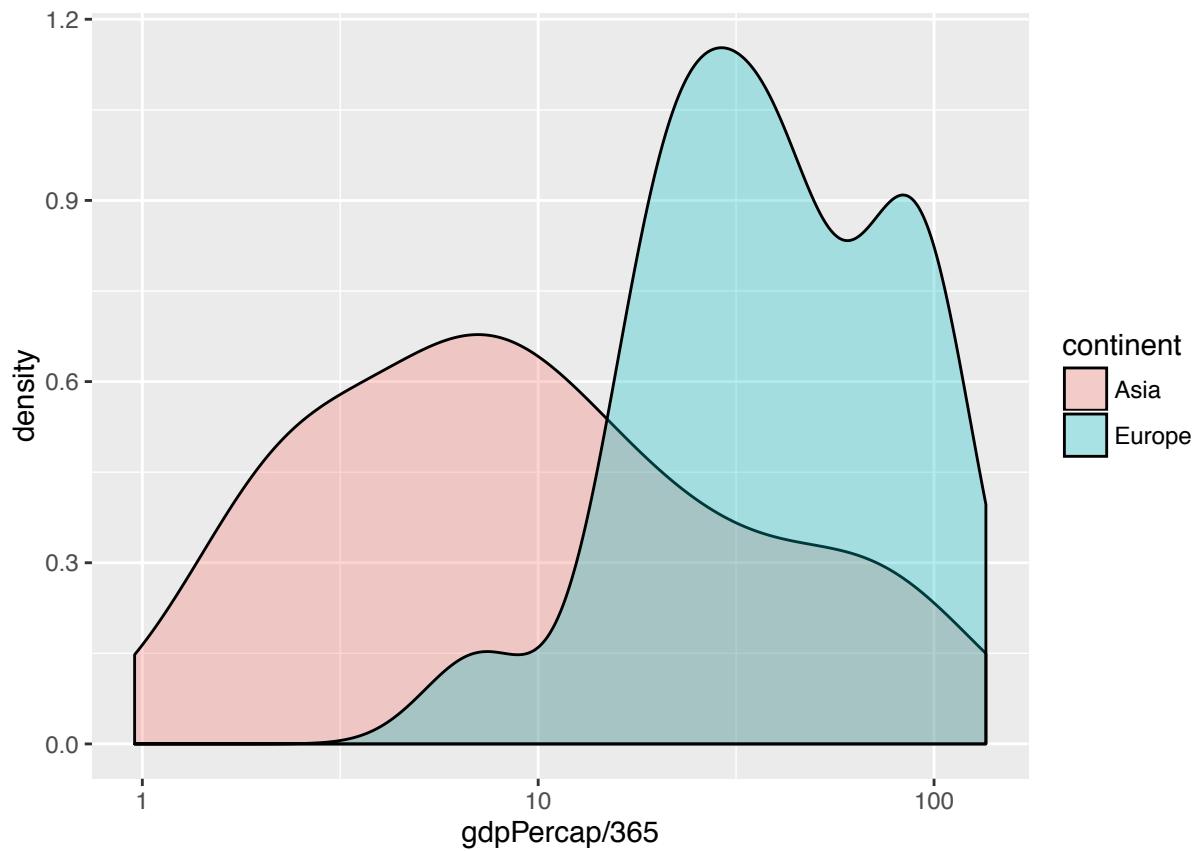
## `stat_bin()` using `bins = 30` . Pick better value with `binwidth` .
```

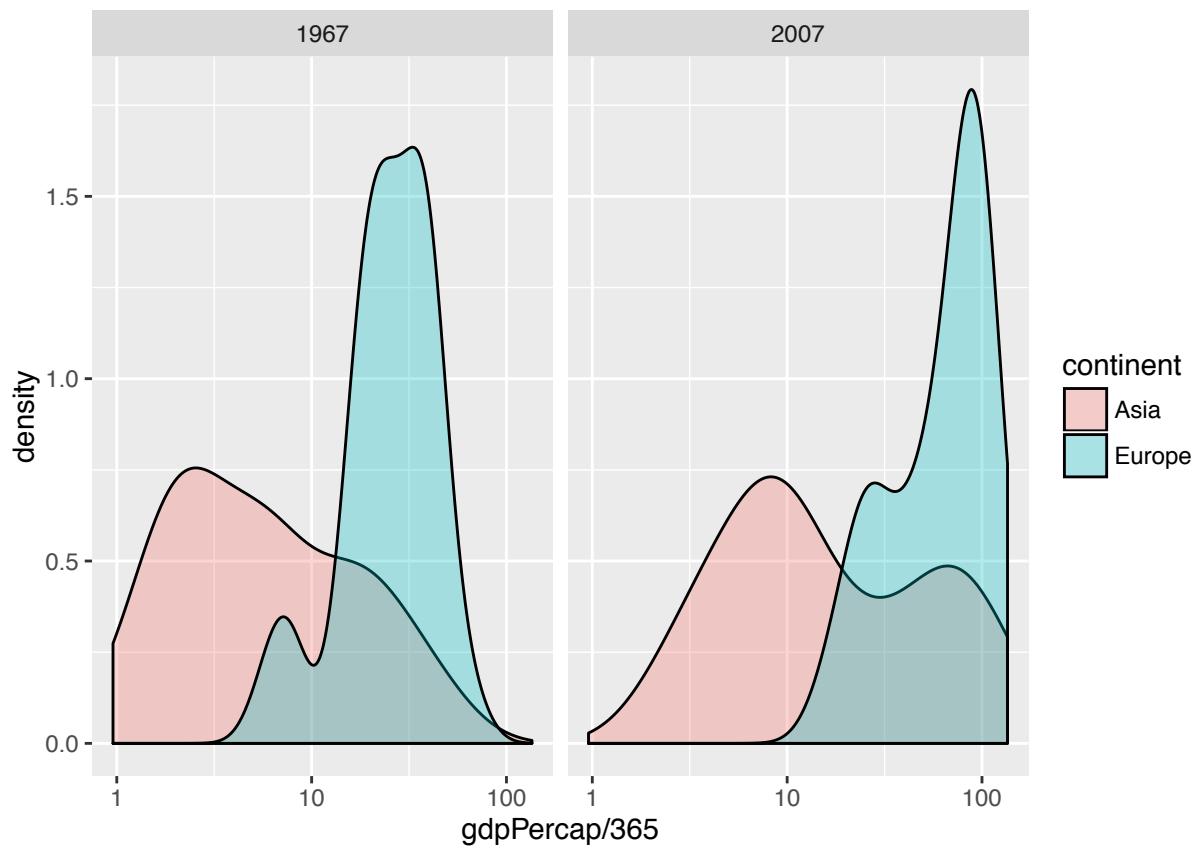


```
p2 + geom_density() + scale_x_log10()
```



```
p2 <- ggplot(filter(gapminder, year%in%c(1967,2007) & country!="Kuwait" & continent%in%c("Europe","Asia"))
              aes(x=gdpPercap/365,fill=continent) )
p2 + geom_density(alpha=.3) + scale_x_log10()
```





## Cheatsheets

- [Data Visualization with ggplot2 from RStudio](#)

# Homework 1: Insights on Poverty

*February 4, 2016*

This homework is due Sunday February 14, 2016 at 11:59 PM. When complete, submit your code in the R Markdown file and the knitted HTML file on Canvas.

## Background

This HW is based on Hans Rosling talks [New Insights on Poverty](#) and [The Best Stats You've Ever Seen](#).

The assignment uses data to answer specific question about global health and economics. The data contradicts commonly held preconceived notions. For example, Hans Rosling starts his talk by asking: (paraphrased) “for each of the six pairs of countries below, which country do you think had the highest child mortality in 2015?”

1. Sri Lanka or Turkey
2. Poland or South Korea
3. Malaysia or Russia
4. Pakistan or Vietnam
5. Thailand or South Africa

Most people get them wrong. Why is this? In part it is due to our preconceived notion that the world is divided into two groups: the *Western world* versus the *third world*, characterized by “long life, small family” and “short life, large family” respectively. In this homework we will use data visualization to gain insights on this topic.

## Problem 1

The first step in our analysis is to download and organize the data. The necessary data to answer these question is available on the [gapminder](#) website.

### Problem 1.1

We will use the following datasets:

1. [Childhood mortality](#)
2. [Life expectancy](#)
3. [Fertility](#)
4. [Population](#)
5. [Total GDP](#)

Create five `tbl_df` table objects, one for each of the tables provided in the above files. Hints: Use the `read_csv` function. Because these are only temporary files, give them short names.

```
library(tidyr)
library(dplyr)
```

```

## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

library(readr)
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.4

theme_set(theme_bw()) ##optional

t1 <- read_csv("http://spreadsheets.google.com/pub?key=0ArfEDsV3bBwCcGhBd2NOQVZ1eWowNVpSNjl1c3lRSWc&output=csv")
t2 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyj2tPLxKvvnNPA&output=csv")
t3 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyj0TA1JeCEzcGQ&output=csv")
t4 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyj0XOoBL_n5tAQ&output=csv")
t5 <- read_csv("http://spreadsheets.google.com/pub?key=pyj6tScZqmEfi4sLVvEqThw&output=csv")
### To keep track of what these are we will define a vector of values
value_names <- c("child_mortality", "life_expectancy", "fertility", "population", "gdp")

```

## Problem 1.2

Write a function called `my_func` that takes a table as an argument and returns the column name. For each of the five tables, what is the name of the column containing the country names? Print out the tables or look at them with `View` to determine the column.

```

my_func <- function(x){
  names(x)[1]
}

answer <- c( my_func(t1), my_func(t2), my_func(t3), my_func(t4), my_func(t5) )

answer

## [1] "Under five mortality"
## [2] "Life expectancy with projections. Yellow is IHME"
## [3] "Total fertility rate"
## [4] "Total population"
## [5] "GDP (constant 2000 US$)"

```

## Problem 1.3

In the previous problem we noted that gapminder is inconsistent in naming their country column. Fix this by assigning a common name to this column in the various tables.

```

the_name <- "country"
names(t1)[1] <- the_name
names(t2)[1] <- the_name
names(t3)[1] <- the_name
names(t4)[1] <- the_name
names(t5)[1] <- the_name

```

## Problem 1.4

Notice that in these tables, years are represented by columns. We want to create a tidy dataset in which each row is a unit or observation and our 5 values of interest, including the year for that unit, are in the columns. The unit here is a country/year pair and each unit gets values:

```

value_names

## [1] "child_mortality" "life_expectancy" "fertility"           "population"
## [5] "gdp"

```

We call this the *long* format. Use the `gather` function from the `tidyverse` package to create a new table for childhood mortality using the long format. Call the new columns `year` and `child_mortality`

```
gather(t1, key=year, value=child_mortality, -country, convert = TRUE)
```

```

## Source: local data frame [59,400 x 3]
##
##              country   year child_mortality
##              (chr)  (int)          (dbl)
## 1          Abkhazia 1800            NA
## 2      Afghanistan 1800        468.58
## 3 Akrotiri and Dhekelia 1800            NA
## 4         Albania 1800        375.20
## 5         Algeria 1800        460.21
## 6 American Samoa 1800            NA
## 7       Andorra 1800            NA
## 8         Angola 1800        485.68
## 9       Anguilla 1800            NA
## 10 Antigua and Barbuda 1800        473.60
## ...          ...    ...

```

Now redefine the remaining tables in this way.

```

t1 <- gather(t1, key=year, value=child_mortality, -country, convert = TRUE)
t2 <- gather(t2, key=year, value=life_expectancy, -country, convert = TRUE)
t3 <- gather(t3, key=year, value=fertility, -country, convert = TRUE)
t4 <- gather(t4, key=year, value=population, -country, convert = TRUE)
t5 <- gather(t5, key=year, value=gdp, -country, convert = TRUE)

```

## Problem 1.5

Now we want to join all these files together. Make one consolidated table containing all the columns

```
dat <- t1 %>% full_join(t2) %>% full_join(t3) %>% full_join(t4) %>% full_join(t5)
```

```
## Joining by: c("country", "year")
```

## Problem 1.6

Add a column to the consolidated table containing the continent for each country. Hint: We have created a file that maps countries to continents [here](#). Hint: Learn to use the `left_join` function.

```
map <- read_delim("https://raw.githubusercontent.com/datasetsciencelabs/data/master/homework_data/continents.csv")
dat <- left_join(dat, map, continent = continents)
```

```
## Joining by: "country"
```

Advanced solution:

```
t1 <- read_csv("http://spreadsheets.google.com/pub?key=0ArfEDsV3bBwCcGhBd2N0QVZ1eWowNVpSNjl1c3lRSWc&output=csv")
t2 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyj2tPLxKvvnNPA&output=csv")
t3 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyj0TA1JeCEzcGQ&output=csv")
t4 <- read_csv("http://spreadsheets.google.com/pub?key=phAwcNAVuyjOXOoBL_n5tAQ&output=csv")
t5 <- read_csv("http://spreadsheets.google.com/pub?key=pyj6tScZqmEfI4sLVvEQtHw&output=csv")
### To keep track of what these are we will define a vector of values
value_names <- c("child_mortality", "life_expectancy", "fertility", "population", "gdp")
l <- list(t1, t2, t3, t4, t5)
rm(t1, t2, t3, t4, t5)
gc()
fix_table <- function(tab, val_name){
  names(tab)[1] <- "country"
  tab <- gather(tab, key=year, value=y, -country, convert = TRUE)
  names(tab)[which(names(tab)=="y")] <- val_name
  return(tab)
}
for(i in seq_along(l) ) l[[i]] <- fix_table(l[[i]], value_names[i])
dat <- Reduce(full_join, l)

dat <- left_join(dat, map, continent = continents)
```

## Problem 2

Report the child mortality rate in 2015 for these 5 pairs:

1. Sri Lanka or Turkey
2. Poland or South Korea
3. Malaysia or Russia
4. Pakistan or Vietnam
5. Thailand or South Africa

```

countries <- c("Sri Lanka", "Turkey", "Poland", "South Korea", "Malaysia", "Russia", "Pakistan", "Vietnam")
answer <- filter(dat, country %in% countries & year == 2015) %>% select(country, child_mortality)
### optionally we can plot them next to each other
answer <- bind_cols( slice(answer, match(countries[ seq(1,9,2)], country)),
                     slice(answer, match(countries[ seq(2,10,2)], country)))
answer

## Source: local data frame [5 x 4]
##
##   country child_mortality      country child_mortality
##   (chr)          (dbl)      (chr)          (dbl)
## 1 Sri Lanka        8.7    Turkey        13.5
## 2 Poland          5.2  South Korea       3.5
## 3 Malaysia         8.2     Russia        9.6
## 4 Pakistan        81.1    Vietnam       21.7
## 5 Thailand        12.3  South Africa     42.1

```

## Problem 3

To examine if in fact there was a long-life-in-a-small-family and short-life-in-a-large-family dichotomy, we will visualize the average number of children per family (fertility) and the life expectancy for each country.

### Problem 3.1

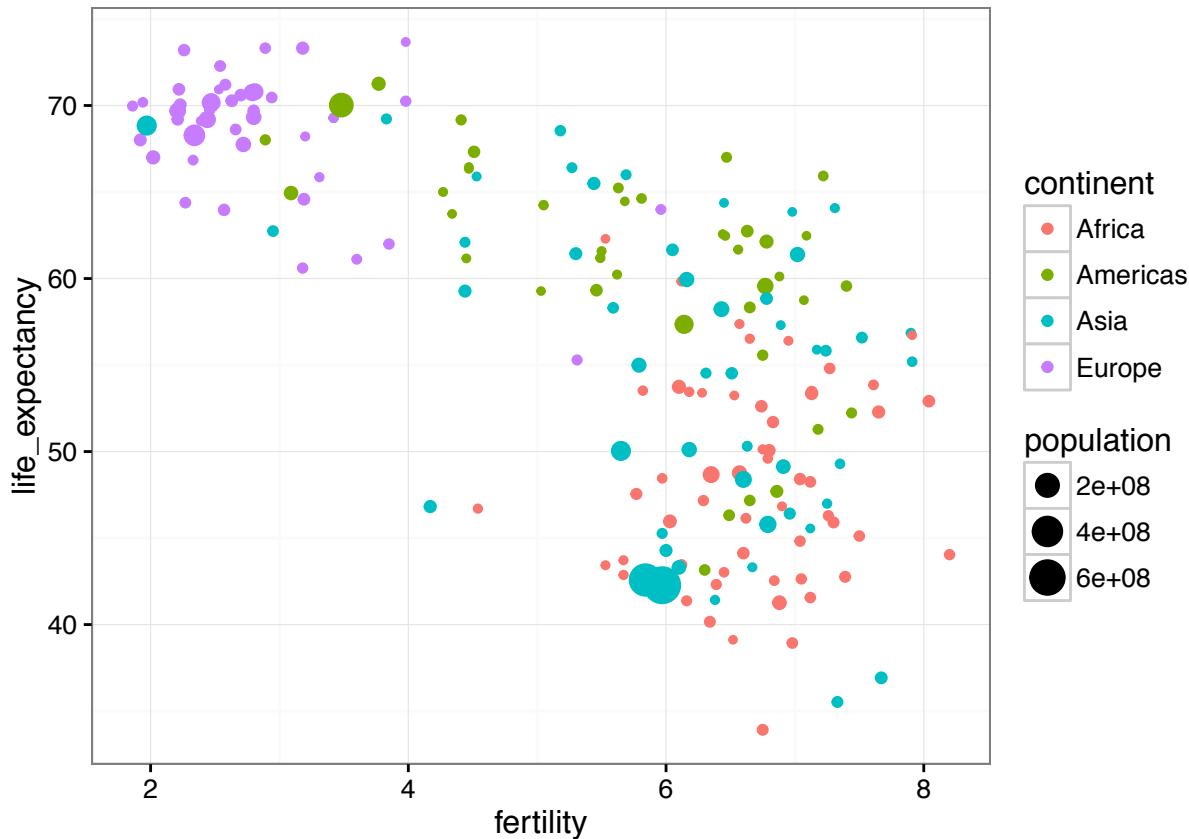
Use `ggplot2` to create a plot of life expectancy versus fertility for 1962 for Africa, Asia, Europe, and the Americas. Use color to denote continent and point size to denote population size:

```

p <- ggplot( filter(dat, year == 1962 & continent != "Oceania"), aes(x=fertility, y=life_expectancy))
p + geom_point( aes( color=continent, size=population))

## Warning: Removed 44 rows containing missing values (geom_point).

```



Do you see a dichotomy? Explain.

### Problem 3.2

Now we will annotate the plot to show different types of countries.

Learn about OECD and OPEC. Add a couple of columns to your consolidated tables containing a logical vector that tells if a country is OECD and OPEC respectively. It is ok to base membership on 2015.

```
oecd <- c("Australia", "Austria", "Belgium", "Canada", "Chile", "Country", "Czech Republic", "Denmark", "Estonia",
    "Iceland", "Ireland", "Israel", "Italy", "Japan", "Korea", "Luxembourg", "Mexico", "Netherlands", "New Zealand",
    "Slovak Republic", "Slovenia", "Spain", "Sweden", "Switzerland", "Turkey", "United Kingdom", "United States")

opec <- c("Algeria", "Angola", "Ecuador", "Iran", "Iraq", "Kuwait", "Libya", "Nigeria",
    "Qatar", "Saudi Arabia", "United Arab Emirates", "Venezuela")

dat <- mutate(dat, oecd = country %in% oecd, opec = country %in% opec)
```

### Problem 3.3

Make the same plot as in Problem 3.1, but this time use color to annotate the OECD countries and OPEC countries. For countries that are not part of these two organization annotate if they are from Africa, Asia, or the Americas.

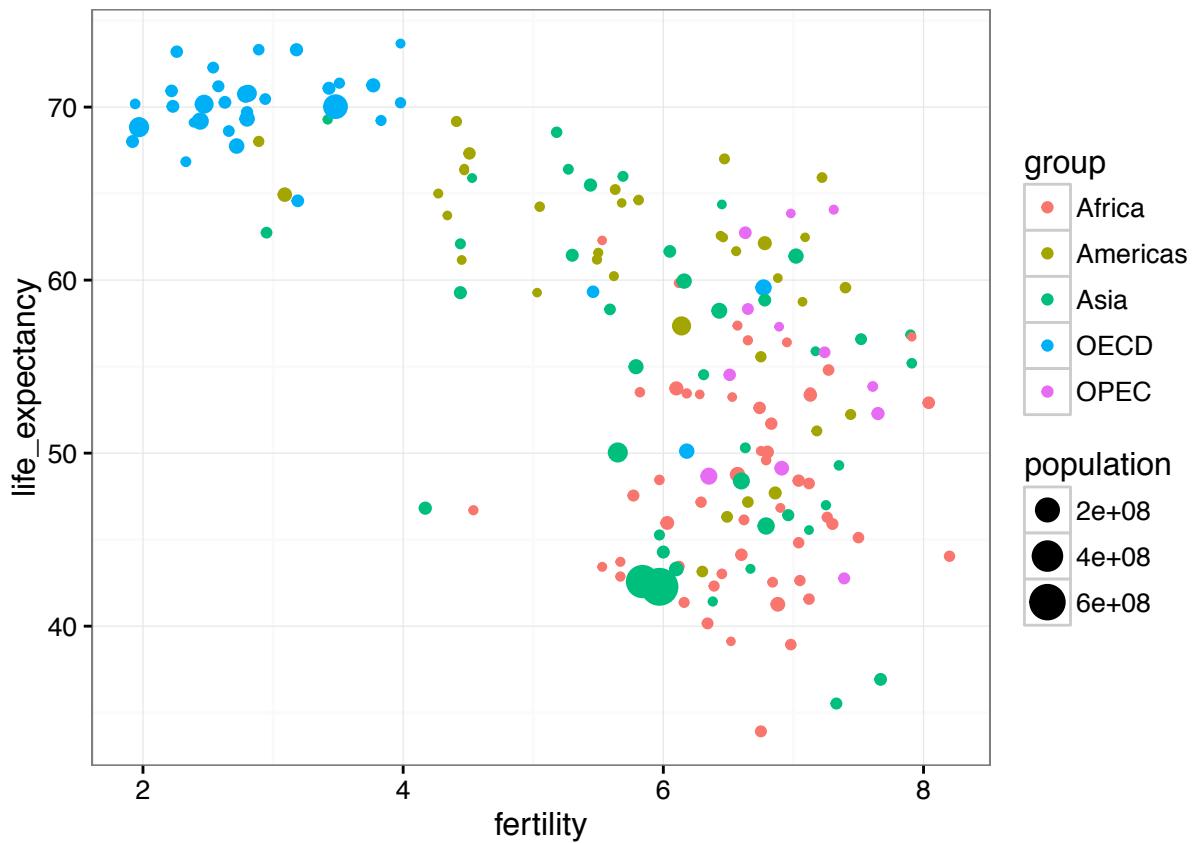
```

group <- ifelse( dat$country%in%oecd , "OECD", dat$continent )
group <- ifelse( dat$country%in%opec , "OPEC", group )
group <- ifelse( group%in%c("Europe","Oceania"), NA, group)

##add group and avoid NAs
dat2 <- mutate(dat, group=group) %>%
  filter(!is.na(group) & !is.na(population) & !is.na(fertility) & !is.na(life_expectancy))

p <- ggplot( mutate(dat2, group = group) %>% filter(year==1962),
             aes(x=fertility, y=life_expectancy))
p + geom_point( aes( color=group, size=population))

```



How would you describe the dichotomy?

### Problem 3.4

Explore how this figure changes across time. Show us 4 figures that demonstrate how this figure changes through time.

```

years <- c(1960, 1980, 2000, 2015)
p <- ggplot( mutate(dat2, group = group) %>% filter(year%in%years),
             aes(x=fertility, y=life_expectancy))
p + geom_point( aes( color=group, size=population)) + facet_wrap(~year)

```



Would you say that the same dichotomy exists today? Explain:

Answer: The dichotomy today is not as clear. If there is a dichotomy it is mainly between sub-saharan Africa and the rest of the world. The transition has been relatively smooth. Extra points: You can see the effects of the AIDS epidemic which may explain why life expectancy is down for some countries in 2000 compared to 1980.

### Problem 3.5 (Optional)

Make an animation with the `gganimate` package.

```
library(gganimate)
p <- ggplot( filter(dat2, year>1950),
  aes(x=fertility, y=life_expectancy)) + coord_cartesian(ylim = c(30, 85)) +
  geom_point( aes( color=group, size=population, frame=year))
gg_animate(p, "output.mp4")
```

## Problem 4

Having time as a third dimension made it somewhat difficult to see specific country trends. Let's now focus on specific countries.

## Problem 4.1

Let's compare France and its former colony Tunisia. Make a plot of fertility versus year with color denoting the country. Do the same for life expectancy. How would you compare Tunisia's improvement compared to France's in the past 60 years? Hint: use `geom_line`

```
countries <- c("Tunisia", "France")
p1 <- ggplot(filter(dat2, country %in% countries & year > 1959),
             aes(year, fertility, group=country, color=group)) + geom_line() + theme(legend.position="none")
p2 <- ggplot(filter(dat2, country %in% countries & year > 1959),
             aes(year, life_expectancy, group=country, color=group)) + geom_line() + theme(legend.position="none")

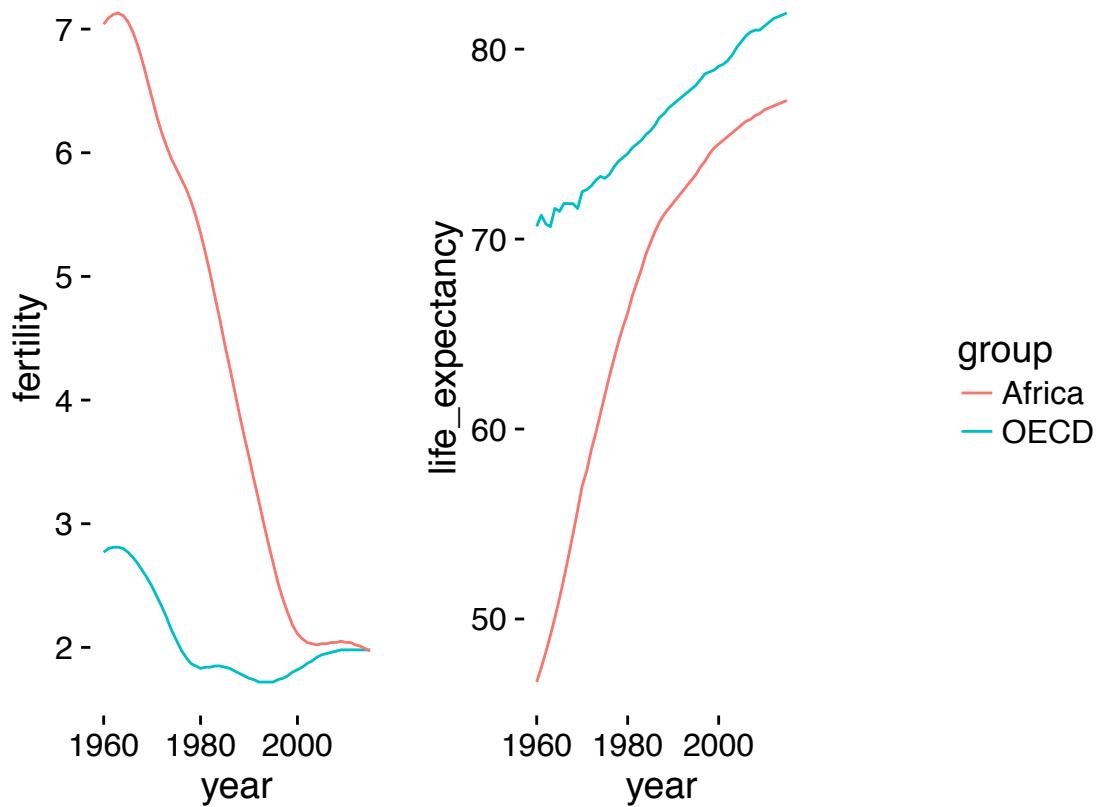
library(cowplot) ##this is optional

## 
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
## 
##     ggsave

g_legend<-function(a.gplot){
  tmp <- ggplot_gtable(ggplot_build(a.gplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}
pleg <- ggplot( filter(dat2,country%in%countries & year > 1959), aes(year,fertility,group=country,color=group))
pleg = g_legend(pleg)

plot_grid(p1, p2, pleg, ncol = 3)
```



### Problem 4.2

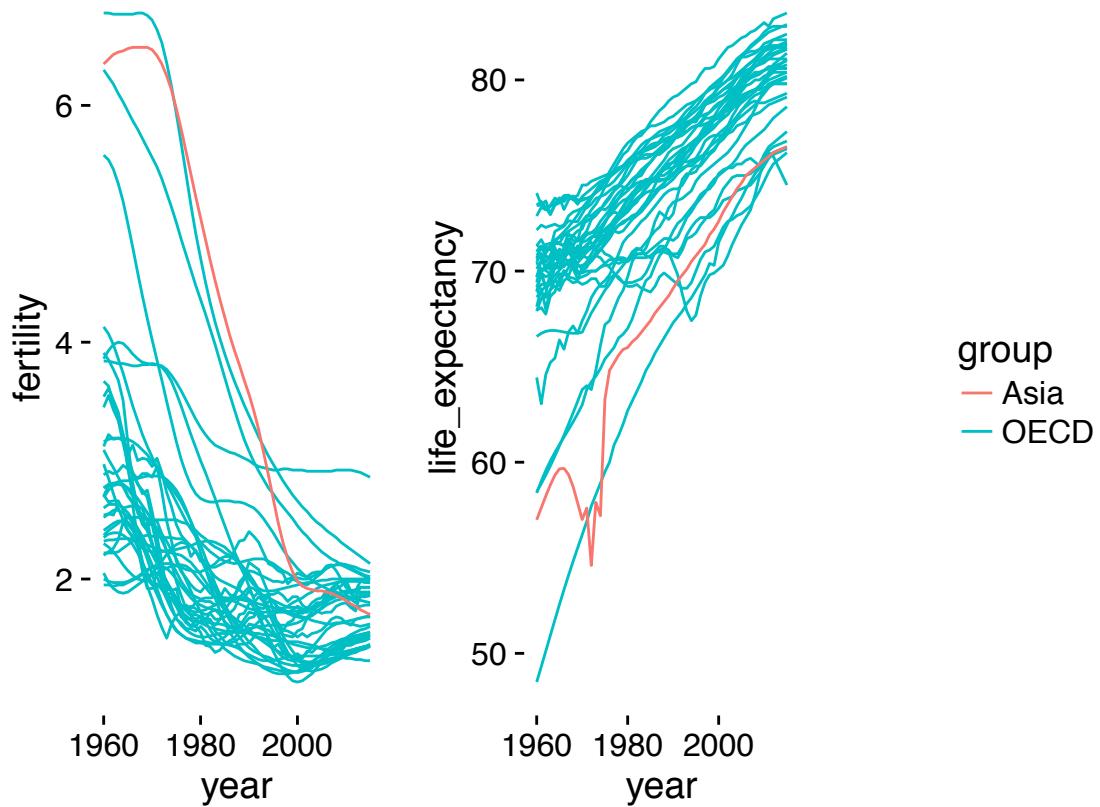
Do the same, but this time compare Vietnam to the OECD countries.

```

countries <- c("Vietnam", oecd)
p1 <- ggplot( filter(dat2, country%in%countries & year > 1959),
              aes(year, fertility, group=country, color=group)) + geom_line() + theme(legend.position="none")
p2 <- ggplot( filter(dat2, country%in%countries & year > 1959),
              aes(year, life_expectancy, group=country, color=group)) + geom_line() + theme(legend.position="none")
library(cowplot) ##this is optional
g_legend<-function(a.gplot){
  tmp <- ggplot_gtable(ggplot_build(a.gplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}
pleg <- ggplot( filter(dat2, country%in%countries & year > 1959), aes(year, fertility, group=country, color=group))
pleg = g_legend(pleg)

plot_grid(p1, p2, pleg, ncol = 3)

```



## Problem 5

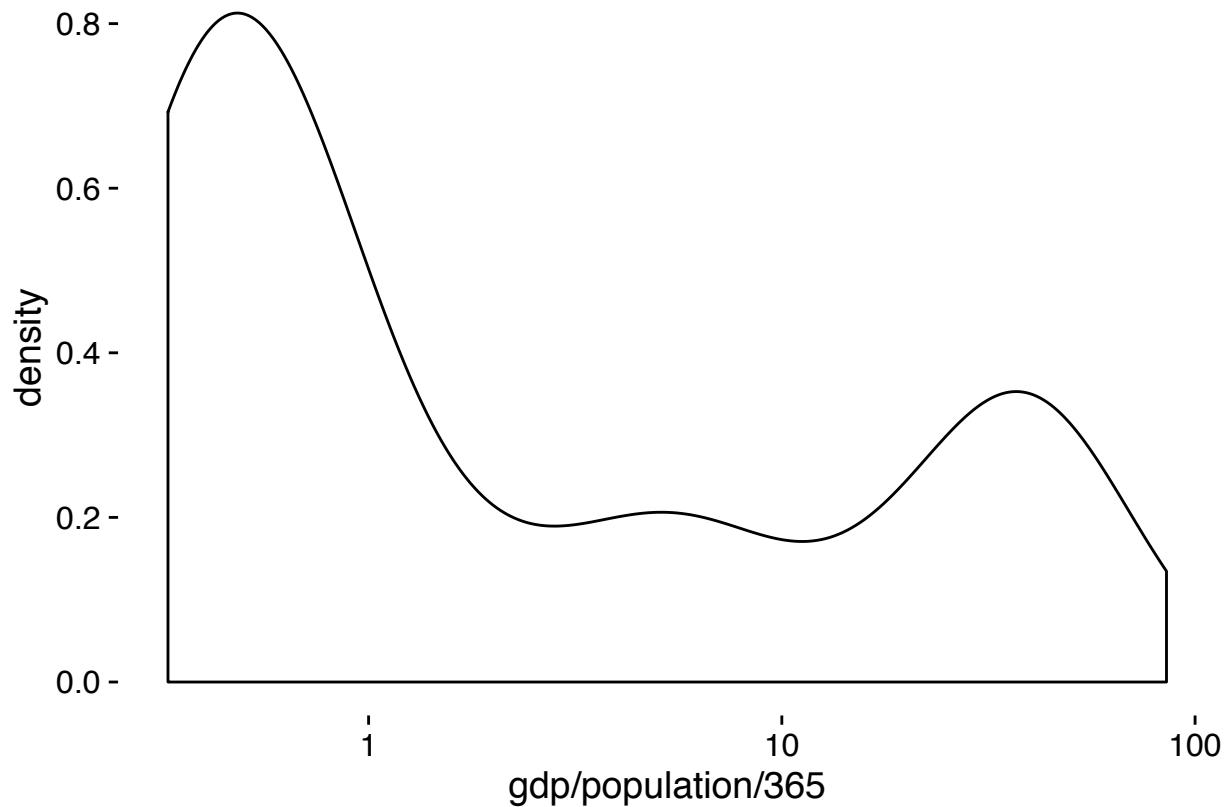
We are now going to examine GDP per capita per day.

### Problem 5.1

Create a smooth density estimate of the distribution of GDP per capita per day across countries in 1970. Include OECD, OPEC, Asia, Africa, and the Americas in the computation. When doing this we want to weigh countries with larger populations more. We can do this using the “weight” argument in `geom_density`.

```
dat2 <- mutate(dat, group=group) %>%
  filter(!is.na(group) & !is.na(population) & !is.na(gdp))

ggplot(filter(dat2, year==1970 & group%in%c("OECD", "OPEC", "Asia", "Africa", "Americas")),
       aes(x = gdp/population/365)) +
  geom_density(aes(weight=population/sum(population))) + scale_x_log10()
```

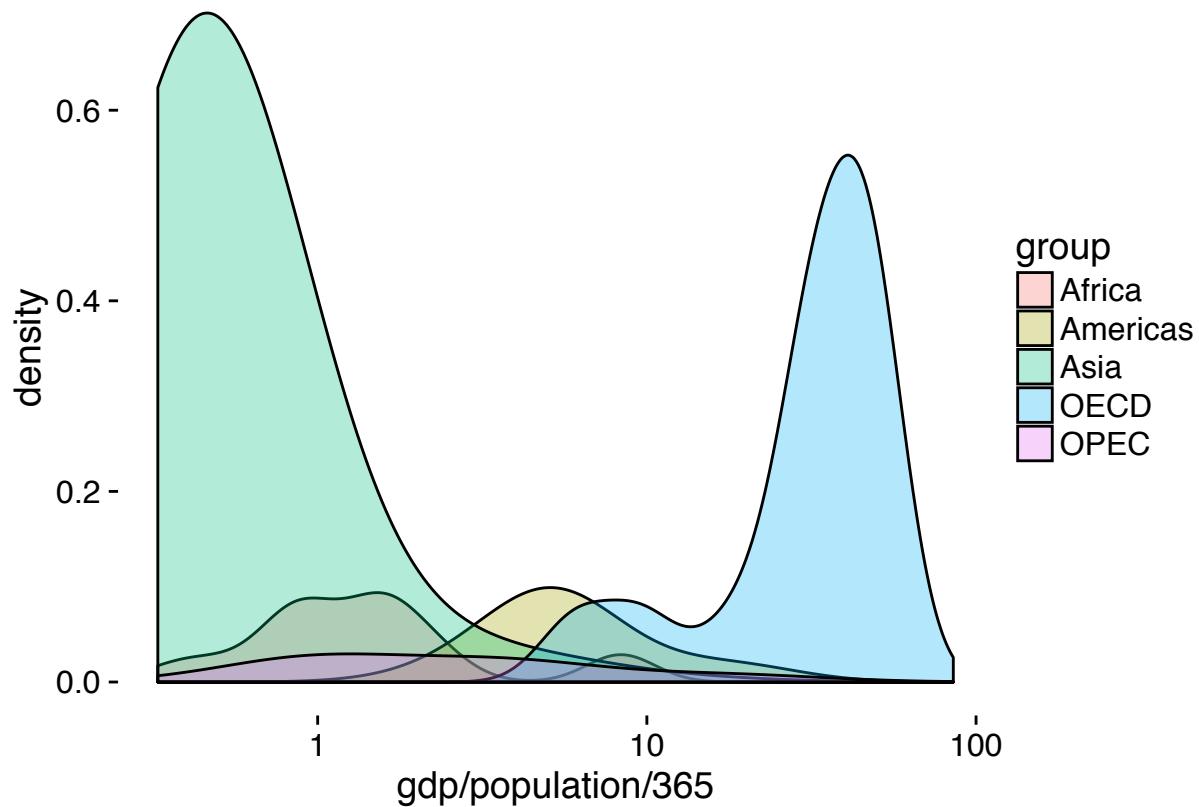


### Problem 5.2

Now do the same but show each of the five groups separately.

```
dat2 <- mutate(dat, group=group) %>%
  filter(!is.na(group) & !is.na(population) & !is.na(gdp))

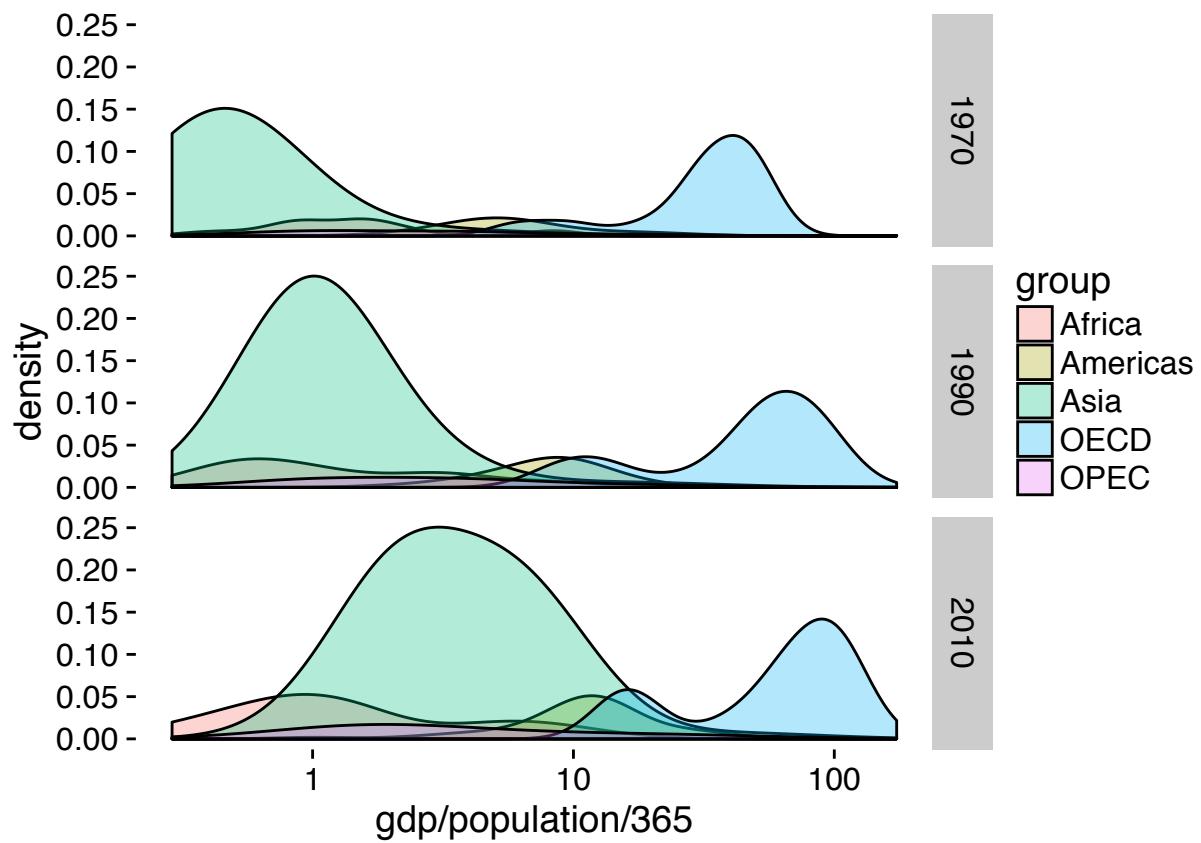
filter(dat2, year==1970 & group%in%c("OECD", "OPEC", "Asia", "Africa", "Americas")) %>%
  ggplot(aes(x = gdp/population/365)) +
  geom_density(aes(weight=population/sum(population), fill=group), alpha=0.3) + scale_x_log10()
```



### Problem 5.3

Visualize these densities for several years. Show a couple of them. Summarize how the distribution has changed through the years.

```
ggplot(filter(dat2, year%in%c(1970,1990,2010) & group%in%c("OECD", "OPEC", "Asia", "Africa", "Americas")),  
       aes(x = gdp/population/365)) +  
  geom_density(aes(weight=population/sum(population), fill=group), alpha=0.3) + scale_x_log10() +  
  facet_grid(year~.)
```



# Homework 2: The Big Short

This homework is due Sunday February 26, 2016 at 5:00 PM. When complete, submit your code in the R Markdown file and the knitted HTML via GitHub.

## Background

This homework is motivated by circumstances surrounding the [financial crisis of 2007-2008](#). We title the homework *The Big Short* because this is the title of a recent book about this topic that was recently made into a movie.

Part of what caused the financial crisis was that the risk of certain [securities](#) sold by financial institutions were underestimated. Specifically, the risk of mortgage-backed securities (MBS) and collateralized debt obligations (CDO), whose price dependents on homeowners making their monthly payments, were grossly underestimated. A combination of factors resulted in many more defaults than were expected. This resulted in a crash of the prices of these securities. As a consequence, banks lost so much money that they needed bailouts to avoid default.

Here we present a **very** simplified version of what happened with some of these securities. Hopefully it will help you understand how a wrong assumption about statistical behavior of events can lead to substantial differences between what the model predicts and what actually happens. Specifically, we will see how using an independence assumption can result in misleading conclusions. Before we start with the specific application we ask you about a simple casino game.

## Problem 1

In the game of [roulette](#) you can bet on several things including black or red. On this bet, if you win, you double your earnings. How does the casino make money on this then? If you look at the [possibilities](#) you realize that the chance of red or black are both slightly less than 1/2. There are two green spots, so the of landing on black (or red) is actually 18/38, or 9/19.

### Problem 1A

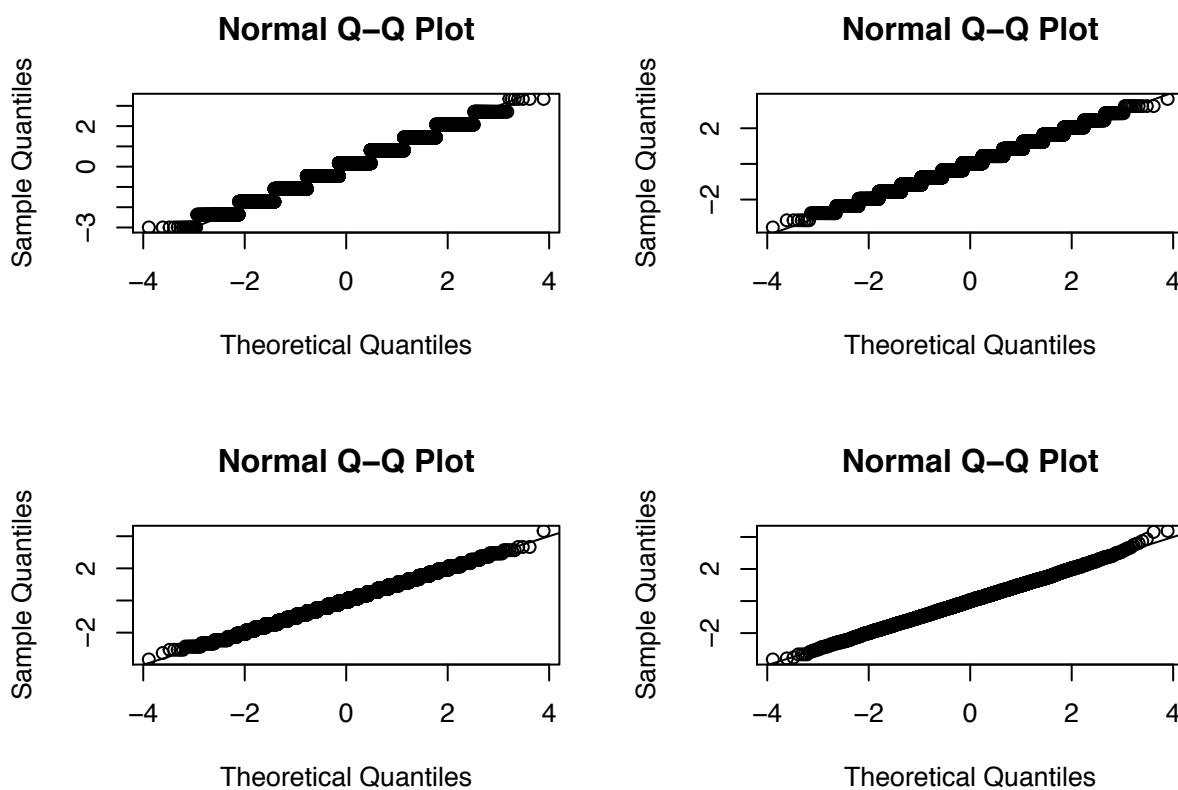
Let's make a quick sampling model for this simple version of roulette. You are going to bet a dollar each time you play and always bet on black. Make a box model for this process using the [sample](#) function. Write a function [get\\_outcome](#) that takes as an argument the number of times you play  $N$  and returns your earnings  $S_N$ .

```
get_outcome <- function(N){  
  x <- sample( c(-1,1), N, replace=TRUE, prob=c(10/19,9/19))  
  sum(x)  
}  
get_outcome(10)  
  
## [1] 8
```

## Problem 1B

Use Monte Carlo simulation to study the distribution of total earnings  $S_N$  for  $N = 10, 25, 100, 1000$ . That is, study the distribution of earnings for different number of plays. What are the distributions of these two random variables? How does the expected values and standard errors change with  $N$ . Then do the same thing for the average winnings  $S_N/N$ . What result that you learned in class predicts this?

```
mysd <- function(x){
  sqrt(mean((x-mean(x))^2))
}
B <- 10^4
par(mfrow=c(2,2)) ##don't worry if you don't know what this is. You can remove it.
for(N in c(10, 25, 100, 1000)){
  winnings <- replicate(B,get_outcome(N) )
  z <- (winnings - mean(winnings))/mysd(winnings)
  qqnorm(z)
  abline(0,1)
  cat("Expected value:", mean(winnings),
      "Standard Error:", mysd(winnings), "\n")
}
## Expected value: -0.5434 Standard Error: 3.155363
## Expected value: -1.207 Standard Error: 4.994552
## Expected value: -5.3158 Standard Error: 10.00461
```



```

## Expected value: -52.6264 Standard Error: 31.66596

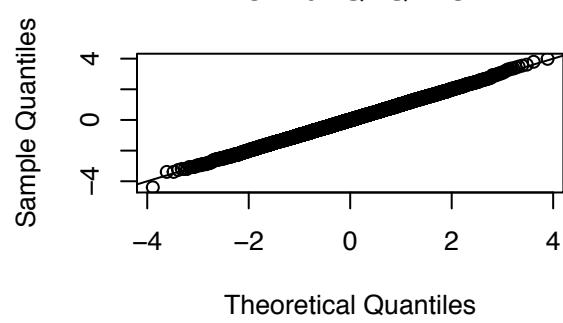
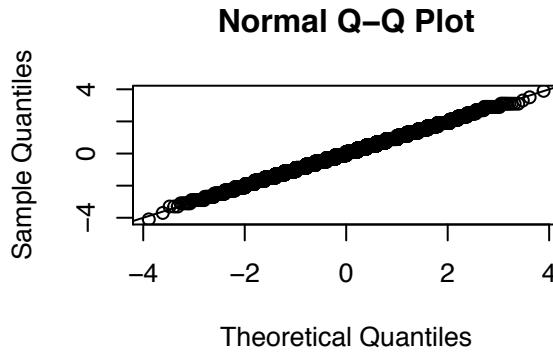
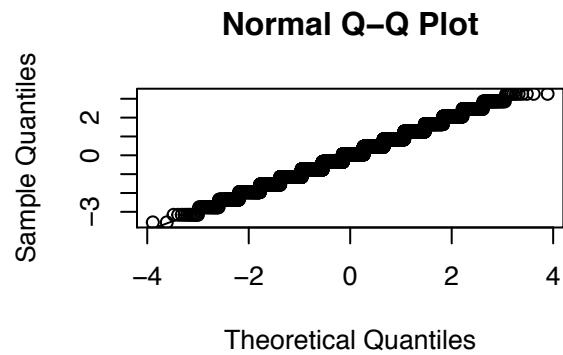
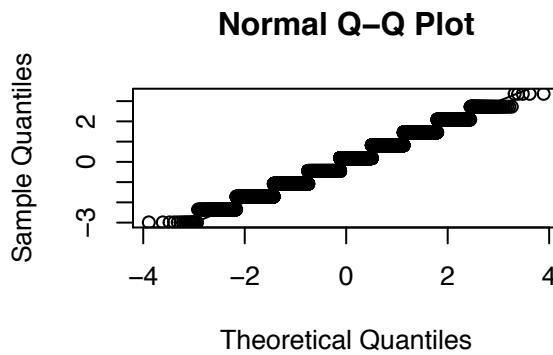
par(mfrow=c(2,2)) ##don't worry if you don't know what this is. You can remove it.
## Try out a few N
for(N in c(10, 25, 100, 1000)){
  ## copy code from above:
  average_winnings <- replicate(B, get_outcome(N) ) / N
  z <- (average_winnings - mean(average_winnings))/mysd(average_winnings)
  qqnorm(z)
  abline(0,1)
  cat("Expected value:", mean(average_winnings),
      "Standard Error:", mysd(average_winnings), "\n")
}

```

```
## Expected value: -0.05858 Standard Error: 0.3151323
```

```
## Expected value: -0.05032 Standard Error: 0.199461
```

```
## Expected value: -0.050258 Standard Error: 0.09996406
```



```
## Expected value: -0.052467 Standard Error: 0.03173256
```

Answer: These are predicted with the central limit theorem that tells us  $S_N$  is approximately normal with expected value  $\mu N$  and standard error  $\sigma\sqrt{N}$ . The average is approximated by the normal as well with expected value  $\mu$  and standard error  $\sigma/\sqrt{N}$ . The SE average gets smaller and smaller with  $N$ .

## Problem 1C

What is the expected value of our sampling model? What is the standard deviation of our sampling model?

Answer: Expectation is  $\mu = -1 \times (1-p) + 1 \times p$  which is  $-1/19$ . The casino makes, on average, about 5 cents on each bet. In general you can envision this as a box model with an infinite number of tickets (since you can imagine playing this game an infinite number of times) with a fraction  $p = 18/38$  tickets labeled 1 and  $1-p = 20/38$  tickets labeled  $-1$ .

Standard deviation is  $\sigma = |1 - -1| \sqrt{(9/19)(10/19)}$  which is 0.998614.

## Problem 1D

If you play 25 times. Use CLT to approximate the probability that the casino loses money. Then use a Monte Carlo simulation to corroborate.

You sum  $S$  is approximately normal with mean  $\mu \times N$  and standard error  $\sqrt{N}\sigma$  with  $N = 25$

$$\Pr(S > 0) = \Pr\left(\frac{S - \mu N}{\sigma \sqrt{N}} > \frac{-\mu N}{\sigma \sqrt{N}}\right)$$
$$1 - \Phi^{-1}\left(\sqrt{N} \frac{-\mu}{\sigma}\right)$$

```
1-pnorm( sqrt(25)*(1/19)/0.998614)
```

```
## [1] 0.3960737
```

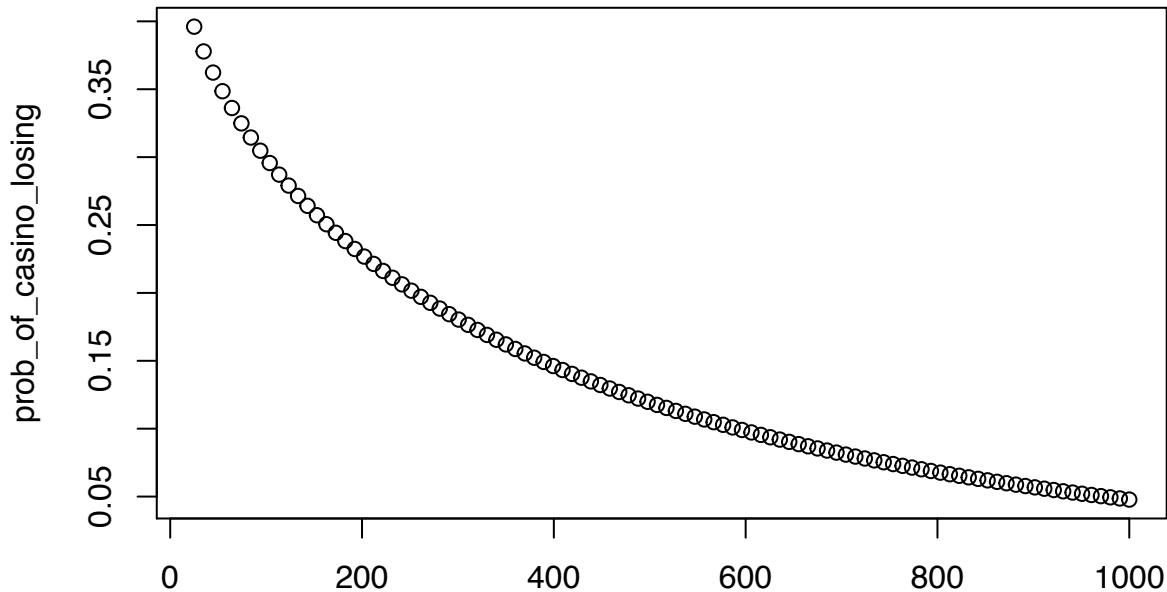
```
B <- 10^5
winnings <- replicate(B, get_outcome(25) )
mean(winnings>0)
```

```
## [1] 0.39245
```

## Problem 1E

In general, what is the probability that the casino loses money as a function of  $N$ ? Make a plot for values ranging from 25 to 1,000. Why does the casino give you free drinks if you keep playing?

```
Ns <- seq(25, 1000, len=100)
prob_of_casino_losing <- 1 - pnorm( sqrt(Ns)*(1/19/0.998614))
plot(Ns, prob_of_casino_losing)
```



Answer: Some of you did simulations to answer this question, but I quickly ran into computational issues with the large sample sizes. This is a perfect example of when an approximate answer is worth the error in approximation. This answer also probably gives you some idea of why casinos try to keep you at the table for long periods of time with free drinks!

## Problem 2

You run a bank that has a history of identifying potential homeowners that can be trusted to make payments. In fact, historically, in a given year, only 2% of your customers default. You want to use stochastic models to get an idea of what interest rates you should charge to guarantee a profit this upcoming year.

### Problem 2A

Your bank gives out 1,000 loans this year. Create a sampling model and use the function `sample` to simulate the number of foreclosure in a year with the information that 2% of customers default. Also suppose your bank loses \$120,000 on each foreclosure. Run the simulation for one year and report your loss.

```
N <- 1000
loss_per_foreclosure <- 120000
p <- 0.02
defaults <- sample( c(0,1), N, prob=c(1-p, p), replace = TRUE)
sum(defaults * loss_per_foreclosure)

## [1] 2640000
```

### Problem 2B

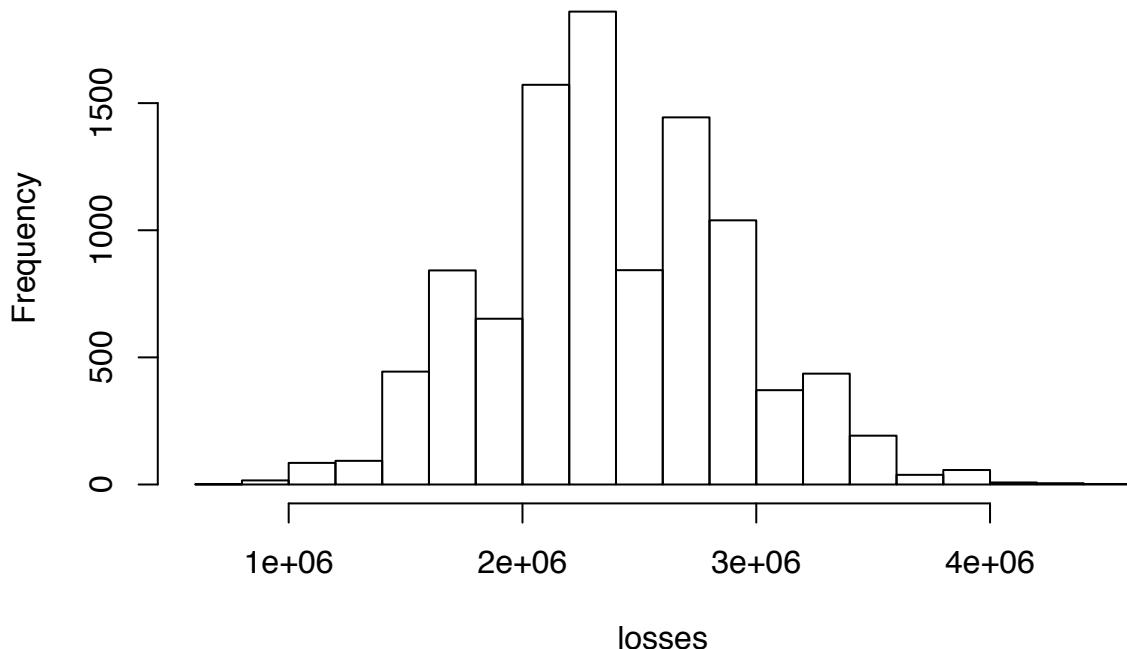
Note that the loss you will incur is a random variable. Use Monte Carlo simulation to estimate the distribution of this random variable. Use summaries and visualization to describe your potential losses to your board of trustees.

```

N <- 1000
loss_per_foreclosure <- 120000
p <- 0.02
B <- 10^4
## Use replicate to generate B random variables
losses <- replicate(B, {
  ## copy code from 2A
  defaults <- sample( c(0,1), N, prob=c(1-p, p), replace = TRUE)
  sum(defaults * loss_per_foreclosure)
})
## Data looks normally distributed:
hist(losses)

```

**Histogram of losses**



```

par(mfrow=c(1,2))
qqnorm(losses)
qqline(losses)
mean(losses)

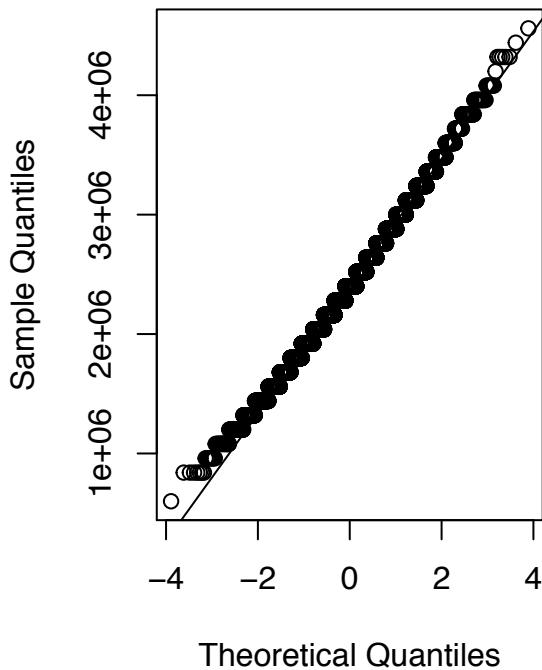
```

```
## [1] 2404752
```

```
sd(losses)
```

```
## [1] 525314.3
```

## Normal Q-Q Plot



### Problem 2C

The 1,000 loans you gave out were for \$180,000. The way your bank can give out loans and not lose money is by charging an interest rate. If you charge an interest rate of, say, 2% you would earn \$3,600 for each loan that doesn't foreclose. At what percentage should you set the interest rate so that your expected profit totals \$100,000. Hint: Create a sampling model with expected value 100 so that when multiplied by the 1,000 loans you get an expectation of \$100,000. Corroborate your answer with a Monte Carlo simulation.

Solution: The box model must have an expected value of 100 so that the sum of 1,000 outcomes has expected value \$100,000. We lose \$120,000 with a probability of  $p = 0.02$ . If we make  $x$  on each loan that doesn't foreclose, our expected value is  $0.98x - 0.02 \times 120000$ . For this to add up to 100 we need This  $x = (100 + 0.02 \times 120000)/0.98$ . Our interest rate  $r$  must be  $200000r = (100 + 0.02 \times 120000)/0.98$  or  $r = (100 + 0.02 \times 120000)/0.98/200000$ .

Check with Monte Carlo simulation

```
N <- 1000
loan <- 180000
p <- 0.02
loss_per_foreclosure <- 120000
interest_rate <- (100+p*loss_per_foreclosure)/(1-p)/loan
cat("interest rate is: ",interest_rate)

## interest rate is: 0.01417234

B <- 10000
profit <- replicate(B, {
  ## We add either a loss of -loss_per_foreclosure or gain of interest_rate*loan
```

```

## with probabilities p and 1-p respectively for each of N loans
x <- sample( c(-loss_per_foreclosure, interest_rate*loan), N, replace=TRUE, prob=c(p, 1-p))
## add gains and losses for total profit
sum(x)
})
mean(profit/N) ##should be about 100

## [1] 106.7158

```

Note: we'll try to ask you to verify things with a simulation but you'll start noticing that it's generally a good idea to do this to catch mistakes since simulations are quick and easy to write up.

## Problem 2D

In problem 2C, you were able to set a very low interest rate. Your customers will be very happy and you are expected to earn \$100,000 in profits. However, that is just an expectation. Our profit is a random variable. If instead of a profit your bank loses money, your bank defaults. Under the conditions of Problem 2C, what is the probability that your profit is less than 0?

```

## Run scripts above then:
mean(profit < 0)

```

```
## [1] 0.4365
```

## Problem 2E

Note that the probability of losing money is quite high. To what value would you have to raise interest rates in order to make the probability of losing money, and your bank and your job, as low as 0.001? What is the expected profit with this interest rate? Corroborate your answer with a Monte Carlo simulation.

Hint: Use the following short cut. If  $p$  fraction of a box are  $as$  and  $(1 - p)$  are  $bs$  then , then the SD of the list if  $|a - b| \sqrt{p(1 - p)}$

Solution: Let  $x = 180000r$  be the earnings we make per loan at an interest rate of  $r$  and  $l = 120000$  the loss resulting from a foreclosure. The *box* in our box model has an expected value of  $\mu = x(1 - p) - lp +$  with  $p$  the probability of default. The standard error is  $\sigma = (x + l)\sqrt{p(1 - p)}$ . We now that the sum  $S$  of the  $N = 1,000$  outcomes is approximately normal with expected value  $N\mu$  and standard error  $\sqrt{N}\sigma$ .

$$\Pr(S < 0) = 0.001$$

$$\Pr\left(\frac{S - N\mu}{\sigma\sqrt{N}} < \frac{-N\mu}{\sigma\sqrt{N}}\right) = 0.001$$

This random variable  $(S - N\mu)/(\sigma\sqrt{N})$  is approximately normal with expected value 0 and standard error 1. So this means that to get a probability of 0.001 we need:

$$\sqrt{N} \frac{\mu}{\sigma} = z \text{ with } z = -\Phi^{-1}(0.001)$$

$$\sqrt{N} \frac{x(1 - p) - lp}{(x + l)\sqrt{p(1 - p)}} = z$$

$$\sqrt{N}x(1-p) - \sqrt{N}lp = zx\sqrt{p(1-p)} + lz\sqrt{p(1-p)}$$

$$x \left\{ (1-p)\sqrt{N} - z\sqrt{p(1-p)} \right\} = l \left( z\sqrt{p(1-p)} + \sqrt{N}p \right)$$

$$x = \frac{l \left( z\sqrt{p(1-p)} + \sqrt{N}p \right)}{(1-p)\sqrt{N} - z\sqrt{p(1-p)}}$$

The interest rate is  $x/180000$ .

Confirm with a Monte Carlo simulation:

```
N <- 1000
loan <- 180000
p <- 0.02
z <- -qnorm(0.001)
loss_per_foreclosure <- 120000
interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
  ((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan
cat("Interest rate is: ", interest_rate, "\n")

## Interest rate is: 0.02323666

expectation <- -loss_per_foreclosure*p + interest_rate*loan*(1-p)
cat("Mean profit: ", expectation*N)

## Mean profit: 1698947

B <- 10^5
profit <- replicate(B, {
  x <- sample( c(-loss_per_foreclosure, interest_rate*loan), N, replace=TRUE, prob=c(p, 1-p))
  sum(x)
})
mean(profit<0)

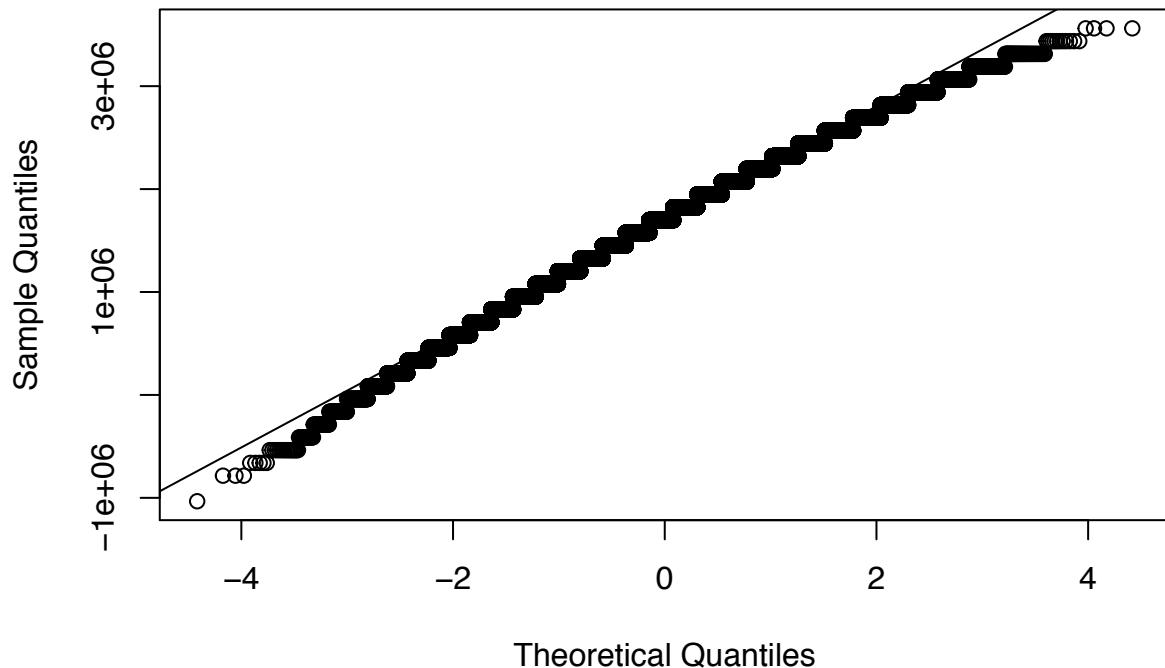
## [1] 0.00251
```

## Problem 2F

Note that the Monte Carlo simulation gave a slightly higher probability than 0.001. What is a possible reason for this? Hint: See if the disparity is smaller for larger values of  $p$ . Also check for probabilities larger than 0.001. Recall we made an assumptions when we calculated the interest rate.

```
## Run script from 2E
##Note that the normal approximation is not that great at the tails
qqnorm(profit)
qqline(profit)
```

## Normal Q-Q Plot

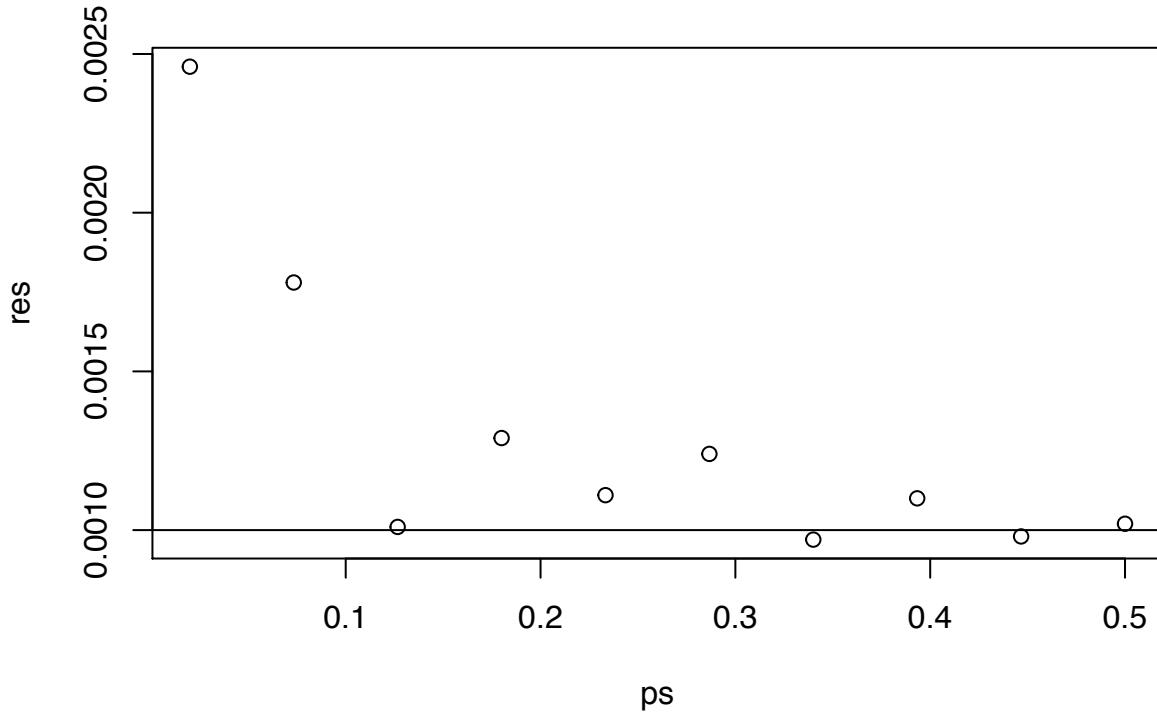


Also note that as the approximation is much better for value of  $p$  closer 0.5

```

N <- 1000
loan <- 180000
p <- 0.02
loss_per_foreclosure <- 120000
z <- -qnorm(0.001)
B <- 10^5
ps <- seq(0.02,0.5,len=10)
res <- sapply(ps,function(p){
  ##same code as above but changing p and therefore the interest_rate:
  interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
    ((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan
  profit <- replicate(B, {
    x <- sample( c(-loss_per_foreclosure, interest_rate*loan), N, replace=TRUE, prob=c(p, 1-p))
    sum(x)
  })
  mean(profit<0) ##should be about 0.001
})
plot(ps,res)
abline(h=0.001)

```



When  $p$  is close to 0 or 1, the central limit theorem needs larger  $N$  for the approximation to work.

### Problem 3

We were able to set interest rate of about 2% that guaranteed a very low probability of having a loss. Furthermore, the expected average was over \$1 million. Now other financial companies noticed the success of our business. They also noted that if we increase the number loans we give, our profits increase. However, the pool of reliable borrowers was limited. So these other companies decided to give loans to less reliable borrowers but at a higher rate.

### Problem 3A

The pool of borrowers they found had a much higher default rate, estimated to be  $p = 0.05$ . What interest rate would give these companies the same expected profit as your bank (Answer to 2E)?

```
##From 2E
N <- 1000
loan <- 180000
loss_per_foreclosure <- 120000
p <- 0.02
z <- -qnorm(0.001)
interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
  ((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan

## expectation per loan is therefore:
expectation <- -loss_per_foreclosure*p + interest_rate*loan*(1-p)

p2 <- 0.05
interest_rate2 <- (expectation +
```

```

loss_per_foreclosure*p2)/(1-p2)/loan
cat("New interest rate is:", interest_rate2)

```

```
## New interest rate is: 0.04502308
```

## Problem 3B

At the interest rate calculated in 3A what is the probability of negative profits? Use both the normal approximation and then confirm with a Monte Carlo simulation.

Answer: Normal approximation tells us the average profit  $S/N$  has expected value  $\mu = \text{expectation}$  and standard deviation  $\sigma/\sqrt{N}$  with  $\sigma = (\text{interest\_rate2} * \text{loan} + \text{loss\_per\_foreclosure}) * \sqrt{p2 * (1-p2)}$ . So we compute

$$\Pr(S/N > 0) = \Pr\left(\sqrt{N}(S/N - \mu)/\sigma > -\sqrt{N}\mu/\sigma\right) = \Pr\left(Z > -\sqrt{N}\mu/\sigma\right)$$

```

## From 3A we have the expectation is
N <- 1000
loan <- 180000
loss_per_foreclosure <- 120000
p <- 0.02
z <- -qnorm(0.001)
interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
  ((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan

expectation <- -loss_per_foreclosure*p + interest_rate*loan*(1-p)

p2 <- 0.05
interest_rate2 <- (expectation +
  loss_per_foreclosure*p2)/(1-p2)/loan

## to compute normal approximation:
mu = expectation
sigma = (interest_rate2*loan + loss_per_foreclosure)*sqrt(p2*(1-p2))
pnorm(-sqrt(N)*mu/sigma)

```

```
## [1] 0.0271593
```

```

## Now confirm with Monte Carlo
B <- 10^4
profit <- replicate(B, {
  x <- sample( c(-loss_per_foreclosure, interest_rate2*loan), N, replace=TRUE, prob=c(p2, 1-p2))
  sum(x)
})
mean(profit<0)

```

```
## [1] 0.0278
```

### Problem 3C

Note that the probability is much higher now. This is because the standard deviation grew. The companies giving out the loans did not want to raise interest rates much more since it would drive away clients. Instead they used a statistical approach. They increased  $N$ . How large does  $N$  need to be for this probability to be 0.001? Use the central limit approximation and then confirm with a Monte Carlo simulation.

Answer: From before

$$\sqrt{N} \frac{\mu}{\sigma} = z \text{ with } z = -\Phi^{-1}(0.001)$$

which implies that with the new default probability  $p_2$  we have

$$\sqrt{N} = z \frac{(x + l) \sqrt{p_2(1 - p_2)}}{x(1 - p_2) - lp_2}$$

```
## From 3B we have the new interest rate is computed like this
N <- 1000
loan <- 180000
loss_per_foreclosure <- 120000
p <- 0.02
z <- -qnorm(0.001)
interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
  ((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan

expectation <- -loss_per_foreclosure*p + interest_rate*loan*(1-p)

p2 <- 0.05
interest_rate2 <- (expectation +
  loss_per_foreclosure*p2)/(1-p2)/loan

## Now let's compute the new N
z <- -qnorm(0.001)
N2 <- (z*(interest_rate2*loan + loss_per_foreclosure)*sqrt(p2*(1-p2)) /
  (interest_rate2*loan*(1-p2) - loss_per_foreclosure*p2 ))^2
cat("N is : ", N2)

## N is : 2578.947

### And confirm that the probability goes down to close to 0.001
B <- 10^4
profit <- replicate(B, {
  x <- sample( c(-loss_per_foreclosure, interest_rate2*loan), N2, replace=TRUE, prob=c(p2, 1-p2))
  sum(x)
})
mean(profit<0)

## [1] 0.0011
```

So by increasing the number of loans we were able to reduce our risk! Now, for this to work, all the assumptions in our model need to be approximately correct, including the assumption that the probability of default was **independent**. This turned out to be false and the main reason for the under estimate of risk.

## Problem 3D

Define the following matrix of outcomes for two borrowers using our previous box model:

```
loan <- 180000
loss_per_foreclosure <- 120000
p2 <- 0.05
interest_rate2 <- 0.05
B <- 10^5
outcomes1 <- replicate(B,{
  sample( c(-loss_per_foreclosure, interest_rate2*loan) , 2, replace=TRUE, prob=c(p2, 1-p2))
})
```

We can confirm independence by computing the probability of default for the second conditioned on the first defaulting:

```
sum( outcomes1[1,] < 0 & outcomes1[2,]<0)/sum(outcomes1[1,]<0)

## [1] 0.05108453
```

This quantity is about the same as the probability of default 0.05.

Now we create a new model. Before generating each set of defaults, we assume that a random event occurred that makes all default probabilities go up or go down by 4 points. We could see how this would happen if, for example, demand for houses decreases and all house prices drop.

```
B <- 10^5
outcomes2 <- replicate(B,{
  add <- sample( c(-0.04,0.04) , 1)
  sample( c(-loss_per_foreclosure, interest_rate2*loan) , 2, replace=TRUE, prob=c(p2+add, 1-(p2+add)))
})
```

Note that the outcomes are no longer independent as demonstrated by this result not being equal to 0.05

```
sum( outcomes2[1,] < 0 & outcomes2[2,]<0)/sum(outcomes2[1,]<0)

## [1] 0.083183
```

Generate a simulation with correlated outcomes such as those above. This time use the interest rate calculated in 3A. What is the expected earnings under this model compared to the previous?

```
### Compute the interest rate from 3A
N <- 1000
loan <- 180000
loss_per_foreclosure <- 120000
p <- 0.02
z <- -qnorm(0.001)
interest_rate <- loss_per_foreclosure*(z*sqrt(p*(1-p))+sqrt(N)*p)/
((1-p)*sqrt(N) - z*sqrt(p*(1-p)))/loan

## expectation per loan is therefore:
expectation <- -loss_per_foreclosure*p + interest_rate*loan*(1-p)
```

```

p2 <- 0.05
interest_rate2 <- (expectation +
                     loss_per_foreclosure*p2)/(1-p2)/loan
## And the N we need to get the same low probability
z <- -qnorm(0.001)
N2 <- (z*(interest_rate2*loan + loss_per_foreclosure)*sqrt(p2*(1-p2)) /
        (interest_rate2*loan*(1-p2) - loss_per_foreclosure*p2 ))^2

## Now run the simulation using the code given above for correlated data
B <- 10^5
profit2 <- replicate(B,{
  add <- sample( c(-0.04,0.04), 1)
  x <- sample( c(-loss_per_foreclosure, interest_rate2*loan ), N2, replace=TRUE, prob=c(p2+add, 1-(p2+add)))
  sum(x)
})

```

Answer:

Note that expected profits are about the same:

```

## Without the added probabitiliy to induce correlation we get:
B <- 10^5
profit <- replicate(B,{
  add <- 0
  x <- sample( c(-loss_per_foreclosure, interest_rate2*loan ), N2, replace=TRUE, prob=c(p2+add, 1-(p2+add)))
  sum(x)
})

mean(profit)

## [1] 4384716

mean(profit2)

## [1] 4370193

```

What is the probability of losing \$1 million compared to the previous?

```

mean(profit < 0)

## [1] 0.00115

mean(profit2 < 0)

## [1] 0.50048

```

What is the probability of losing \$10 million compared to the previous?

```
mean(profit < -10^6)
```

```
## [1] 9e-05
```

```
mean(profit2 < -10^6)
```

```
## [1] 0.50048
```

## Problem 4

Read [this wikipedia page](#) about the financial crisis. Write a paragraph describing how what you learned in this homework help explain the conditions that led to the crisis.

Example answer: If the creators of the securities assumed independence when computing the probability but the reality was that events were not independent, then they grossly underestimated the chances of losing money. This resulted in them selling securities at too low a price since the risk was much higher than what they calculated. This led to billions of dollars of losses once defaults started happening at a much higher rate than expected.

Of course this was not the only reason for the underestimation of the risk, but we hope this gave you some idea of the types of analysis that could help explain things.