Section 001 CDS 101 *Introduction to Computational and Data Sciences*

Class 24
Normal distribution

November 20, 2017

Applying the normal distribution with R

SAT scores example revisited

- How can we compare the SAT score to the ACT score?

SAT scores example revisited

- How can we compare the SAT score to the ACT score?
- Compare the percentiles

SAT scores example revisited

- How can we compare the SAT score to the ACT score?
- Compare the percentiles
- Pam's SAT score was 1800 and Jim's ACT score was 24.

SAT scores example revisited

- How can we compare the SAT score to the ACT score?
- Compare the percentiles
- Pam's SAT score was 1800 and Jim's ACT score was 24.
- The SAT score distribution is normally distributed with a mean of 1500 and a standard deviation of 300.

SAT scores example revisited

- How can we compare the SAT score to the ACT score?
- Compare the percentiles
- Pam's SAT score was 1800 and Jim's ACT score was 24.
- The SAT score distribution is normally distributed with a mean of 1500 and a standard deviation of 300.
- The ACT score distribution is normally distributed with a mean of 21 and a standard deviation of 5.

SAT scores example revisited

- How can we compare the SAT score to the ACT score?
- Compare the percentiles
- Pam's SAT score was 1800 and Jim's ACT score was 24.
- The SAT score distribution is normally distributed with a mean of 1500 and a standard deviation of 300.
- The ACT score distribution is normally distributed with a mean of 21 and a standard deviation of 5.

We can use `pnorm( )` to compute the percentile.

SAT scores example revisited

Pam's percentile is:

SAT scores example revisited

Pam's percentile is:

```r
pam_score <- 1800
pam_percentile <- pnorm(q = pam_score, mean = 1500, sd = 300)
```

SAT scores example revisited

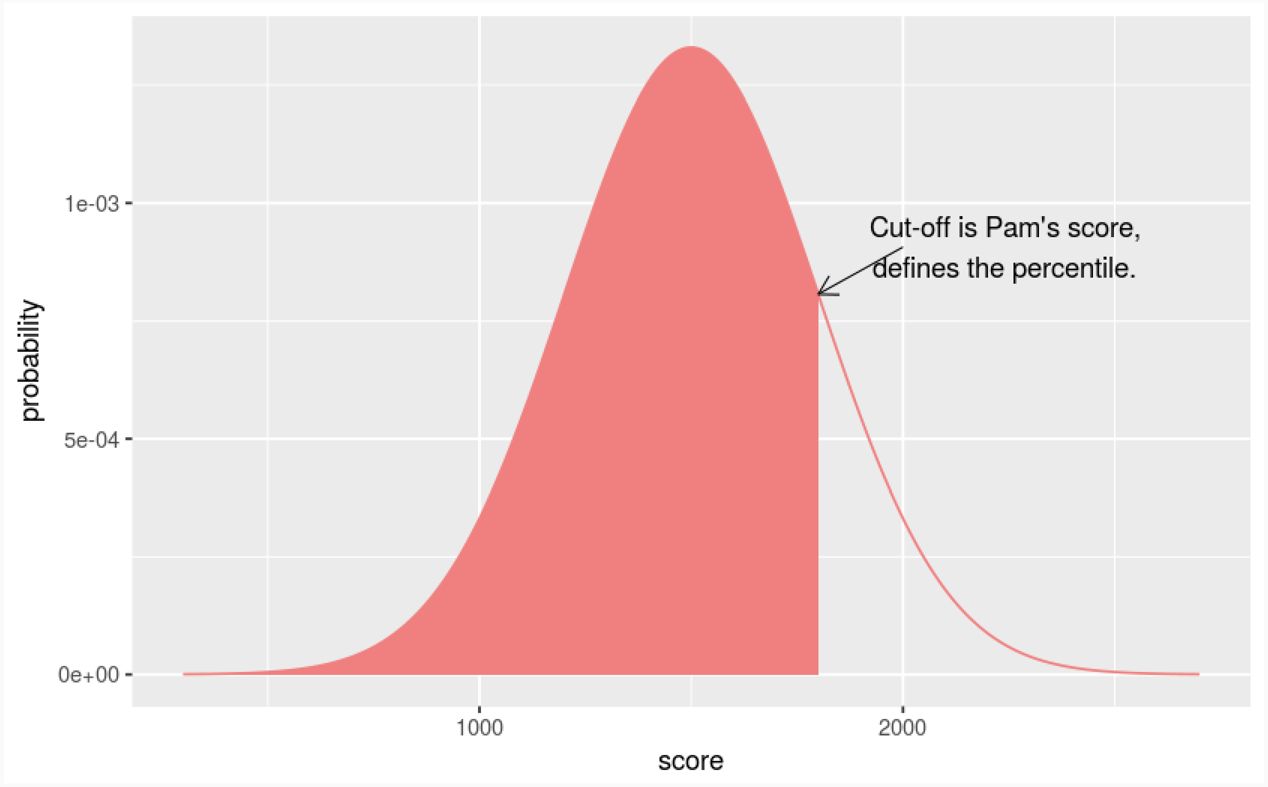
Pam's percentile is:

```r
pam_score <- 1800
pam_percentile <- pnorm(q = pam_score, mean = 1500, sd = 300)
```

```
## [1] 0.8413447
```

SAT scores example revisited

Visually, this corresponds to the following area under the normal distribution:



SAT scores example revisited

Jim's percentile is:

SAT scores example revisited

Jim's percentile is:

```r
jim_score <- 24
jim_percentile <- pnorm(q = jim_score, mean = 21, sd = 5)
```

SAT scores example revisited

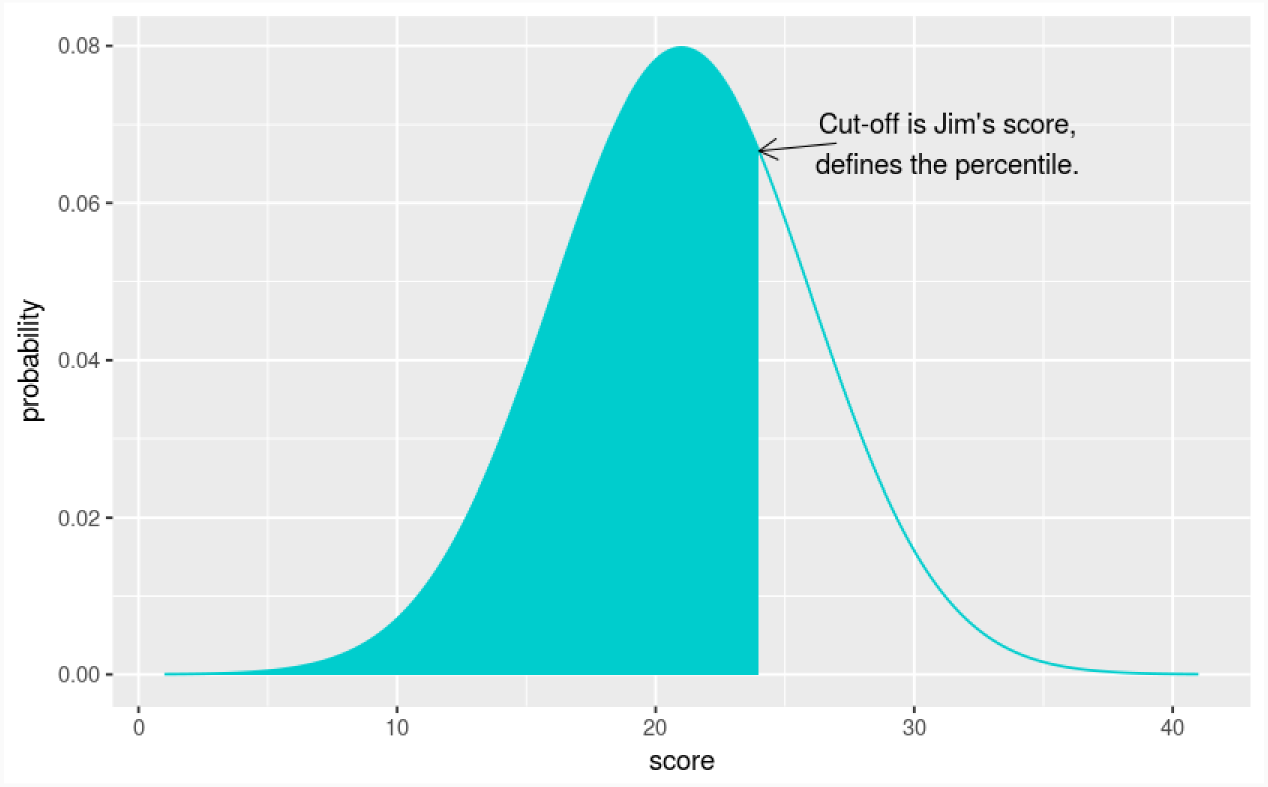
Jim's percentile is:

```r
jim_score <- 24
jim_percentile <- pnorm(q = jim_score, mean = 21, sd = 5)
```

```
## [1] 0.7257469
```

SAT scores example revisited

Visually, this corresponds to the following area under the normal distribution:



Cumulative distribution function of normal distribution

- Using percentiles is a useful way to compare distributions to see if they're the same

Cumulative distribution function of normal distribution

- Using percentiles is a useful way to compare distributions to see if they're the same
- Instead of calculating percentiles one by one, use the cumulative distribution function (CDF)

Cumulative distribution function of normal distribution

- Using percentiles is a useful way to compare distributions to see if they're the same
- Instead of calculating percentiles one by one, use the cumulative distribution function (CDF)
- Maps the percentile values to the corresponding values in the data set.

Cumulative distribution function of normal distribution

- Using percentiles is a useful way to compare distributions to see if they're the same
- Instead of calculating percentiles one by one, use the cumulative distribution function (CDF)
- Maps the percentile values to the corresponding values in the data set.
- The CDF for the normal distribution model (not for imported datasets) can be accessed with `qnorm( )`

Cumulative distribution function of normal distribution

The CDF for the SAT scores is generated in the following way:

Cumulative distribution function of normal distribution

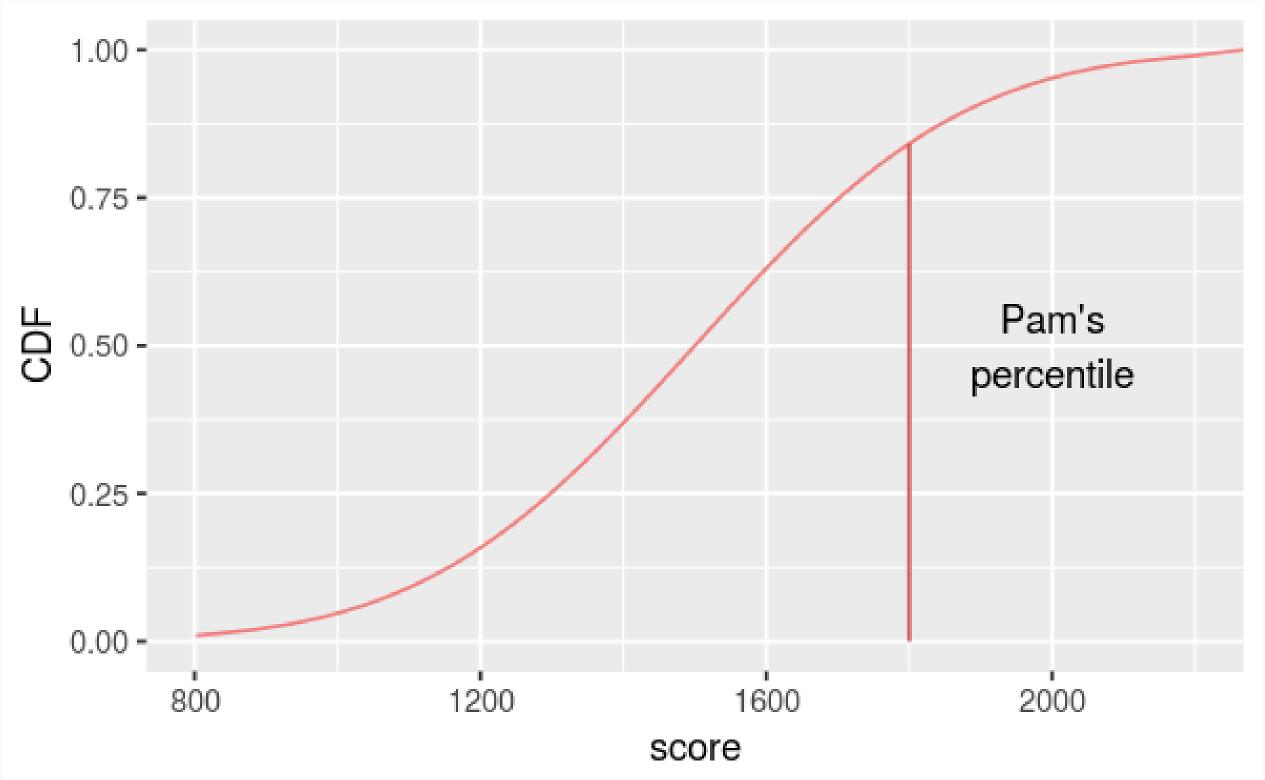
The CDF for the SAT scores is generated in the following way:

```r
sat_score_percentiles <- seq(0.01, 1, 0.01)
sat_score_cdf <- tibble(
  CDF = sat_score_percentiles,
  score = qnorm(p = sat_score_percentiles, mean = 1500, sd = 300))

ggplot(sat_score_cdf) + geom_line(mapping = aes(x = score, y = CDF))
```

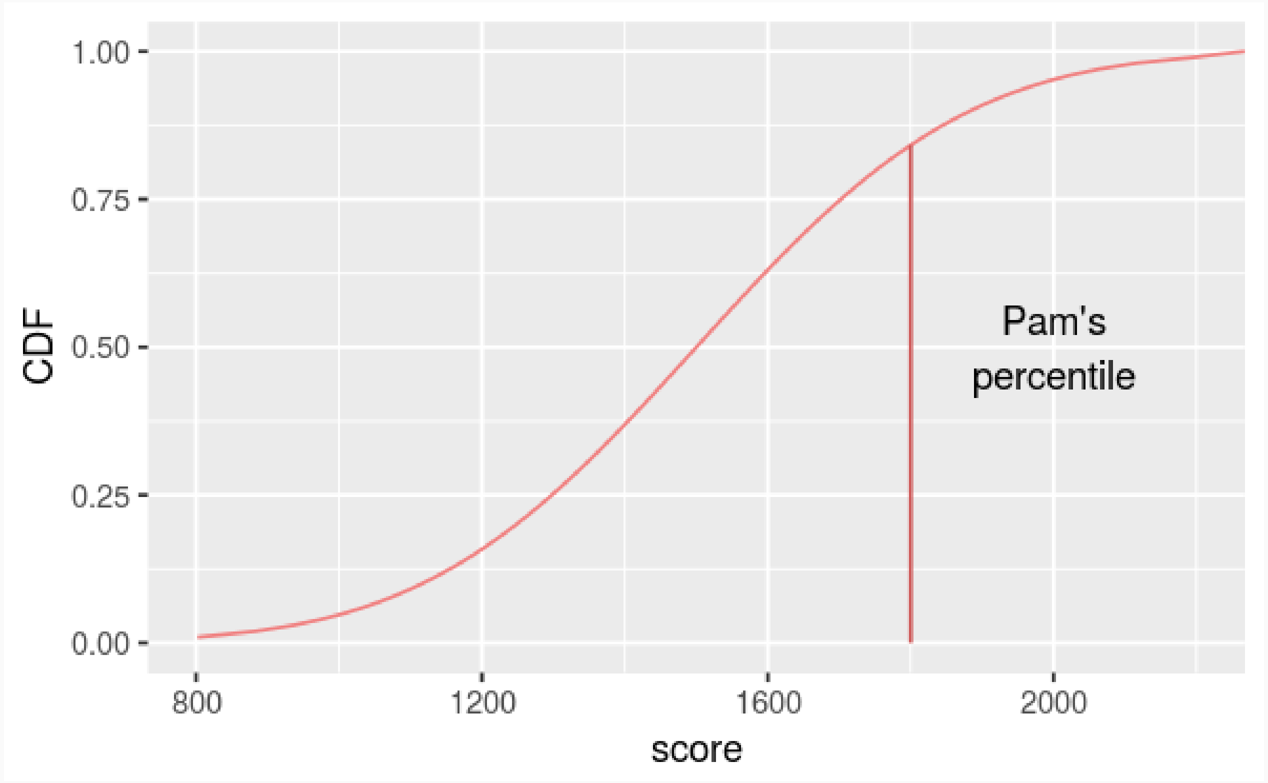
Cumulative distribution function of normal distribution

The CDF for the SAT scores is generated in the following way:



Cumulative distribution function of normal distribution

The CDF for the SAT scores is generated in the following way:



This shows how the CDF maps Pam's SAT score to a percentile.

Q-Q Plots

Load dataset on children's heights.

```
heights <- read_csv(file = "child_height_data.csv")
```

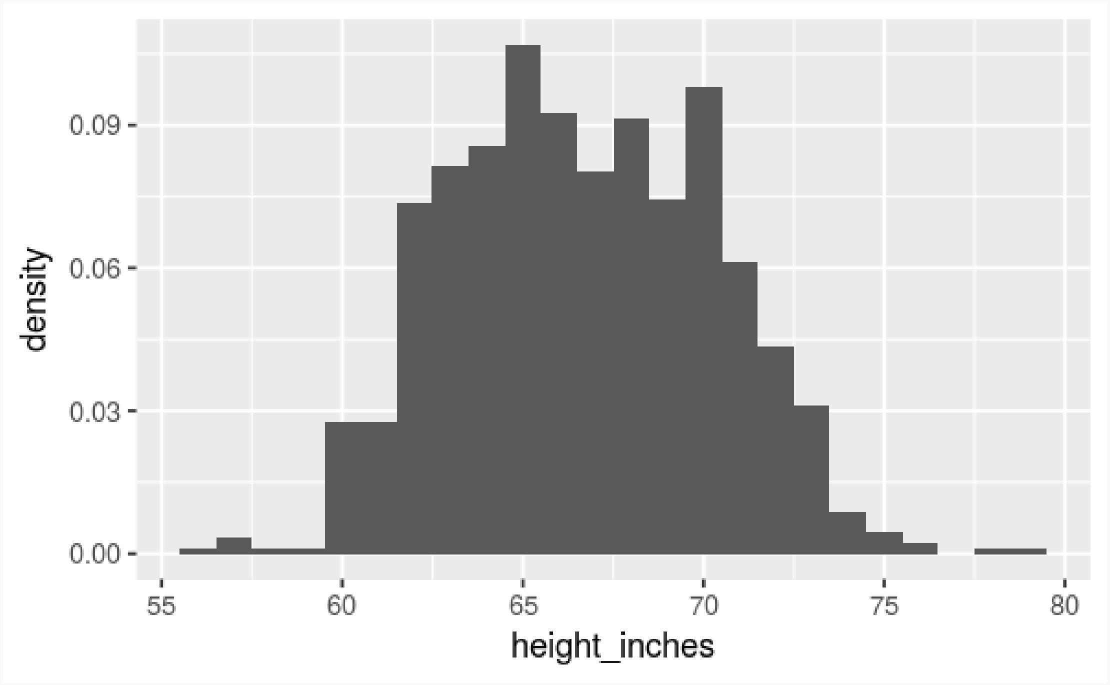
The first few lines in the dataset look like the following:

```
## # A tibble: 6 x 2
##     sex height_inches
##   <chr>         <dbl>
## 1     M          73.2
## 2     F          69.2
## 3     F          69.0
## 4     F          69.0
## 5     M          73.5
## 6     M          72.5
```

Q-Q Plots

Compute the PMF histogram:

```
ggplot(heights) +
  geom_histogram(mapping = aes(x = height_inches, y = ..density..),
                 binwidth = 1)
```



Q-Q Plots

First, let's compute theoretical line for ideal agreement:

- Find the 1st and 3rd quartiles

```r
qq_y <- quantile(heights$height_inches, c(0.25, 0.75))
```

Q-Q Plots

First, let's compute theoretical line for ideal agreement:

- Find the 1st and 3rd quartiles

```
qq_y <- quantile(heights$height_inches, c(0.25, 0.75))
```

- Find the matching normal values on the x-axis

```
qq_x <- qnorm(c(0.25, 0.75))
```

Q-Q Plots

First, let's compute theoretical line for ideal agreement:

- Find the 1st and 3rd quartiles

```
qq_y <- quantile(heights$height_inches, c(0.25, 0.75))
```

- Find the matching normal values on the x-axis

```
qq_x <- qnorm(c(0.25, 0.75))
```

- Compute line slope

```
qq_slope <- diff(qq_y) / diff(qq_x)
```

Q-Q Plots

First, let's compute theoretical line for ideal agreement:

- Find the 1st and 3rd quartiles

```r
qq_y <- quantile(heights$height_inches, c(0.25, 0.75))
```

- Find the matching normal values on the x-axis

```r
qq_x <- qnorm(c(0.25, 0.75))
```

- Compute line slope

```r
qq_slope <- diff(qq_y) / diff(qq_x)
```

- Compute line intercept

```r
qq_int <- qq_y[1] - qq_slope * qq_x[1]
```

Q-Q Plots

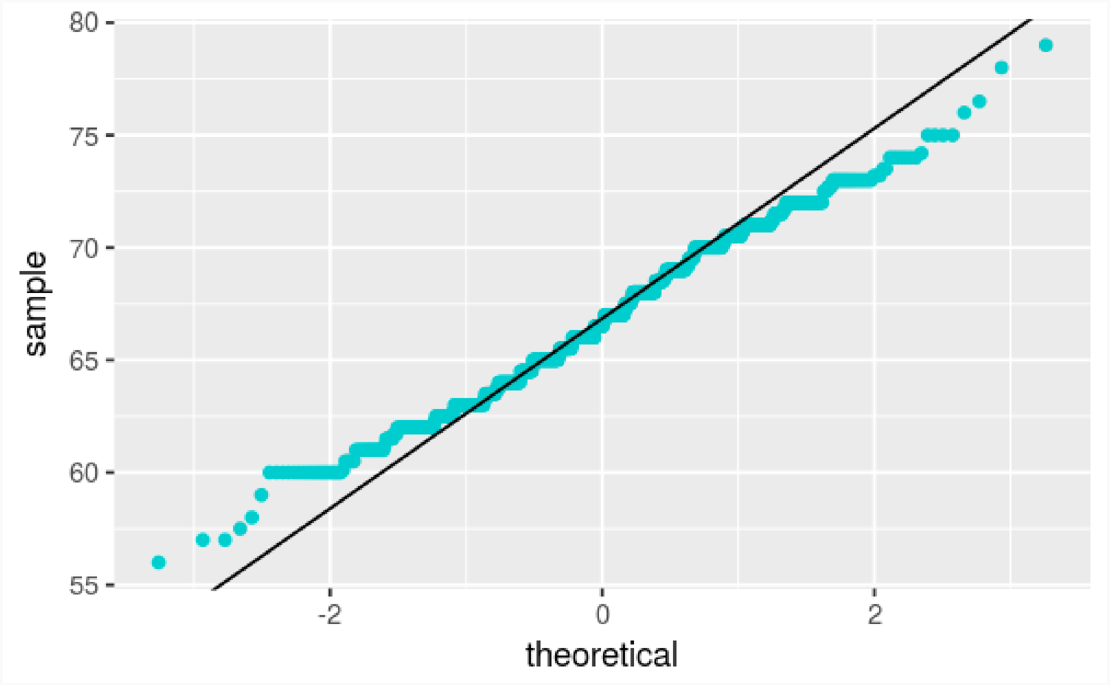
Now create the plot:

```r
ggplot(heights) +
  stat_qq(mapping = aes(sample = height_inches)) +
  geom_abline(intercept = qq_int, slope = qq_slope)
```

Q-Q Plots

Now create the plot:

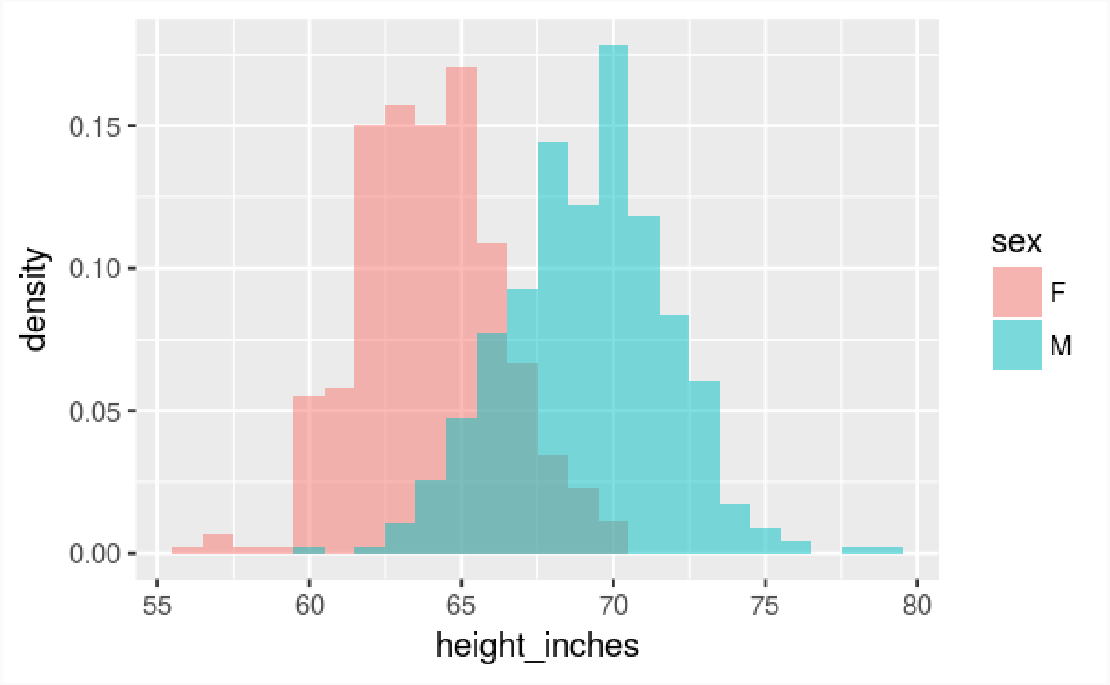
```
ggplot(heights) +
  stat_qq(mapping = aes(sample = height_inches)) +
  geom_abline(intercept = qq_int, slope = qq_slope)
```



Q-Q Plots

Check histograms for male and female separated.

```r
ggplot(heights) + geom_histogram(
  mapping = aes(x = height_inches, y = ..density.., fill = sex),
  binwidth = 1, position = "identity", alpha = 0.5)
```



Q-Q Plots

Re-run Q-Q Plot for male and female separated:

```r
# Male heights
heights_male <- filter(heights, sex == "M")
heights_female <- filter(heights, sex == "F")

# First, compute theoretical line for ideal agreement
# Find the 1st and 3rd quartiles
qq_y_male <- quantile(heights_male$height_inches,
                      c(0.25, 0.75))
qq_y_female <- quantile(heights_female$height_inches,
                        c(0.25, 0.75))

# Find the matching normal values on the x-axis
qq_x <- qnorm(c(0.25, 0.75))
```

Q-Q Plots

Re-run Q-Q Plot for male and female separated:

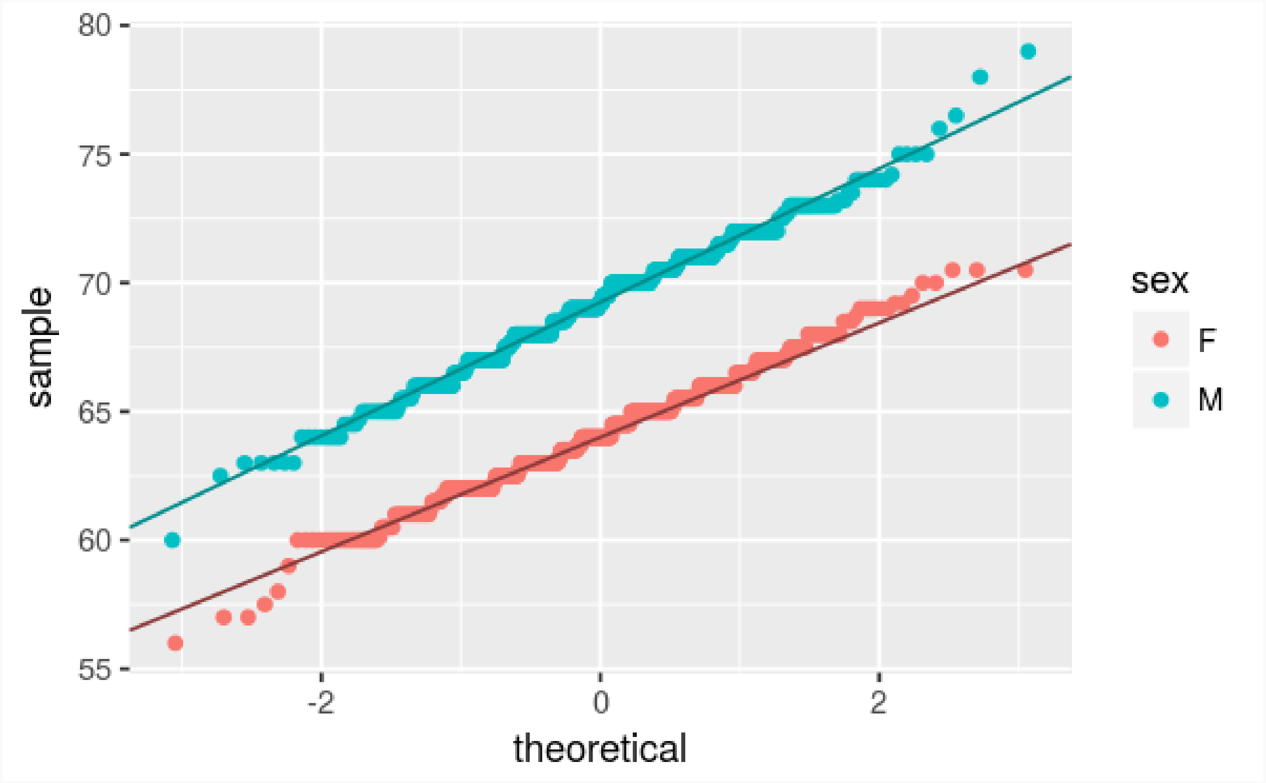
```r
# Compute line slope
qq_slope_male <- diff(qq_y_male) / diff(qq_x)
qq_slope_female <- diff(qq_y_female) / diff(qq_x)

# Compute line intercept
qq_int_male <- qq_y_male[1] - qq_slope_male * qq_x[1]
qq_int_female <- qq_y_female[1] - qq_slope_female * qq_x[1]

# Make the plot
ggplot(heights) +
  stat_qq(mapping = aes(sample = height_inches, color = sex)) +
  geom_abline(intercept = qq_int_male, slope = qq_slope_male) +
  geom_abline(intercept = qq_int_female, slope = qq_slope_female)
```

Q-Q Plots

Re-run Q-Q Plot for male and female separated:


sample
theoretical
sex
F
M