QR decomposition
The Gram-Schmidt algorithm takes a list of vectors \( \vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k \), forming a basis for a subspace \( L \), and returns a new list of vectors \( \vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k \), which is orthonormal and also forms a basis for \( L \).

For example, if

\[
\vec{v}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \vec{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

then the Gram-Schmidt algorithm returns

\[
\vec{u}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \vec{u}_2 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ -2 \end{bmatrix}.
\]

Here are the original vectors expressed in terms of the new vectors:

\[
\vec{v}_1 = \sqrt{2} \vec{u}_1 \quad \vec{v}_2 = \frac{1}{\sqrt{2}} \vec{u}_1 + \frac{\sqrt{6}}{2} \vec{u}_2.
\]

Notice that \( \vec{v}_j \) is a linear combination of \( \vec{u}_1, \vec{u}_2, \ldots, \vec{u}_j \).
\[ \vec{v}_1 = \sqrt{2} \vec{u}_1 \]
\[ \vec{v}_2 = \frac{1}{\sqrt{2}} \vec{u}_1 + \frac{\sqrt{6}}{2} \vec{u}_2 \]

We can write this using matrices:

\[
\begin{bmatrix}
\vec{v}_1 \\
\vec{v}_2
\end{bmatrix}
= 
\begin{bmatrix}
\vec{u}_1 \\
\vec{u}_2
\end{bmatrix}
\begin{bmatrix}
\sqrt{2} & \frac{1}{\sqrt{2}} \\
0 & \frac{\sqrt{6}}{2}
\end{bmatrix}.
\]

The first matrix, \( Q \), is called \( Q \). We note that \( Q \) has orthogonal columns; in other words, \( Q^T Q = \text{Id} \).

The second matrix, \( R \), is called \( R \). Note that it is upper triangular.
To recall what we talked about in the last lecture, the columns of $Q$ give an orthonormal basis for the image of $A$.

The orthogonal projection onto the image of $A$ is given by $QQ^T$. 
Some details

$QR$-decomposition is not unique, but it is very close to unique: The only freedom is choosing the signs of the $\vec{u}_1, \ldots, \vec{u}_k$. So

$$
\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
0 & -\frac{2}{\sqrt{6}}
\end{bmatrix}
\begin{bmatrix}
\sqrt{2} & 1/\sqrt{2} \\
0 & \sqrt{6}
\end{bmatrix}
= 
\begin{bmatrix}
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
0 & -\frac{2}{\sqrt{6}}
\end{bmatrix}
\begin{bmatrix}
-\sqrt{2} & -1/\sqrt{2} \\
0 & \sqrt{6}
\end{bmatrix}
= 
\begin{bmatrix}
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\
0 & 2/\sqrt{6}
\end{bmatrix}
\begin{bmatrix}
\sqrt{2} & 1/\sqrt{2} \\
0 & -\sqrt{6}
\end{bmatrix}
= 
\begin{bmatrix}
-\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\
0 & 2/\sqrt{6}
\end{bmatrix}
\begin{bmatrix}
-\sqrt{2} & -1/\sqrt{2} \\
0 & \sqrt{6}
\end{bmatrix}.
$$

are all $QR$-decompositions of $\begin{bmatrix}
1 & 0 \\
-1 & 1 \\
0 & -1
\end{bmatrix}$. 
Some details

According to some sources, the first matrix should be square (with orthogonal columns). So our example would look like

\[
\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\
0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}}
\end{bmatrix}
\begin{bmatrix}
\sqrt{2} & 1/\sqrt{2} \\
0 & \sqrt{6} \\
0 & 0
\end{bmatrix}
\]

instead of

\[
\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\
0 & -\frac{2}{\sqrt{6}}
\end{bmatrix}
\begin{bmatrix}
\sqrt{2} & 1/\sqrt{2} \\
0 & \sqrt{6}
\end{bmatrix}.
\]

If we write that first matrix as \([Q_1 \ Q_2]\), then the columns of \(Q_1\) are an orthonormal basis for \(\text{Image}(A)\) and the columns of \(Q_2\) are an orthonormal basis for \(\text{Image}(A)\)†.

Your textbook refuses to commit to a definition on this point, and only defines \(QR\)-decomposition for a square matrix.
Once you are done with this course, you will never want to compute Gram-Schmidt by hand.

- In Mathematica, `QRDecomposition[A]` will return the ordered pair `{QT, R}`.
- In MATLAB, `[Q,R] = qr(A,0)` will store $Q$ and $R$ in the variables $Q$ and $R$. If you call `[Q,R] = qr(A)` instead, you’ll get the version where $Q$ is a square matrix. If you just call `qr(A)`, you get the matrix $Q$.
- In NumPy, `numpy.linalg.qr(A, mode='reduced')` will return $(Q, R)$. If you call `numpy.linalg.qr(A, mode='complete')` instead, then you’ll get the version where $Q$ is a square matrix.