Additional Features of ggplot2

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School
w.evan.johnson@rutgers.edu

2025-02-16

Case Study: Describing Student Heights

Suppose that we have want to summarize to the heights of our classmates. We collect data on a set of individuals and save it in the `heights` data frame:

```
data(heights)
```

One way to convey the distribution of heights to simply provide list of 1050 heights. But there are much more effective ways to convey this information, and understanding the concept of a **distribution** will help.

Distributions

The most basic statistical summary of a list of objects or numbers is its distribution. The simplest way to think of a distribution is as a compact description of a list with many entries. With categorical data, the distribution simply describes the proportion of each unique category. The sex represented in the heights dataset is:

```
##
##    Female      Male
## 0.2266667 0.7733333
```

This **frequency table** is the simplest form of a distribution. We don't need to visualize it since one number describes everything we need to know: 23% are females and the rest are males.

Distributions

Here is another frequency table for the murders (region) data:

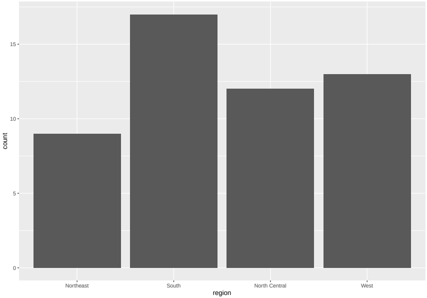
```
data(murders)
tab <- murders %>%
  count(region) %>%
  mutate(proportion = n/sum(n))
tab
```

```
##          region  n proportion
## 1     Northeast  9  0.1764706
## 2         South 17  0.3333333
## 3 North Central 12  0.2352941
## 4          West 13  0.2549020
```

Barplots

To generate a barplot to display the distribution of these data, we can use **barplot** with the `geom_bar` geometry. Here is the plot for the regions of the US:

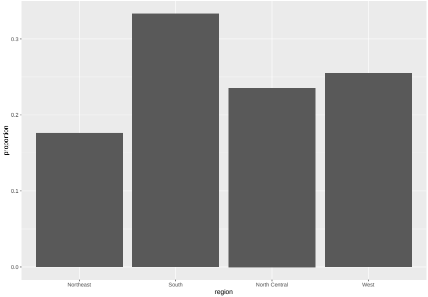
```
murders %>% ggplot(aes(region)) + geom_bar()
```



Barplots

We can also use the proportion variable for our barplot. For this we need to provide x (the categories) and y (the values) and use the stat="identity" option.

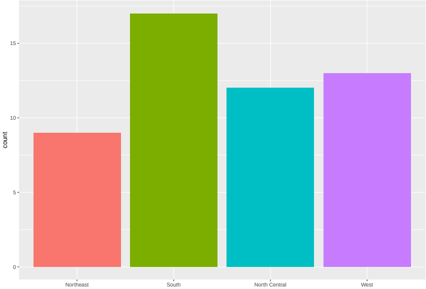
```r
tab %>% ggplot(aes(region, proportion)) +
  geom_bar(stat = "identity")
```



Barplots

We can also color the bars using the `fill` argument:

```
murders %>% ggplot(aes(region, fill=region)) +
  geom_bar(show.legend = FALSE) + xlab("")
```



Cumulative Distribution Functions

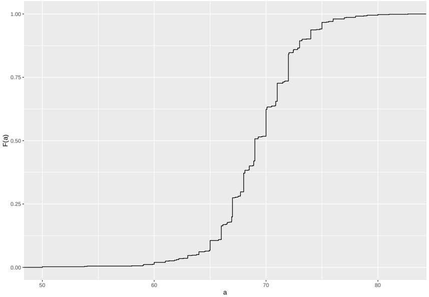
Statistics textbooks teach us that a more useful way to define a distribution for numeric data is to define a function that reports the proportion of the data below *a* for all possible values of *a*. This function is called the cumulative distribution function (CDF). In statistics, the following notation is used:

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

Cumulative Distribution Functions

Here is a plot of F for the male height data using `stat_ecdf`:

```
heights %>% filter(sex=="Male") %>%
  ggplot(aes(height)) +
  stat_ecdf() + ylab("F(a)") + xlab("a")
```



Cumulative Distribution Functions

Similar to what the frequency table does for categorical data, the CDF defines the distribution for numerical data.

From the plot, we can see that $F(66) = 0.1638$ of the values are below 65 or that $F(72) = 0.8411$ of the values are below 72, and so on. In fact, we can report the proportion of values between any two heights, say a and b, by computing $F(b) - F(a)$.

Cumulative Distribution Functions

This means that we have all the information needed to reconstruct the entire list. Paraphrasing the expression “a picture is worth a thousand words”, in this case, a picture is as informative as 812 numbers.

A final note: because CDFs can be defined mathematically the word **empirical** is added to make the distinction when data is used. We therefore use the term empirical CDF (eCDF).

Histograms

Histograms may be a more intuitive choice for these data. Histograms sacrifice just a bit of information to produce plots that are much easier to interpret.

A histogram divides the span of our data into non-overlapping bins of the same size. Then, for each bin, we count the number of values that fall in that interval. The histogram plots these counts as bars with the base of the bar defined by the intervals.

Histograms

To generate histograms we use the `geom_histogram` geometry. The only required argument is `x`, the variable for which we will construct a histogram. The code looks like this:

```r
heights %>% filter(sex == "Male") %>%
  ggplot(aes(height)) + geom_histogram()
```

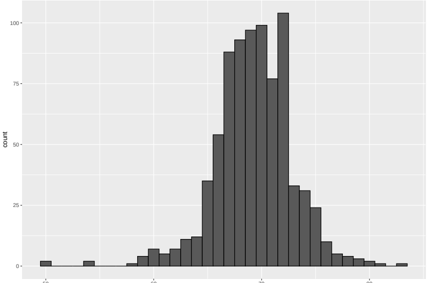
If we run the code above, it gives us a message:

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

Histograms

Here is the histogram for the height data splitting the range of values into one inch intervals: $(49.5, 50.5], \dots, (82.5, 83.5]$

```r
heights %>% filter(sex=="Male") %>%
  ggplot(aes(height)) +
  geom_histogram(binwidth = 1, color = "black")
```



Histograms

A histogram is similar to a barplot, but it differs in that the x-axis is numerical, not categorical.

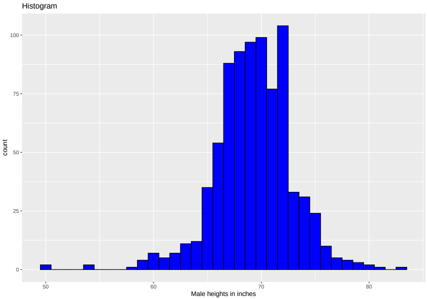
In this plot, we immediately learn some important properties about our data. First, the range of the data is from 50 to 84 with the majority (more than 95%) between 63 and 75 inches. Second, the heights are close to symmetric around 69 inches. Also, by adding up counts, we can obtain a very good approximation of the proportion of the data in any interval.

Therefore, the histogram above is not only easy to interpret, but also provides almost all the information contained in the raw list of 812 heights with a binwidth of 1.

Histograms

Finally, if for aesthetic reasons we want to add color, we use the arguments described in the help file. We also add labels and a title:

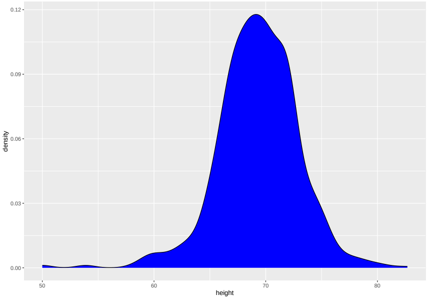
```
heights %>% filter(sex == "Male") %>% ggplot(aes(height)) +
  geom_histogram(binwidth = 1, fill = "blue", col = "black") +
  xlab("Male heights in inches") + ggtitle("Histogram")
```



Smoothed Density

Smooth density plots are aesthetically more appealing:

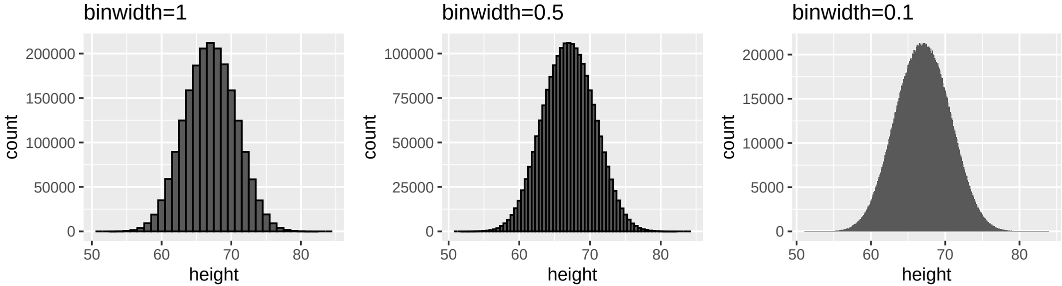
```
heights %>% filter(sex=="Male") %>%
  ggplot(aes(height)) + geom_density(fill= "blue")
```



Smoothed Density

In this plot, we no longer have sharp edges at the interval boundaries and many of the local peaks have been removed. Also, the scale of the y-axis changed from counts to **density**.

A density is like a **smoothed** histogram if you had, say, 1,000,000 values, measured very precisely. The smaller we make the bins, the smoother the histogram gets, for example:

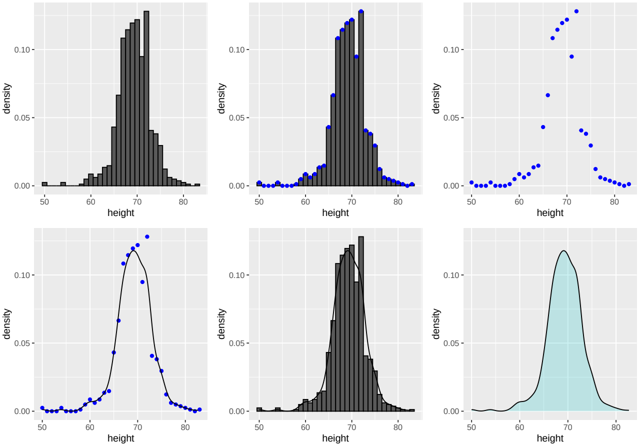


Smoothed Density

Now, back to reality. We don't have millions of measurements. Instead, we have 812 and we can't make a histogram with very small bins.

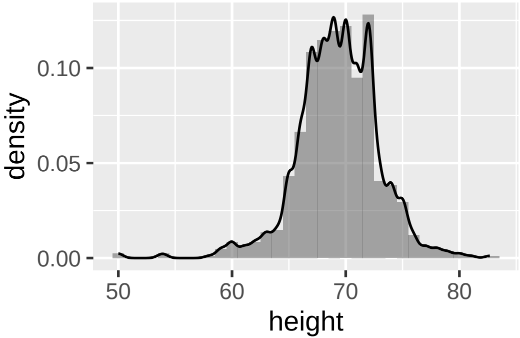
We therefore make a histogram, using bin sizes appropriate for our data and computing frequencies rather than counts, and we draw a smooth curve that goes through the tops of the histogram bars. The plots on the following slide demonstrate the steps that lead to a smooth density:

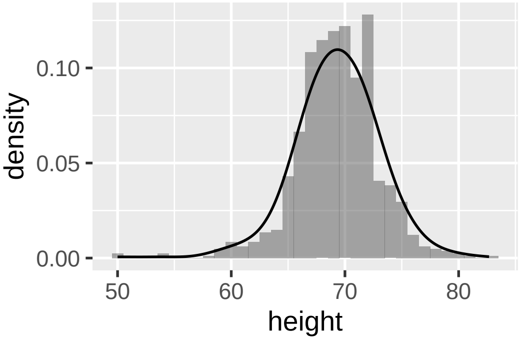
Smoothed Density



Smoothed Density

However, remember that **smooth** is a relative term. We can actually control the **smoothness** of the curve that defines the smooth density through an option in the function that computes the smooth density curve. For example:

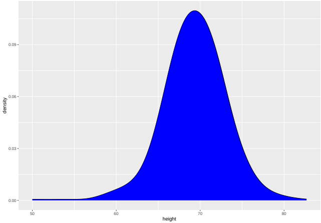




Density plots

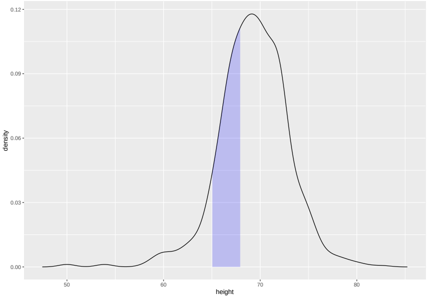
To change the smoothness of the density, we use the `adjust` argument to multiply the default value by that `adjust`. For example, if we want the bandwidth to be twice as big we use:

```r
heights %>% filter(sex=="Male") %>%
  ggplot(aes(height)) +
  geom_density(fill= "blue", adjust=2)
```



Interpreting the y-axis

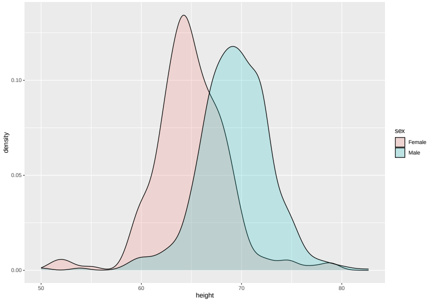
The y-axis of a smooth density plot is scaled so that the area under the density curve adds up to 1. To determine the proportion of data in that interval we compute the proportion of the total area contained in that interval. For example, here are the proportion of values between 65 and 68:



Densities Permit Stratification

Another advantage of smooth densities over histograms is that densities make it easier to compare two distributions. Comparing male and female heights:

```r
heights %>% ggplot(aes(height, fill=sex)) +
  geom_density(alpha = 0.2)
```



The Normal Distribution

A **normal distribution**, also known as a bell curve or Gaussian distribution, is a very famous mathematical concept.

Normal (or approximately normal) distributions occur in many situations, including gambling winnings, heights, weights, blood pressure, standardized test scores, and experimental measurement.

Here we focus on how a normal distribution helps us summarize and explore data.

The Normal Distribution

The normal distribution is defined with a mathematical formula. For any interval (a, b), the proportion of values in that interval can be computed using this formula:

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

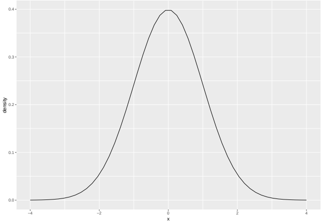
You don't need to memorize or understand the details of the formula. But note that it is completely defined by just two parameters: the *mean* (μ) and the *standard deviation* (σ).

The rest of the symbols in the formula represent the interval we determine, a and b, and known mathematical constants π and e.

The Normal Distribution

The distribution is symmetric, centered at μ, and most values (about 95%) are within 2σ from μ. Here is a normal distribution with $\mu = 0$ and $\sigma = 1$:

```
mu <- 0; s <- 1; norm_dist <-
  data.frame(x=seq(-4,4,len=50)*s+mu) %>%
  mutate(density=dnorm(x,mu,s))
norm_dist %>% ggplot(aes(x,density)) + geom_line()
```



The Normal Distribution

Let’s compute the values for the height for males which we will store in the object x:

```r
index <- heights$sex == "Male"
x <- heights$height[index]
```

The pre-built functions `mean` and `sd` can be used here:

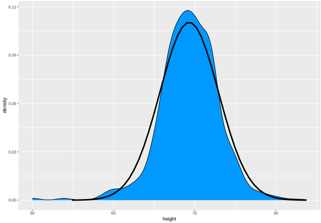
```r
m <- mean(x)
s <- sd(x)
c(mu = m, sig = s)
```

```
##        mu       sig
## 69.314755  3.611024
```

The Normal Distribution

Here is a plot of the smooth density of our heights data and the normal distribution with $\mu = 69.3$ and $\sigma = 3.6$:

```
norm_dist <- data.frame(x = seq(-4, 4, len=50)*s + m) %>%
  mutate(density = dnorm(x, m, s))
heights %>% filter(sex == "Male") %>% ggplot(aes(height)) +
  geom_density(fill="#0099FF") +
  geom_line(aes(x, density),  data = norm_dist, lwd=1.5)
```



The normal distribution seems to be a good approximation here.

Quantile-quantile Plots

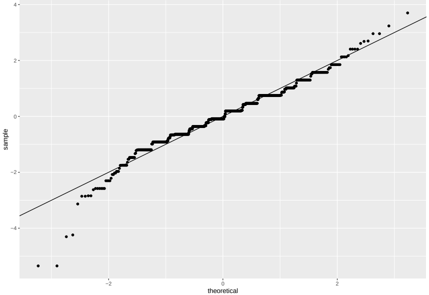
A systematic way to assess how well the normal distribution fits the data is to use a quantile-quantile plot (QQ-plot). To construct a QQ-plot, we do the following:

1. Define a vector of m proportions $p_1, p_2, \ldots, p_m$.
2. Define a vector of quantiles $q_1, \ldots, q_m$ for your data for the proportions $p_1, \ldots, p_m$. These are the **sample quantiles**.
3. Define a vector of theoretical quantiles for the proportions $p_1, \ldots, p_m$ for a normal distribution with the same average and standard deviation as the data.
4. Plot the sample quantiles versus the theoretical quantiles.

Quantile-quantile Plots

In practice we can use ggplot2 to generate a QQ plot:

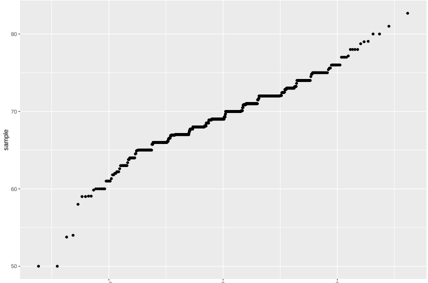
```
heights %>% filter(sex == "Male") %>%
  ggplot(aes(sample = scale(height))) +
  geom_qq() + geom_abline()
```



QQ-plots

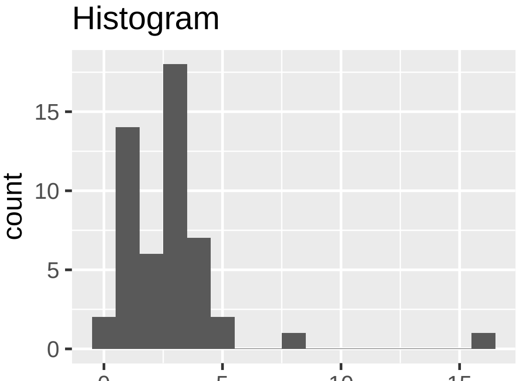
For qq-plots we use the `geom_qq` geometry. From the help file, we learn that we need to specify the `sample` (we will learn about samples in a later chapter). Here is the qqplot for men heights.

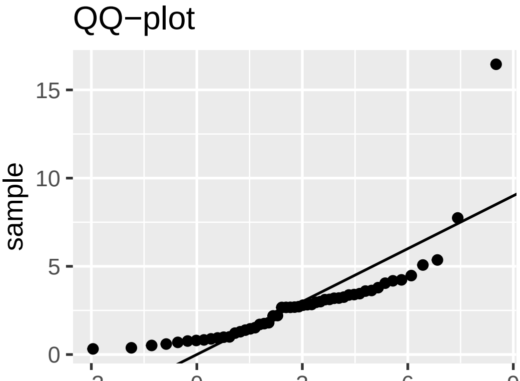
```
heights %>% filter(sex=="Male") %>%
  ggplot(aes(sample = height)) + geom_qq()
```



Boxplots

To introduce boxplots we will go back to the US murder data. Suppose we want to summarize the murder rate distribution. Using the data visualization technique we have learned, we can quickly see that the normal approximation does not apply here:





Boxplots

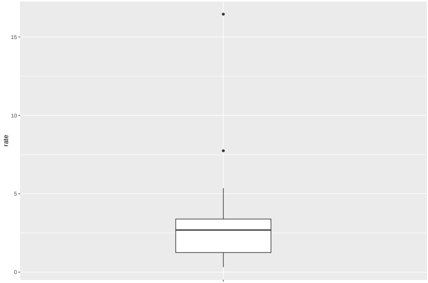
In this case, the histogram above or a smooth density plot would serve as a relatively succinct summary.

Now suppose those used to receiving just two numbers as summaries ask us for a more compact numerical summary.

Here Tukey offered some advice. Provide a five-number summary composed of the range along with the quartiles (the 25th, 50th, and 75th percentiles). Tukey further suggested that we ignore **outliers** when computing the range and instead plot these as independent points. We provide a detailed explanation of outliers later. Finally, he suggested we plot these numbers as a "box" with "whiskers".

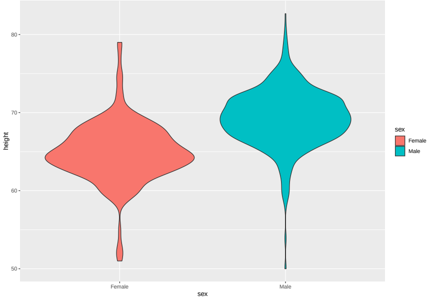
Boxplots

The box defined by the 25% and 75% percentile and the whiskers showing the range. The distance between these two is called the **interquartile range**. The two points are outliers according to Tukey's definition. The median is shown with a horizontal line. Today, we call these **boxplots**.



Boxplots

The geometry for boxplot is `geom_boxplot`. As discussed, boxplots are useful for comparing distributions. For example, below are the previously shown heights for women, but compared to men. For this geometry, we need arguments `x` as the categories, and `y` as the values.

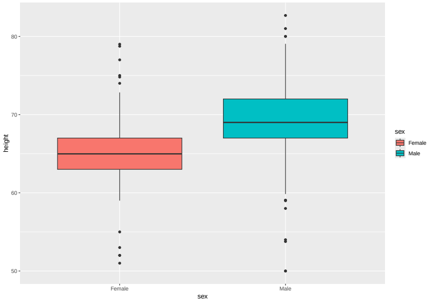


Case study: describing student heights (continued)

Using the histogram, density plots, and QQ-plots, we have become convinced that the male height data is well approximated with a normal distribution. In this case, we report back to ET a very succinct summary: male heights follow a normal distribution with an average of 69.3 inches and a SD of 3.6 inches. With this information, ET will have a good idea of what to expect when he meets our male students. However, to provide a complete picture we need to also provide a summary of the female heights.

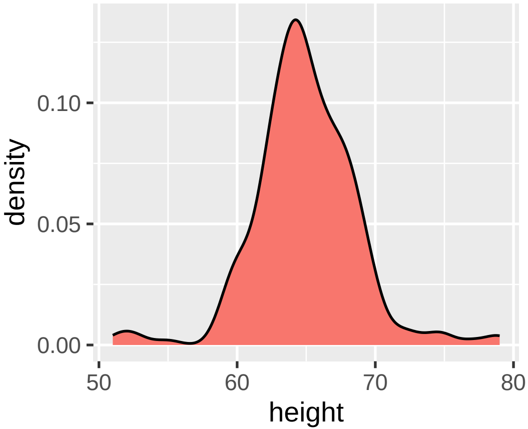
We learned that boxplots are useful when we want to quickly compare two or more distributions. Here are the heights for men and women:

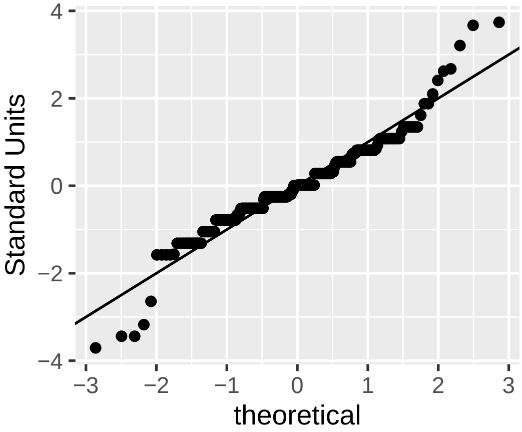
Case study: describing student heights (continued)



The plot immediately reveals that males are, on average, taller than females. The standard deviations appear to be similar. But does the normal approximation also work for the female height data collected by the survey? We expect that they will follow a normal distribution, just like males. However, exploratory plots reveal that the approximation is not as useful:

Case study: describing student heights (continued)





Case study: describing student heights (continued)

Regarding the five smallest values, note that these values are:

```r
heights %>% filter(sex == "Female") %>%
  top_n(5, desc(height)) %>%
  pull(height)
```

```
## [1] 51 53 55 52 52
```

Because these are reported heights, a possibility is that the student meant to enter `5'1"`, `5'2"`, `5'3"` or `5'5"`.

Session Info

```r
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
## 
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib 
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.0
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## time zone: America/Denver
## tzcode source: internal
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
##  [1] gridExtra_2.3   dslabs_0.8.0    lubridate_1.9.4 forcats_1.0.0  
##  [5] stringr_1.5.1   dplyr_1.1.4     purrr_1.0.2     readr_2.1.5    
##  [9] tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0
## 
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6      crayon_1.5.3      compiler_4.4.2    tinytex_0.54     
##  [5] tidyselect_1.2.1  scales_1.3.0      yaml_2.3.10       fastmap_1.2.0    
##  [9] R6_2.5.1          labeling_0.4.3    generics_0.1.3    knitr_1.49
```