Univariate Descriptives

Will Doyle

9/9/2021

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
nba<-readRDS("nba_players_2018.Rds")
```

Univariate Data Analysis

Univariate is pretty much what it sounds like: one variable. When undertaking univariate data analysis, we need first and foremost to figure what type of variable it is that we're working with. Once we do that, we can choose the appropriate use of the variable, either as an outcome or as a possible predictor.

Motivating Question

Today we'll be working with data from every NBA player who was active during the 2018-19 season. We're interested in the following questions:

- Do certain colleges produce players that have more field goals? What about free throw percentage above a certain level? Are certain colleges in the east or the west more likely to produce higher scorers? How does this vary as a player has more seasons?

To answer these questions we need to look at the following variables:

- Field goals
- Free throw percentage above .25
- Colleges
- Player seasons
- Region

We're going to go through a pretty standard set of steps for each variable. First, examine some cases. Second, based on our examination, we'll try either a plot or a table. Once we've seen the plot or the table, we'll think a bit about ordering, and then choose an appropriate measure of central tendency, and maybe variation.

Types of Variables

It's really important to understand the types of variables you're working with. Many times analysts are indifferent to this step particularly with larger datasets. This can lead to a great deal of confusion down the road. Below are the variable types we'll be working with this semester and the definition of each.

- Continuous: a continuous variable can theoretically be subdivided at any arbitrarily small measure and can still be identified. You may have encountered further subdivision of continuous variables into "interval" or "ratio" data in other classes. We RARELY use these distinctions in practice. The distinction between a continuous and a categorical variable is hugely consequential, but the distinction between interval and ratio is not really all that important in practice.

The mean is the most widely used measure of central tendency for a continuous variable. If the distribution of the variable isn't very symmetric or there are large outliers, then the median is a much better measure of central tendency.

- Categorical: a categorical variables divides the sample up into a set of mutually exclusive and exhaustive categories. Mutually exclusive means that each case can only be one, and exhaustive means that the categories cover every possible option. Categorical is sort of the "top" level classification for variables of this type. Within the broad classification of categorical there are multiple types of other variables.
- Categorical: ordered– an ordered categorical variable has– you guessed it– some kind of sensible order that can be applied. For instance, the educational attainment of an individual: high shool diploma, associates degree, bachelor's degree, graduate degree– is an ordered categorical variable.

Ordered categorical variables should be arranged in the order of the variable, with proportions or percentages associated with each order. The mode, or the category with the highest proportion, is a reasonable measure of central tendency, but with fewer than ten categories the analyst should generally just show the proportion in each category.

- Categorical: ordered, binary– an ordered binary variable has just two levels, but can be ordered. For instance, is a bird undertaking its first migration: yes or no? A "no" means that the bird has more than one.

The mean of a binary variable is exactly the same thing as the proportion of the sample with that characteristic. So, the mean of a binary variable for "first migration" where 1="yes" will give the proportion of birds migrating for the first time.

- Categorical: unordered– an unordered categorical variable has no sensible ordering that can be applied. Think about something like college major. There's no "number" we might apply to philosophy that has any meaningful distance from a number we might apply to chemical engineering.

Unlike an ordered variable, an unordered categorical variable should be ordered in terms of the proportions falling into each of the categories. As with an unordered variable, it's best just to show the proportions in each category for variables with less than ten levels. The mode is a reasonable single variable summary of an unordered categorical variable.

- Categorical: unordered, binary– this kind of variable has no particular order, but can be just binary. A "1" means that the case has that characteristics, a "0" means the case does not have that characteristic. For instance, whether a tree is deciduous or not.

An unordered binary variable can also be summarized by the mean, which is the same thing as the proportion of the sample with that characteristic.

In R, categorical variables CAN be stored as text or numbers. Don't count on the data to help you out– you as the analyst need to figure this out.

We probably need to talk about factors. In R, a factor is a way of storing categorical variables. The factor provides additional information, including an ordering of the variable and a number assigned to each "level" of the factor. A categorical variable is a general term that's understood across statistics. A factor variable is a specific R term. Most of the time it's best not to have a categorical variable structured as a factor unless you know you want it to be a factor. More on this later . . .

The Process: #TrustTheProcess

I'm going to walk you through how an analyst might typically decide what type of variables they're working with. It generally works like this:

1. Take a look at a few observations and form a guess as to what type of variable it is.
2. Based on that guess, create an appropriate plot or table.Plots are generally used for continuous variables, while tables mostly are helpful in sorting out categorical variables. No promises, though.

3. If the plot or table looks as expected, calculate some summary measures. If not, go back to 1.

"Glimpse" to start: what's in here anyway?

Basically the first thing we're going to do with any dataset is just to take a quick look. We can call the data itself, but that will just show the first few cases and the first few variables. Far better is the glimpse command, which shows us all variables and the first few observations for all of the variables. Here's a link to the codebook for this dataset.

The six variables we're going to think about are field goals, free throw percentage, college attended, rookie season, conference and nba seasons played.

```
glimpse(nba)
```

```
## Rows: 530
## Columns: 37
## $ namePlayer         <chr> "LaMarcus Aldridge", "Quincy Acy", "Steven Adams", ~
## $ idPlayer           <dbl> 200746, 203112, 203500, 203518, 1628389, 1628959, 1~
## $ slugSeason         <chr> "2018-19", "2018-19", "2018-19", "2018-19", "2018-1~
## $ numberPlayerSeason <dbl> 12, 6, 5, 2, 1, 0, 0, 0, 0, 0, 8, 5, 4, 3, 1, 1, 1,~
## $ isRookie           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE~
## $ slugTeam           <chr> "SAS", "PHX", "OKC", "OKC", "MIA", "CHI", "UTA", "C~
## $ idTeam             <dbl> 1610612759, 1610612756, 1610612760, 1610612760, 161~
## $ gp                 <dbl> 81, 10, 80, 31, 82, 10, 38, 19, 34, 7, 81, 72, 43, ~
## $ gs                 <dbl> 81, 0, 80, 2, 28, 1, 2, 3, 1, 0, 81, 72, 40, 4, 80,~
## $ fgm                <dbl> 684, 4, 481, 56, 280, 13, 67, 11, 38, 3, 257, 721, ~
## $ fga                <dbl> 1319, 18, 809, 157, 486, 39, 178, 36, 110, 10, 593,~
## $ pctFG              <dbl> 0.519, 0.222, 0.595, 0.357, 0.576, 0.333, 0.376, 0.~
## $ fg3m               <dbl> 10, 2, 0, 41, 3, 3, 32, 6, 25, 0, 96, 52, 9, 24, 6,~
## $ fg3a               <dbl> 42, 15, 2, 127, 15, 12, 99, 23, 74, 4, 280, 203, 34~
## $ pctFG3             <dbl> 0.2380952, 0.1333333, 0.0000000, 0.3228346, 0.20000~
## $ pctFT              <dbl> 0.847, 0.700, 0.500, 0.923, 0.735, 0.667, 0.750, 1.~
## $ fg2m               <dbl> 674, 2, 481, 15, 277, 10, 35, 5, 13, 3, 161, 669, 1~
## $ fg2a               <dbl> 1277, 3, 807, 30, 471, 27, 79, 13, 36, 6, 313, 1044~
## $ pctFG2             <dbl> 0.5277995, 0.6666667, 0.5960347, 0.5000000, 0.58811~
## $ agePlayer          <dbl> 33, 28, 25, 25, 21, 21, 23, 22, 23, 26, 28, 24, 25,~
## $ minutes            <dbl> 2687, 123, 2669, 588, 1913, 120, 416, 194, 428, 22,~
## $ ftm                <dbl> 349, 7, 146, 12, 166, 8, 45, 4, 7, 1, 150, 500, 37,~
## $ fta                <dbl> 412, 10, 292, 13, 226, 12, 60, 4, 9, 2, 173, 686, 6~
## $ oreb               <dbl> 251, 3, 391, 5, 165, 11, 3, 3, 11, 1, 112, 159, 48,~
## $ dreb               <dbl> 493, 22, 369, 43, 432, 15, 20, 16, 49, 3, 498, 739,~
## $ treb               <dbl> 744, 25, 760, 48, 597, 26, 23, 19, 60, 4, 610, 898,~
## $ ast                <dbl> 194, 8, 124, 20, 184, 13, 25, 5, 65, 6, 104, 424, 1~
## $ stl                <dbl> 43, 1, 117, 17, 71, 1, 6, 1, 14, 2, 68, 92, 54, 22,~
## $ blk                <dbl> 107, 4, 76, 6, 65, 0, 6, 4, 5, 0, 33, 110, 37, 13, ~
## $ tov                <dbl> 144, 4, 135, 14, 121, 8, 33, 6, 28, 2, 72, 268, 58,~
## $ pf                 <dbl> 179, 24, 204, 53, 203, 7, 47, 13, 45, 4, 143, 232, ~
```

```
## $ pts                 <dbl> 1727, 17, 1108, 165, 729, 37, 211, 32, 108, 7, 760,~
## $ urlNBAAPI           <chr> "https://stats.nba.com/stats/playercareerstats?Leag~
## $ n                   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ org                 <fct> Texas, NA, Other, FC Barcelona Basquet, Kentucky, N~
## $ country             <chr> NA, NA, NA, "Spain", NA, NA, NA, NA, NA, NA, NA, "S~
## $ idConference        <int> 2, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, ~
```

Continuous

Let's start by taking a look at field goals. It seems pretty likely that this is a continuous variable. Let's take a look at the top 50 spots.

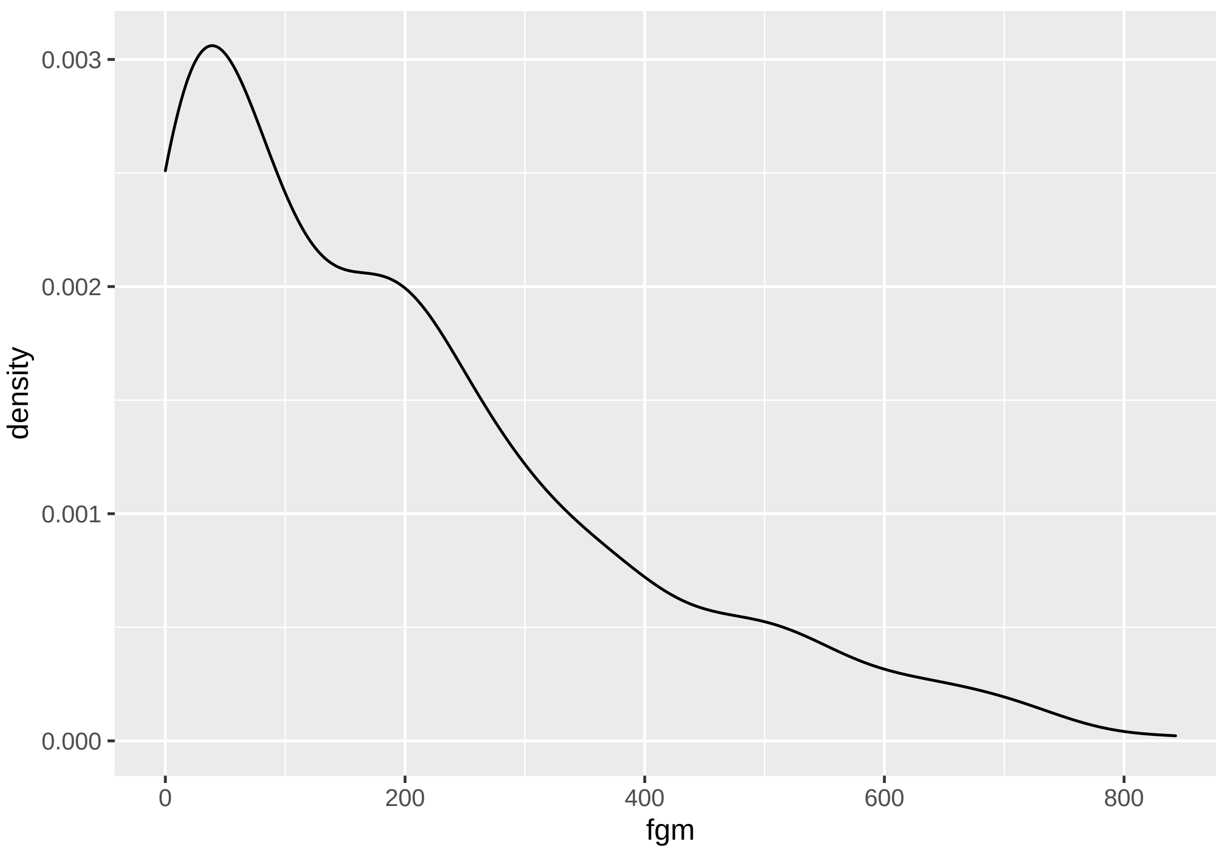
```
nba%>% ## Start with the dataset
  select(namePlayer,slugTeam,fgm)%>% ## and then select a few variables
  arrange(-fgm)%>% ## arrange in reverse order of field goals
  print(n=50) ## print out the top 50
```

```
## # A tibble: 530 x 3
##    namePlayer            slugTeam   fgm
##    <chr>                 <chr>    <dbl>
##  1 James Harden          HOU        843
##  2 Bradley Beal          WAS        764
##  3 Kemba Walker          CHA        731
##  4 Giannis Antetokounmpo MIL        721
##  5 Kevin Durant          GSW        721
##  6 Paul George           OKC        707
##  7 Nikola Vucevic        ORL        701
##  8 LaMarcus Aldridge     SAS        684
##  9 Damian Lillard        POR        681
## 10 Karl-Anthony Towns    MIN        681
## 11 Donovan Mitchell      UTA        661
## 12 D'Angelo Russell      BKN        659
## 13 Klay Thompson         GSW        655
## 14 Stephen Curry         GSW        632
## 15 DeMar DeRozan         SAS        631
## 16 Russell Westbrook     OKC        630
## 17 Buddy Hield           SAC        623
## 18 Blake Griffin         DET        619
## 19 Nikola Jokic          DEN        616
## 20 Tobias Harris         MIN        611
## 21 Kyrie Irving          BOS        604
## 22 Devin Booker          PHX        586
## 23 Joel Embiid           PHI        580
## 24 CJ McCollum           POR        571
## 25 Julius Randle         NOP        571
## 26 Andre Drummond        DET        561
## 27 Kawhi Leonard         TOR        560
## 28 LeBron James          LAL        558
## 29 Jrue Holiday          NOP        547
## 30 Montrezl Harrell      LAC        546
## 31 Ben Simmons           PHI        540
## 32 Anthony Davis         NOP        530
## 33 Zach LaVine           CHI        530
## 34 Jordan Clarkson       CLE        529
## 35 Trae Young            ATL        525
```

```
## 36 Bojan Bogdanovic      IND        522
## 37 Pascal Siakam         TOR        519
## 38 Collin Sexton         CLE        519
## 39 Jamal Murray          DEN        513
## 40 Deandre Ayton         PHX        509
## 41 Luka Doncic           DAL        506
## 42 Khris Middleton       MIL        506
## 43 De'Aaron Fox          SAC        505
## 44 Andrew Wiggins        MIN        498
## 45 Kyle Kuzma            LAL        496
## 46 Mike Conley           MEM        490
## 47 Lou Williams          LAC        484
## 48 Steven Adams          OKC        481
## 49 Rudy Gobert           UTA        476
## 50 Clint Capela          HOU        474
## # ... with 480 more rows
```

So what I'm seeing here is that field goals aren't "clumped" at certain levels. Let's confirm that by looking at a kernel density plot.

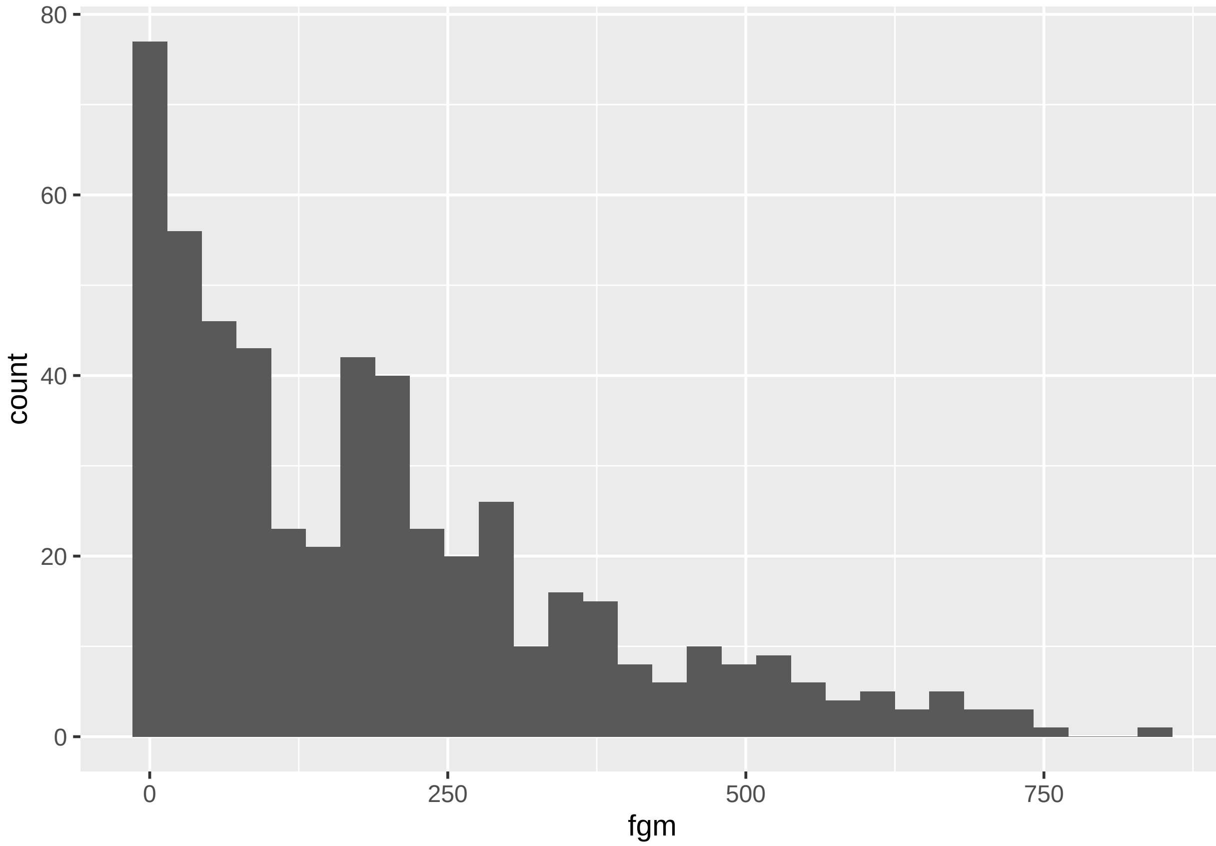
```
nba%>%
  ggplot(aes(x=fgm))+
  geom_density()
```



We can also use a histogram to figure out much the same thing.

```
nba%>%
  ggplot(aes(x=fgm))+
  geom_histogram()
```

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



Now, technically field goals don't meet the definition I set out above as being a continuous variable because they aren't divisible below a certain amount. Usually in practice though we just ignore this– this variable is "as good as" continuous, given that it varies smoothly over the range and isn't confined to a relatively small set of possible values.

Quick Exercise: Do the same thing for field goal percentage and think about what kind of variable it is

Measures for Continuous Variables

The mean is used most of the time for continuous variables, but it's VERY sensitive to outliers. The median (50th percentile) is usually better, but it can be difficult to explain to general audiences.

```
nba%>%
  summarize(mean_fgm=mean(fgm))
```

```
## # A tibble: 1 x 1
##   mean_fgm
##      <dbl>
## 1     191.
```

```
nba%>%
  summarize(median_fgm=median(fgm))
```

```
## # A tibble: 1 x 1
##   median_fgm
##        <dbl>
```

```
## 1        157
```

In this case I'd really prefer the mean as a single measure of field goal production, but depending on the audience I still might just go ahead and use the median.

Quick Exercise: What measure would you prefer for field goal percentage? Why?

Categorical: ordered

Let's take a look at player seasons.

```
nba%>%
  select(namePlayer,numberPlayerSeason)%>%
  arrange(-numberPlayerSeason)%>%
  print(n=50)
```

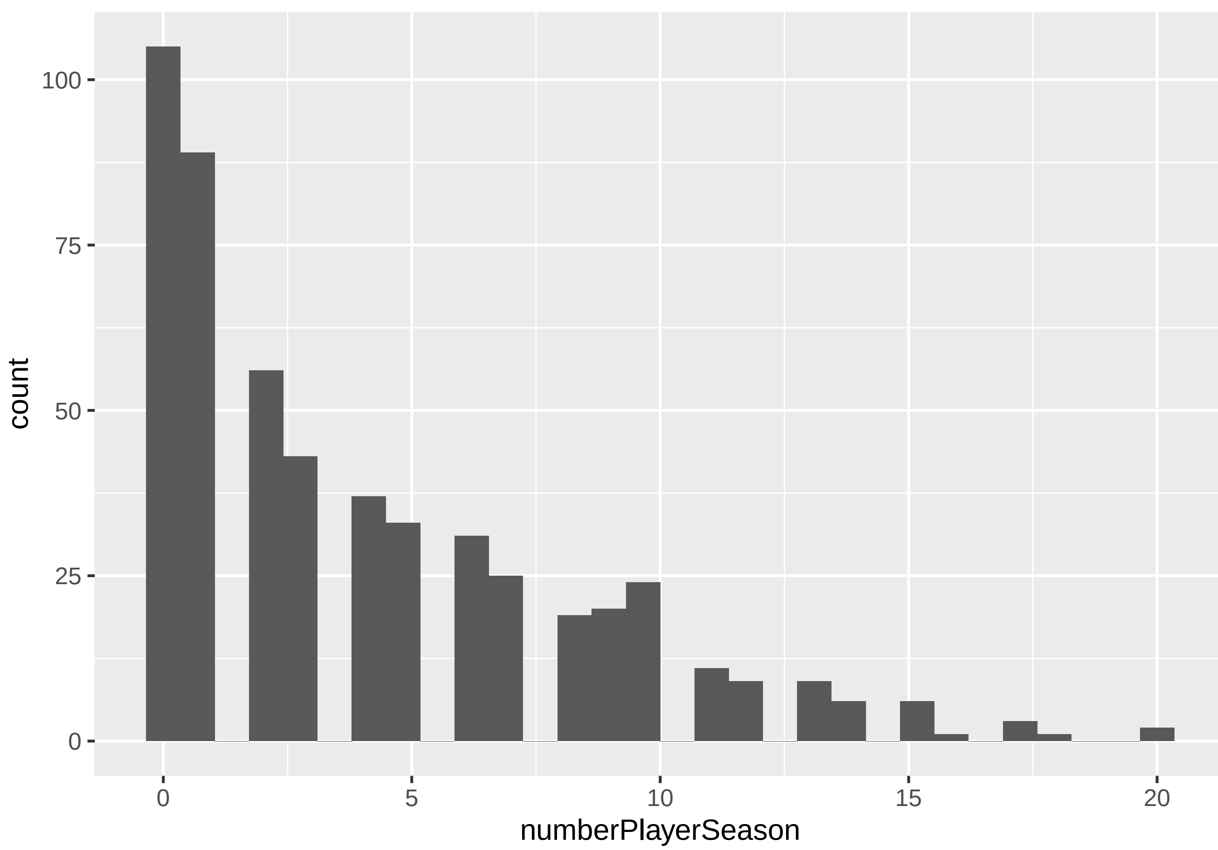
```
## # A tibble: 530 x 2
##    namePlayer        numberPlayerSeason
##    <chr>                          <dbl>
##  1 Vince Carter                      20
##  2 Dirk Nowitzki                     20
##  3 Jamal Crawford                    18
##  4 Tony Parker                       17
##  5 Tyson Chandler                    17
##  6 Pau Gasol                         17
##  7 Nene                              16
##  8 Carmelo Anthony                   15
##  9 Udonis Haslem                     15
## 10 LeBron James                      15
## 11 Zaza Pachulia                     15
## 12 Dwyane Wade                       15
## 13 Kyle Korver                       15
## 14 Luol Deng                         14
## 15 Devin Harris                      14
## 16 Dwight Howard                     14
## 17 Andre Iguodala                    14
## 18 JR Smith                          14
## 19 Trevor Ariza                      14
## 20 Andrew Bogut                      13
## 21 Jose Calderon                     13
## 22 Raymond Felton                    13
## 23 Amir Johnson                      13
## 24 Shaun Livingston                  13
## 25 Chris Paul                        13
## 26 Marvin Williams                   13
## 27 Lou Williams                      13
## 28 CJ Miles                          13
## 29 LaMarcus Aldridge                 12
## 30 J.J. Barea                        12
## 31 Channing Frye                     12
## 32 Rudy Gay                          12
## 33 Kyle Lowry                        12
## 34 Paul Millsap                      12
## 35 JJ Redick                         12
## 36 Rajon Rondo                       12
## 37 Thabo Sefolosha                   12
```

```
## 38 Marco Belinelli                  11
## 39 Mike Conley                      11
## 40 Kevin Durant                     11
## 41 Jared Dudley                     11
## 42 Marcin Gortat                    11
## 43 Gerald Green                     11
## 44 Al Horford                       11
## 45 Joakim Noah                      11
## 46 Thaddeus Young                   11
## 47 Nick Young                       11
## 48 Corey Brewer                     11
## 49 D.J. Augustin                    10
## 50 Jerryd Bayless                   10
## # ... with 480 more rows
```

Looks like it might be continuous? Let's plot it:

```
nba%>%
  ggplot(aes(x=numberPlayerSeason))+
  geom_histogram()
```

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



Nope. See how it falls into a small set of possible categories? This is an ordered categorical variable. That means we should calculate the proportions in each category

```
nba%>%
  group_by(numberPlayerSeason)%>%
  count(name="total_in_group")%>%
```

```
  ungroup()%>%
  mutate(proportion=total_in_group/sum(total_in_group))
```

```
## # A tibble: 20 x 3
##    numberPlayerSeason total_in_group proportion
##                 <dbl>          <int>      <dbl>
##  1                  0            105    0.198
##  2                  1             89    0.168
##  3                  2             56    0.106
##  4                  3             43    0.0811
##  5                  4             37    0.0698
##  6                  5             33    0.0623
##  7                  6             31    0.0585
##  8                  7             25    0.0472
##  9                  8             19    0.0358
## 10                  9             20    0.0377
## 11                 10             24    0.0453
## 12                 11             11    0.0208
## 13                 12              9    0.0170
## 14                 13              9    0.0170
## 15                 14              6    0.0113
## 16                 15              6    0.0113
## 17                 16              1    0.00189
## 18                 17              3    0.00566
## 19                 18              1    0.00189
## 20                 20              2    0.00377
```

What does this tell us?

Categorical: ordered, binary

Let's take a look at the variable for Rookie season.

```
nba%>%select(namePlayer,isRookie)
```

```
## # A tibble: 530 x 2
##    namePlayer             isRookie
##    <chr>                  <lgl>
##  1 LaMarcus Aldridge      FALSE
##  2 Quincy Acy             FALSE
##  3 Steven Adams           FALSE
##  4 Alex Abrines           FALSE
##  5 Bam Adebayo            FALSE
##  6 Rawle Alkins           TRUE
##  7 Grayson Allen          TRUE
##  8 Deng Adel              TRUE
##  9 Jaylen Adams           TRUE
## 10 DeVaughn Akoon-Purcell TRUE
## # ... with 520 more rows
```

Okay, so that's set to a logical. In R, TRUE or FALSE are special values that indicate the result of a logical question. In this it's whether or not the player is a rookie.

Usually we want a binary variable to have at least one version that's structured so that 1= TRUE and 2=FALSE. This makes data analysis much easier. Let's do that with this variable.

This code uses `ifelse` to create a new variable called `isRookiebin` that's set to 1 if the `isRookie` variable is true, and 0 otherwise.

```
nba<-nba%>%
  mutate(isRookie_bin=ifelse(isRookie==TRUE,1,0))
```

```
nba%>%summarize(mean=mean(isRookie_bin))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1 0.198
```

Categorical: unordered

College attended

```
nba%>%
  select(org)%>%
  glimpse()
```

```
## Rows: 530
## Columns: 1
## $ org <fct> Texas, NA, Other, FC Barcelona Basquet, Kentucky, NA, Duke, NA, NA~
```

This look like team or college names, so this would be a categorical variable.

```
nba%>%
  group_by(org)%>%
  count()%>%
  arrange(-n)%>%
  print(n=50)
```

```
## # A tibble: 68 x 2
## # Groups:   org [68]
##    org                         n
##    <fct>                   <int>
##  1 <NA>                      157
##  2 Other                      85
##  3 Kentucky                   25
##  4 Duke                       17
##  5 California-Los Angeles     15
##  6 Kansas                     11
##  7 Arizona                    10
##  8 Texas                      10
##  9 North Carolina              9
## 10 Michigan                    8
## 11 Villanova                   7
## 12 Indiana                     6
## 13 Southern California         6
## 14 Syracuse                    6
## 15 California                  5
## 16 Louisville                  5
## 17 Ohio State                  5
## 18 Wake Forest                 5
## 19 Colorado                    4
## 20 Connecticut                 4
```

```
## 21 Creighton                4
## 22 FC Barcelona Basquet     4
## 23 Florida                  4
## 24 Georgia Tech             4
## 25 Michigan State           4
## 26 Oregon                   4
## 27 Utah                     4
## 28 Washington               4
## 29 Wisconsin                4
## 30 Boston College           3
## 31 Florida State            3
## 32 Georgetown               3
## 33 Gonzaga                  3
## 34 Iowa State               3
## 35 Marquette                3
## 36 Maryland                 3
## 37 Miami (FL)               3
## 38 North Carolina State     3
## 39 Notre Dame               3
## 40 Oklahoma                 3
## 41 Purdue                   3
## 42 Southern Methodist       3
## 43 Stanford                 3
## 44 Tennessee                3
## 45 Virginia                 3
## 46 Anadolu Efes S.K.        2
## 47 Baylor                   2
## 48 Butler                   2
## 49 Cincinnati               2
## 50 Kansas State             2
## # ... with 18 more rows
```

Here we have a problem. If we're interested just in colleges, we're going to need to structure this a bit more. The code below filters out three categories that we don't want: missing data, anything classified as others, and sports teams from other countries. The last is incomplete– I probably missed some! If I were doing this for real, I would use a list of colleges and only include those names.

```
nba%>%
  filter(!is.na(org))%>%
  filter(!org=="Other")%>%
  filter(!str_detect(org,"CB|KK|rytas|FC|B.C.|S.K.|Madrid"))%>%
  group_by(org)%>%
  count()%>%
  arrange(-n)%>%
  print(n=50)
```

```
## # A tibble: 57 x 2
## # Groups:   org [57]
##    org                         n
##    <fct>                   <int>
##  1 Kentucky                   25
##  2 Duke                       17
##  3 California-Los Angeles     15
##  4 Kansas                     11
##  5 Arizona                    10
```

```
##   6 Texas                      10
##   7 North Carolina              9
##   8 Michigan                    8
##   9 Villanova                   7
##  10 Indiana                     6
##  11 Southern California         6
##  12 Syracuse                    6
##  13 California                  5
##  14 Louisville                  5
##  15 Ohio State                  5
##  16 Wake Forest                 5
##  17 Colorado                    4
##  18 Connecticut                 4
##  19 Creighton                   4
##  20 Florida                     4
##  21 Georgia Tech                4
##  22 Michigan State              4
##  23 Oregon                      4
##  24 Utah                        4
##  25 Washington                  4
##  26 Wisconsin                   4
##  27 Boston College              3
##  28 Florida State               3
##  29 Georgetown                  3
##  30 Gonzaga                     3
##  31 Iowa State                  3
##  32 Marquette                   3
##  33 Maryland                    3
##  34 Miami (FL)                  3
##  35 North Carolina State        3
##  36 Notre Dame                  3
##  37 Oklahoma                    3
##  38 Purdue                      3
##  39 Southern Methodist          3
##  40 Stanford                    3
##  41 Tennessee                   3
##  42 Virginia                    3
##  43 Baylor                      2
##  44 Butler                      2
##  45 Cincinnati                  2
##  46 Kansas State                2
##  47 Louisiana State             2
##  48 Memphis                     2
##  49 Missouri                    2
##  50 Murray State                2
## # ... with 7 more rows
```

```
nba%>%
  filter(!is.na(org))%>%
  filter(!org=="Other")%>%
  filter(!str_detect(org,"CB|KK|rytas|FC|B.C.|S.K.|Madrid"))%>%
  group_by(org)%>%
  count(name="total_in_group")%>%
  ungroup()%>%
```

```
  mutate(proportion=total_in_group/sum(total_in_group))%>%
  arrange(-proportion)
```

```
## # A tibble: 57 x 3
##    org                     total_in_group proportion
##    <fct>                            <int>      <dbl>
##  1 Kentucky                            25     0.0933
##  2 Duke                                17     0.0634
##  3 California-Los Angeles              15     0.0560
##  4 Kansas                              11     0.0410
##  5 Arizona                             10     0.0373
##  6 Texas                               10     0.0373
##  7 North Carolina                       9     0.0336
##  8 Michigan                             8     0.0299
##  9 Villanova                            7     0.0261
## 10 Indiana                              6     0.0224
## # ... with 47 more rows
```

Categorica: unordered, binary

Let's look at the nba conferences.

```
nba%>%select(idConference)%>%
  glimpse()
```

```
## Rows: 530
## Columns: 1
## $ idConference <int> 2, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, ~
```

It looks like conference is structured as numeric, but a "1" or a "2".

Quick Exercise: Figure out which conference is 1 and which is 2

Because it's best to have this set up as a binary. It's simple in this case: just subtract 1!

```
nba<-nba%>%
  mutate(conference_bin=idConference-1)
```

And now we can summarize it:

```
nba%>%
  summarize(mean(conference_bin))
```

```
## # A tibble: 1 x 1
##   `mean(conference_bin)`
##                    <dbl>
## 1                  0.492
```

Analysis

Ok, now that we know how this works, we can do some summary analysis. First of all, what does the total number of field goals made look like by college?

Top 50 Colleges by Total FG

```
nba%>%
  filter(!is.na(org))%>%
```

```r
  filter(!org=="Other")%>%
  filter(!str_detect(org,"CB|KK|rytas|FC|B.C.|S.K.|Madrid"))%>%
  group_by(org)%>%
  summarize(mean_fg=sum(fgm))%>%
  arrange(-mean_fg)%>%
  print(n=50)
```

```
## # A tibble: 57 x 2
##    org                     mean_fg
##    <fct>                     <dbl>
##  1 Kentucky                   6594
##  2 Duke                       4623
##  3 Texas                      3437
##  4 California-Los Angeles     3382
##  5 Kansas                     2765
##  6 Arizona                    2101
##  7 Oklahoma                   1767
##  8 Southern California        1758
##  9 Louisville                 1679
## 10 North Carolina             1659
## 11 Indiana                    1522
## 12 Ohio State                 1486
## 13 Michigan                   1392
## 14 Wake Forest                1364
## 15 Connecticut                1299
## 16 Villanova                  1222
## 17 Georgia Tech               1169
## 18 Tennessee                  1095
## 19 Stanford                    949
## 20 Utah                        943
## 21 Marquette                   873
## 22 Gonzaga                     863
## 23 Michigan State              820
## 24 Colorado                    818
## 25 Virginia                    816
## 26 Maryland                    811
## 27 Missouri                    756
## 28 California                  734
## 29 Florida State               733
## 30 Georgetown                  717
## 31 Memphis                     620
## 32 Florida                     618
## 33 North Carolina State        598
## 34 Boston College              586
## 35 Louisiana State             583
## 36 Syracuse                    567
## 37 Iowa State                  523
## 38 Butler                      459
## 39 Wisconsin                   456
## 40 Creighton                   432
## 41 Oregon                      352
## 42 Texas A&M                   322
## 43 Baylor                      312
## 44 Providence                  291
```

```
## 45 Purdue                   275
## 46 Notre Dame               263
## 47 Ulkerspor                252
## 48 Southern Methodist       246
## 49 Oklahoma State           242
## 50 West Virginia            236
## # ... with 7 more rows
```

Next, what about field goal percentage?

Top 50 Colleges by Average Field Goal Percent

```
nba%>%
  filter(!is.na(org))%>%
  filter(!org=="Other")%>%
  filter(!str_detect(org,"CB|KK|rytas|FC|B.C.|S.K.|Madrid"))%>%
  group_by(org)%>%
  summarize(mean_ftp=mean(pctFT))%>%
  arrange(-mean_ftp)%>%
  print(n=50)
```

```
## # A tibble: 57 x 2
##    org                   mean_ftp
##    <fct>                    <dbl>
##  1 Tennessee                0.842
##  2 Virginia                 0.833
##  3 Oklahoma                 0.823
##  4 North Carolina State     0.817
##  5 West Virginia            0.804
##  6 Ulkerspor                0.803
##  7 Missouri                 0.802
##  8 Wake Forest              0.802
##  9 Florida State            0.801
## 10 Murray State             0.798
## 11 Iowa State               0.795
## 12 Notre Dame               0.792
## 13 Memphis                  0.788
## 14 Florida                  0.784
## 15 Michigan                 0.783
## 16 Stanford                 0.779
## 17 Georgetown               0.775
## 18 Marquette                0.774
## 19 Utah                     0.770
## 20 Kansas State             0.767
## 21 Butler                   0.762
## 22 Gonzaga                  0.761
## 23 North Carolina           0.756
## 24 Villanova                0.755
## 25 Texas                    0.752
## 26 Connecticut              0.748
## 27 Providence               0.747
## 28 Boston College           0.742
## 29 Michigan State           0.730
## 30 Kansas                   0.729
```

```
## 31 Indiana                 0.729
## 32 Duke                    0.728
## 33 Baylor                  0.726
## 34 Arizona                 0.721
## 35 Pallacanestro Biella    0.718
## 36 Wisconsin               0.712
## 37 Kentucky                0.712
## 38 Georgia Tech            0.712
## 39 Louisiana State         0.709
## 40 Creighton               0.698
## 41 Maryland                0.695
## 42 Vanderbilt              0.688
## 43 Washington              0.680
## 44 Louisville              0.679
## 45 Ohio State              0.679
## 46 California              0.675
## 47 Southern Methodist      0.673
## 48 Oregon                  0.662
## 49 Texas A&M               0.652
## 50 Southern California     0.648
## # ... with 7 more rows
```

Quick Exercise: calculate field goals made by player season