Exploratory Data Analysis with R

Displaying Distributions and Statistical Hypothesis Testing

Xuemao Zhang
East Stroudsburg University

October 14, 2022

Outline

- Characterizing a Distribution
    - center
    - spread
    - error-bars
- Visualization of Statistical Hypothesis Testing

Characterizing a Distribution - Center

- Distribution is the description of data values and frequencies of a data set. We have seen **histograms** and **density plots** for univartiate data.
- **Mean**
  - $\bar{x} = \sum_{i=1}^{n} x_i / n$
  - Location *parameter*: for example $\mu = E(X)$

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

- **Median**

```
median(mtcars$mpg)
```

```
## [1] 19.2
```

- **Mode** is the most frequently occurring value in a data set.

```
library(modeest)
mfv(mtcars$mpg)
```

```
## [1] 10.4 15.2 19.2 21.0 21.4 22.8 30.4
```

Characterizing a Distribution - Center

- In some situations the **geometric mean** can be useful to describe the location of a distribution.
- It is formula is

$$mean_{geometric} = \left(\prod_{i=1}^{n} x_i\right)^{1/n} = exp\left(\frac{\sum_{i=1}^{n} ln(x_i)}{n}\right)$$

```
exp(mean(log(mtcars$mpg)))
```

```
## [1] 19.25006
```

```
library(psych)
geometric.mean(mtcars$mpg)
```

```
## [1] 19.25006
```

Characterizing a Distribution - Spread

- **Range**: max-min

```
a=range(mtcars$mpg)
a[2]-a[1]
```

```
## [1] 23.5
```

```
# max(mtcars$mpg)-min(mtcars$mpg)
```

- The **cumulative distribution function** or cdf of a (random) variable X, denoted by $F_X(x)$, is defined by

$$F_X(x) = P(X \leq x) \text{ for all } x.$$

Characterizing a Distribution - Spread

- **Percentiles** are just the inverse of the CDF, and give the value below which a given percentage of the data values occur.
  - The 50th percentile is the median.

```
quantile(mtcars$mpg, c(0.32, 0.50, 0.97))
```

```
##    32%    50%    97%
## 16.352 19.200 32.505
```

Characterizing a Distribution - Spread

- Sample **variance**
  - $s^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)$
- Sample **standard deviation**
  - $s = \sqrt{s^2}$

```
var(mtcars$mpg);
```

```
## [1] 36.3241
```

```
sd(mtcars$mpg);
```

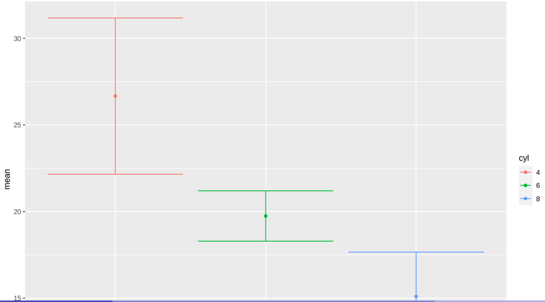
```
## [1] 6.026948
```

- The **standard error** is the estimate of the standard deviation of a statistic when the statistics is considered as a random variable.
- For normally distributed data, the standard error (SE) of the sample mean $\bar{x}$ is $SE(\bar{x}) = \frac{s}{\sqrt{n}}$.

Characterizing a Distribution - Error Bars

- Error-bars are a common way to show mean value and variability (eg., standard deviation) when comparing measurement values. We must explicitly state if the error-bars correspond to what type of spread: the standard deviation or standard error or some other measures.

```
library(dplyr)
mtcars$cyl=as.factor(mtcars$cyl)
mtcars%>%group_by(cyl)%>%summarise(mean=mean(mpg),sd=sd(mpg)) -> sum_data
ggplot(data=sum_data, aes(x=cyl,y=mean, color=cyl)) +
  geom_point()+
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd))
```



Characterizing a Distribution - Error Bars

- Can we plot error bars with point plot?
  - We can use `mutate` instead of `summarise`

```
mtcars%>%group_by(cyl)%>%mutate(mean=mean(mpg),sd=sd(mpg)) %>%
  ggplot(aes(x=cyl,y=mean, color=cyl)) +
  geom_point(color="black")+
  geom_jitter(aes(x=cyl,y=mpg))+ #or geom_point(aes(x=cyl,y=mpg))
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd))
```

- or a better way

```
library(Hmisc)
mtcars%>%ggplot(aes(x=cyl,y=mpg, color=cyl)) +
  geom_jitter()+
  stat_summary(fun.data="mean_sdl", fun.args = list(mult=1),
               geom="crossbar", width=0.5)
```

- For more information about error bars, see ggplot2 error bars : Quick start guide - R software and data visualization: http://www.sthda.com/english/wiki/ggplot2-error-bars-quick-start-guide-r-software-and-data-visualization

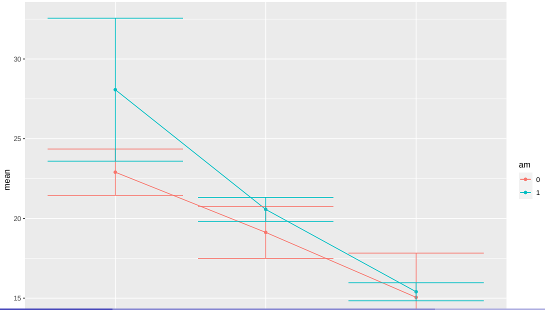
Characterizing a Distribution - Error Bars

- Line plot with error bars

```
mtcars$cyl=as.factor(mtcars$cyl);
mtcars$am=as.factor(mtcars$am);
mtcars%>%group_by(cyl,am)%>%summarise(mean=mean(mpg),sd=sd(mpg))->sum_data2;
```

```
## `summarise()` has grouped output by 'cyl'. You can override using the `.groups`
## argument.
```

```
ggplot(data=sum_data2, aes(x=cyl,y=mean, color=am)) +
  geom_point()+
  geom_line(aes(group=am))+
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd));
```



Characterizing a Distribution - Spread

- In statistical analysis of a data set it is common to find the **confidence interval** of an unknown parameter
- For example, the $100(1-\alpha)\%$ of the mean parameter is

$$\mathit{estimate} \pm \mathit{quantile}_{1-\alpha/2} \cdot \mathit{SE}(\mathit{estimate}).$$

```
test=t.test(mtcars$mpg,alternative="two.sided",conf.level = 0.95)
test$conf.int
```

```
## [1] 17.91768 22.26357
## attr(,"conf.level")
## [1] 0.95
```

```
ME=(test$conf.int[2]-test$conf.int[1])/2;
#half length of the CI symmetric about the sample mean
```

- Thus, we can visualize confidence intervals as well.

Visualizing Statistical Hypothesis Testing

- `infer` implements an expressive grammar to perform statistical inference that coheres with the tidyverse design framework. See infer Reference manual
- In hypothesis testing, we want to answer "is the effect/difference in our observed data'' significant?
  - Assume population based on null hypothesis is true
  - Test statistic: standardized point estimate to have a typical distribution
  - p-value: the probability that the observed test statistic is due to chance (small p-value leads to rejection of null hypothesis)
- Four main verbs (functions)
  - `specify()` allows you to specify the variable, or relationship between variables, that you're interested in.
  - `hypothesize()` allows you to declare the null hypothesis.
  - `generate()` allows you to generate/simulate data reflecting the null hypothesis.
  - `calculate()` allows you to calculate a distribution of statistics from the generated data to form the null distribution.

Visualizing Statistical Hypothesis Testing

- For the data `mtcars$mpg`, let's test $H_0 : \mu = 18$ versus $H_1 : \mu \neq 18$.
- The `specify` function can be used to specify which of the variables in the dataset you're interested in.

```
library(infer);
mtcars%>%specify(response = mpg)
```

```
## Response: mpg (numeric)
## # A tibble: 32 x 1
##       mpg
##     <dbl>
##  1   21
##  2   21
##  3   22.8
##  4   21.4
##  5   18.7
##  6   18.1
##  7   14.3
##  8   24.4
##  9   22.8
## 10   19.2
## # ... with 22 more rows
```

Visualizing Statistical Hypothesis Testing

- `hypothesize()`: Declaring the Null Hypothesis

```
mtcars%>%specify(response = mpg)%>%
hypothesize(null = "point", mu = 18)
```

```
## Response: mpg (numeric)
## Null Hypothesis: point
## # A tibble: 32 x 1
##      mpg
##    <dbl>
##  1  21
##  2  21
##  3  22.8
##  4  21.4
##  5  18.7
##  6  18.1
##  7  14.3
##  8  24.4
##  9  22.8
## 10  19.2
## # ... with 22 more rows
```

Visualizing Statistical Hypothesis Testing

- `generate()`: Generating the Null Distribution. We can construct a null distribution based on this hypothesis, We can do this using one of several methods
  - `bootstrap`: A bootstrap sample will be drawn for each replicate, where a sample of size equal to the input sample size is drawn (with replacement) from the input sample data.
  - `permute`: For each replicate, each input value will be randomly reassigned (without replacement) to a new output value in the sample.
  - `simulate`: A value will be sampled from a theoretical distribution with parameters specified in hypothesize() for each replicate. (This option is currently only applicable for testing point estimates.)

Visualizing Statistical Hypothesis Testing

```
mtcars%>%specify(response = mpg)%>%
hypothesize(null = "point", mu = 18)%>%
  generate(reps = 1000, type = "bootstrap")
```

```
## Response: mpg (numeric)
## Null Hypothesis: point
## # A tibble: 32,000 x 2
## # Groups:   replicate [1,000]
##    replicate   mpg
##        <int> <dbl>
##  1         1 17.1
##  2         1 13.7
##  3         1 17.1
##  4         1 12.6
##  5         1 28.3
##  6         1  8.31
##  7         1 12.2
##  8         1 16.0
##  9         1  8.31
## 10         1 15.7
```

Visualizing Statistical Hypothesis Testing

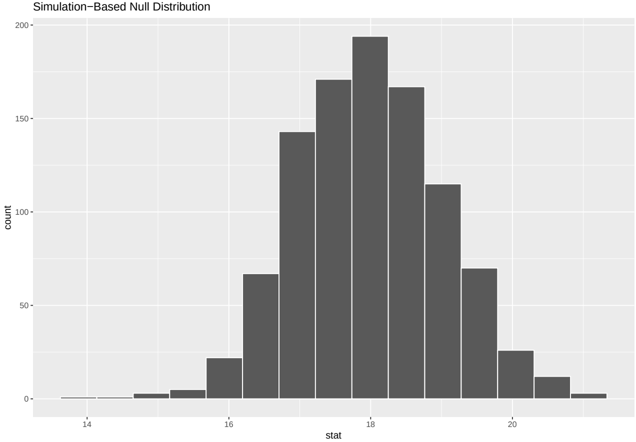
- `calculate()`: Calculating Summary Statistics (point estimation). The function, for one, takes in a stat argument, which is currently one of "mean", "median", "sum", "sd", "prop", "count", "diff in means", "diff in medians", "diff in props", "Chisq", "F", "t", "z", "slope", or "correlation".

```
null_dist <-mtcars%>%specify(response = mpg)%>%
hypothesize(null = "point", mu = 18)%>%
  generate(reps = 1000, type = "bootstrap")%>%
  calculate(stat = "mean")
```

Visualizing Statistical Hypothesis Testing

- Visualize the null distribution.

```
null_dist%>%visualize()
```



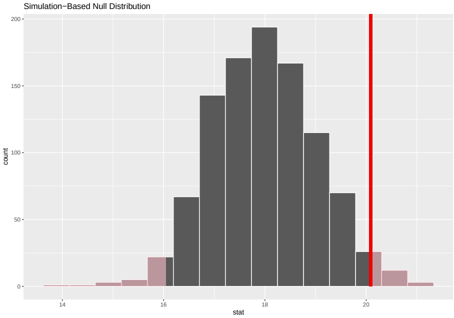
Visualizing Statistical Hypothesis Testing

- Find the point estimate

```
point_estimate <- mtcars%>%specify(response = mpg)%>%
  calculate(stat = "mean")
```

- Where does our sample's observed statistic lie on this distribution? We can use the obs_stat argument to specify this.

```
null_dist%>%visualize()+
  shade_p_value(obs_stat=point_estimate,direction="two-sided")
```



Visualizing Statistical Hypothesis Testing

- Get a p-value for the test

```
p_value <- null_dist %>%
  get_p_value(obs_stat = point_estimate, direction = "two-sided")
p_value
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.044
```

Visualizing Statistical Hypothesis Testing

- To get a confidence interval around our estimate

```
# start with the null distribution
null_dist %>%
  # calculate the confidence interval around the point estimate
  get_confidence_interval(point_estimate=point_estimate,
                          # at the 95% confidence level
                          level = .95,
                          # using the standard error
                          type = "se")
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     18.1     22.1
```

Visualizing Statistical Hypothesis Testing

- The above inference is using non-parametric method
- `infer` also provides functionality to use **theoretical**(parametric) methods for "Chisq", "F" and "t" test.
- Define a t-distribution to use the t-test

```
null_dist_theoretical<-mtcars%>%specify(response = mpg)%>%
  assume(distribution = "t")
```

- Calculate the test-statistic

```
t_statistic<- mtcars%>%specify(response = mpg)%>%
  hypothesize(null = "point", mu = 18)%>%
  calculate(stat = "t")
```

Visualizing Statistical Hypothesis Testing

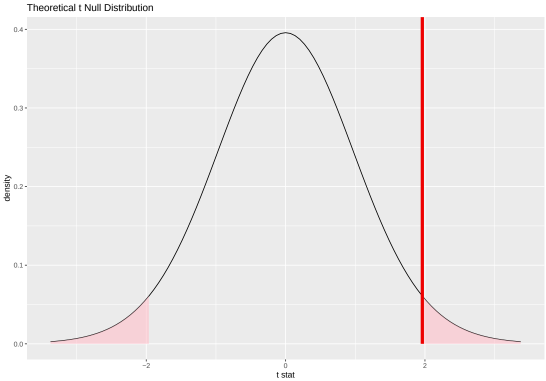
- Visulization of the p-value of the t-test

```
t_test(mtcars, response=mpg,mu = 18, alternative = "two-sided")
```

```
## # A tibble: 1 x 7
##   statistic  t_df p_value alternative estimate lower_ci upper_ci
##       <dbl> <dbl>   <dbl> <chr>          <dbl>    <dbl>    <dbl>
## 1      1.96    31  0.0588 two.sided       20.1     17.9     22.3
```

Visualizing Statistical Hypothesis Testing

```
visualize(null_dist_theoretical, method = "theoretical") +
  shade_p_value(obs_stat = t_statistic, direction = "two-sided")
```



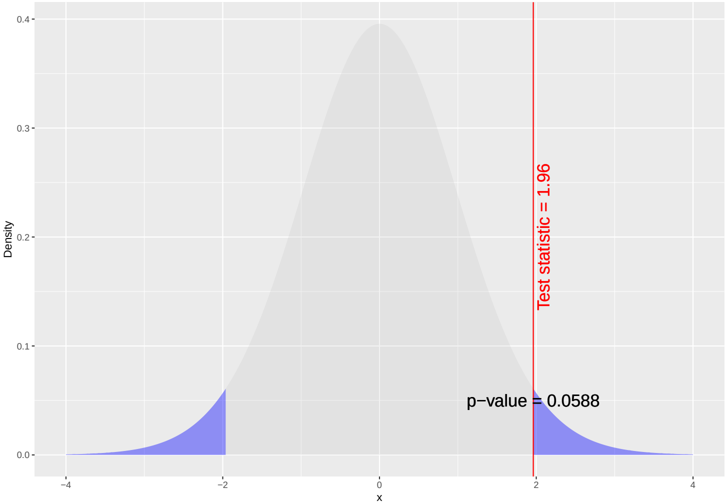
Visualizing Statistical Hypothesis Testing

- We may use ggplot2 to visualize the test statistic and p-value

```
test=t.test(mtcars$mpg, mu=18, alternative = "two.sided")
t_stat=test$statistic
df=test$parameter
pvalue=test$p.value
library(ggplot2)
ggplot(data.frame(x = c(-4, 4)), aes(x = x))+
  geom_area(stat = "function", fun = dt, args = list(df=df),
            xlim = c(-4,-t_stat),fill="blue",alpha = 0.4)+
        geom_area(stat = "function", fun = dt,args = list(df=df),
                xlim = c(-t_stat, t_stat),fill="grey",alpha = 0.2) +
        geom_area(stat = "function", fun = dt,args = list(df=df),
                xlim = c(t_stat, 4),fill="blue",alpha = 0.4)+
        geom_vline(xintercept = t_stat, color = "red")+
  geom_text(aes(x = t_stat,
              label = paste0("Test statistic = ", round(t_stat, 2)), y = 0.2),
          colour = "red", angle = 90, vjust = 1.3, size = 6) +
  geom_text(aes(x = t_stat,
              label = paste0("p-value = ", round(pvalue, 4)), y = 0.05),
      size = 6)+
     ylab("Density")
```

Visualizing Statistical Hypothesis Testing

- We may use `ggplot2` to visualize the test statistic and p-value



Visualizing Statistical Hypothesis Testing

Reading: another example from the infer vignette.

License