Contents

Testing Figure Numbering

Exploratory data analysis for quantitative variables

Here, we'll run a fuller range of *summary* or *descriptive* statistics.

Quantitative variables and descriptive statistics

Descriptive statistics are numeric summaries of a dataset. In fact, frequencies (which we covered in the last section) are one type of descriptive statistic: they tell us how many of each unique value there are.

Let's walk through what some of these statistics are, then how to calculate them in Excel.

Range

The range of a dataset is the difference between the minimum and maximum value. This measure is easiest to calculate if your data is sorted from low to high:

Fig. 1 Here is my figure caption!

Variable types

We know that variables vary across observations. But variables can also be quite different from each other. Even in our relatively small dataset, we have text, numbers, and *yes/no* results all as variables.

It can help to classify these variable types, as these distinctions will be important as we continue our analysis.

Keep in mind that these distinctions are somewhat arbitrary and may change based on the purpose of our analysis and the size of our datasets. You will see that EDA, and analytics in general, is highly iterative.

=-=-=-=-=-=-=-=-=

Fig. 2 Here is my figure caption!

Classifying variables is somewhat arbitrary and, like much of analytics, built on rules of thumb rather than hard-and-fast-criteria.

=-=-=-=-=-=-=-=-=

We will discuss the variable types as shown on this tree diagram.

Fig. 3 Here is my figure caption!

Variance

This gives us the spread of the *entire* dataset without much context for how different each observation is from some point of reference.

A good point of reference is the mean, and *variance* tells us how far away observations fall from it.

This measure is a bit more intensive to calculate than those we've covered this far. Our steps will be:

1. Find the mean of our dataset.

2. Subtract the mean from each observation. This is the *deviation*.
3. Take the sum of the squares of each deviation
4. Divide the sum of square of each deviation by the number of observations.

That's a lot of steps! For operations this intensive, it can be helpful to use mathematical notation. I know they can some getting used to and be very intimidating at first, but consider the alternative of the written list above!

We can express the operations needed to find the variance like so:

$$s^2 = \frac{\sum(X-\bar{X})^2}{N}$$

s^2 is our variance. $(X - \bar{X})^2$ tells us that we need to subtract each observation X from the average $\bar{X}$, and square it. $\sum$ tells us to sum those results .

Finally, we will divide that by the number of observations N.

Test equation with tag

$$\$a := x^2 - y^3\$ \quad \text{(eqn)}$$

Test equation with label $\$a := x^2 - y^3\$$

We'll use mathematical notation a few more times in the book. I'll make sure to label and walk through each equation, but I hope you can see that with practice, these actually become easier to read than a written list of steps.

Testing R exercises

```
library(tidyverse)
```

```
-- [1mAttaching packages [22m --------------------------------------- tidyverse 1.3.0 --

 [32mv [39m [34mggplot2 [39m 3.3.2      [32mv [39m [34mpurrr   [39m 0.3.4
 [32mv [39m [34mtibble  [39m 3.0.3      [32mv [39m [34mdplyr   [39m 1.0.2
 [32mv [39m [34mtidyr   [39m 1.1.2      [32mv [39m [34mstringr [39m 1.4.0
 [32mv [39m [34mreadr   [39m 1.3.1      [32mv [39m [34mforcats [39m 0.5.0

-- [1mConflicts [22m ------------------------------------------ tidyverse_conflicts() --
 [31mx [39m [34mdplyr [39m:: [32mfilter() [39m masks [34mstats [39m::filter()
 [31mx [39m [34mdplyr [39m:: [32mlag() [39m    masks [34mstats [39m::lag()
```

```
data(iris)
head(iris)
```

Fig. 4 Here is my figure caption!

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

A data.frame: 6 × 5

```
fit1 <- lm(Sepal.Length ~ Petal.Width, data = iris)
summary(fit1)
```

```
Call:
lm(formula = Sepal.Length ~ Petal.Width, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.38822 -0.29358 -0.04393  0.26429  1.34521

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.77763    0.07293   65.51   <2e-16 ***
Petal.Width  0.88858    0.05137   17.30   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.478 on 148 degrees of freedom
Multiple R-squared:  0.669,     Adjusted R-squared:  0.6668
F-statistic: 299.2 on 1 and 148 DF,  p-value: < 2.2e-16
```

library(ggplot2)

ggplot(iris, aes(x = Petal.Width, y = Sepal.Length)) + geom_point() + stat_smooth(method = "lm", col = "red")

```
library(readxl)
housing <- read_xlsx("housing.xlsx")
head(housing)
```

price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea	neighborhood
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>
42000	5850	3	1	2	yes	no	yes	no	no	1	no	Blmngtn
38500	4000	2	1	1	yes	no	no	no	no	0	no	SawyerW
49500	3060	3	1	1	yes	no	no	no	no	0	no	SawyerW
60500	6650	3	1	2	yes	yes	no	no	no	0	no	NWAmes
61000	6360	2	1	1	yes	no	no	no	no	0	no	Blmngtn
66000	4160	3	1	1	yes	yes	yes	no	yes	0	no	Gilbert

A tibble: 6 × 13

```
t.test(price ~ airco, data=housing)
```

```
        Welch Two Sample t-test

data:  price by airco
t = -10.699, df = 265.03, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -30779.85 -21211.62
sample estimates:
 mean in group no mean in group yes
         59884.85          85880.59
```

```
# Overlaid histograms
ggplot(housing, aes(x=price, fill=airco)) +
  geom_histogram()
```

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


```
# Overlaid histograms
ggplot(housing, aes(x=price,fill=airco)) +
  geom_histogram()
```

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


```
ggplot(housing, aes(x=airco, y=price)) +
  geom_boxplot()
```

```
head(mpg)
```

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact

A tibble: 6 × 11

```
names(mpg)
```

'manufacturer' · 'model' · 'displ' · 'year' · 'cyl' · 'trans' · 'drv' · 'cty' · 'hwy' · 'fl' · 'class'

```
# Scatterplot
ggplot(data=mpg, aes(x=hwy, y=displ)) +
    geom_point()
```

```
mpg_reg <- lm(displ ~ hwy, data=mpg)
summary(mpg_reg)
```

```
Call:
lm(formula = displ ~ hwy, data = mpg)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4126 -0.5710 -0.1105  0.4571  3.6212

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.367570   0.221422   33.27   <2e-16 ***
hwy         -0.166201   0.009157  -18.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8323 on 232 degrees of freedom
Multiple R-squared:  0.5868,    Adjusted R-squared:  0.585
F-statistic: 329.5 on 1 and 232 DF,  p-value: < 2.2e-16
```

```
# Add regression line
ggplot(data=mpg, aes(x=hwy, y=displ)) +
    geom_point() +
    geom_smooth(method=lm)
```

```
`geom_smooth()` using formula 'y ~ x'
```

Testing Python exercises

1. Assign the sum of -10 and 2 to `a`.
2. Assign the absolute value of `a` to `b`.
3. Assign `b` minus 1 as `d`.
4. Print the result of `d`. What is the value? What type is this variable?

```
a = -10 + 2
b = abs(a)
d = b - 1

print(d)
print(type(d))
```

```
7
<class 'int'>
```

Fig. 5 Here is my figure caption!

Section in chapter shows up in toc

1. Create a list containing the values `North`, `East`, `South` and `West`.
2. What is the result of the below?

```
len(['Monday','Tuesday','Wednesday','Thursday','Friday',['Saturday','Sunday']])
```

```
directions = ['North','East','South','West']
print(directions)

print(len(['Monday','Tuesday','Wednesday','Thursday','Friday',['Saturday','Sunday']]))
```

```
['North', 'East', 'South', 'West']
6
```

DRILL 3

1. What do you expect to be the result of the following? Run the code and see how you did.

```
my_week = (['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
my_week.sort()
print(my_week)
```

1. Pass the `clear()` method to `my_week` from above. Re-print `my_week`. What happens?

```
my_week = (['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
my_week.sort()
print(my_week)
```

```
['Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday', 'Wednesday']
```

```
my_week.clear()
print(my_week)
```

```
[]
```

Drill little 1

Practice some more slicing below:

```
my_list = [7,12,5,10,9]

# Get the first through third elements
print(my_list[0:3])

# Get the third-last to second-last elements
print(my_list[-3:-1])

# Get the second through last elements
print(my_list[1:5])
```

```
[7, 12, 5]
[5, 10]
[12, 5, 10, 9]
```

DRILL little 2

Practice slicing lists below.

These operations will work the same regardless of whether your list contains floats, strings, or other data types.

```
this_list = ["Slicing","works","on","lists","of","strings","identically"]

# Get the third to final elements
print(this_list[2:])


# Get everything up to the fourth element
print(this_list[:4])


# Get everything starting with the second-last element
print(this_list[-2:])
```

```
['on', 'lists', 'of', 'strings', 'identically']
['Slicing', 'works', 'on', 'lists']
['strings', 'identically']
```

Drill little 3

The `factorial()` function from `math` will take the factorial of a number `x`.

Find the factorial of 10 using this function.

```
import math

math.factorial(10)
```

```
3628800
```

Drill 4

Install the `seaborn` package.

```
#!pip install seaborn
```
