Data science and analysis in Neuroscience

Kevin Allen

December 3, 2020

Today's plan

1. Combining information from several data frames
2. Reshaping data frames
3. Common statistical procedures with R

- Linear correlation
- t-test
- Common errors in statistics

Combining information from several data frames

Load the tmaze data set for a few more exercises.

```
myFile="~/repo/dataNeuroCourse/dataSets/tmaze.csv"
df<-read_csv(myFile)
df<-mutate(df, correct = sample != choice)
print(df,n=5)
```

```
## # A tibble: 1,120 x 8
##   mouse  date       injection block trialNo sample choice correct
##   <chr>  <date>     <chr>     <dbl>   <dbl> <chr>  <chr>  <lgl>
## 1 Mn4656 2019-10-09 Saline        1       1 L      R      TRUE
## 2 Mn4656 2019-10-09 Saline        1       2 L      L      FALSE
## 3 Mn4656 2019-10-09 Saline        1       3 R      R      FALSE
## 4 Mn4656 2019-10-09 Saline        1       4 L      R      TRUE
## 5 Mn4656 2019-10-09 Saline        1       5 R      L      TRUE
## # … with 1,115 more rows
```

Combining information from several data frames

In most projects, you need to work with several tables.

Relations are defined between a pair of tables.

`dplyr` has several **join** functions to do this.

Combining information from several data frames

```
dfGeno <- tibble(mouse=c("Mn4656","Mn848","Mn4672","Mn4673",
                         "Mn7712","Mn7735","Mn829"),
           genotype=c("wt","wt","wt","wt",
                      "ko","ko","ko"))
dfGeno
```

```
## # A tibble: 7 x 2
##   mouse  genotype
##   <chr>  <chr>
## 1 Mn4656 wt
## 2 Mn848  wt
## 3 Mn4672 wt
## 4 Mn4673 wt
## 5 Mn7712 ko
## 6 Mn7735 ko
## 7 Mn829  ko
```

How is `df` related to `dfGeno`?

```
colnames(df)
```

```
## [1] "mouse"     "date"      "injection" "block"     "trialNo"   "sample"
## [7] "choice"    "correct"
```

```
colnames(dfGeno)
```

```
## [1] "mouse"    "genotype"
```

```
colnames(dfGeno)[colnames(dfGeno) %in% colnames(df)]
```

```
## [1] "mouse"
```

`mouse` is a **key**, a variable that connect a pair of tables.

- A **primary key** uniquely identifies an observation in its table.

```
dfGeno %>%
  count(mouse)
```

```
## # A tibble: 7 x 2
##   mouse      n
##   <chr>  <int>
## 1 Mn4656     1
## 2 Mn4672     1
## 3 Mn4673     1
## 4 Mn7712     1
## 5 Mn7735     1
## 6 Mn829      1
## 7 Mn848      1
```

- A **foreign key** uniquely identifies an observation in *another* table.

```
df %>%
  count(mouse)
```

```
## # A tibble: 7 x 2
##   mouse      n
##   <chr>  <int>
## 1 Mn4656   160
## 2 Mn4672   160
## 3 Mn4673   160
## 4 Mn7712   160
## 5 Mn7735   160
## 6 Mn829    160
## 7 Mn848    160
```

Mutating joins

- It first matches observations by their keys.
- Then copies across variables from one table to the other.

```
df_join <- df %>%
  left_join(dfGeno,by="mouse") # match with mouse
print(df_join,n=6)
```

```
## # A tibble: 1,120 x 9
##   mouse  date       injection block trialNo sample choice correct genotype
##   <chr>  <date>     <chr>     <dbl>   <dbl> <chr>  <chr>  <lgl>   <chr>
## 1 Mn4656 2019-10-09 Saline        1       1 L      R      TRUE    wt
## 2 Mn4656 2019-10-09 Saline        1       2 L      L      FALSE   wt
## 3 Mn4656 2019-10-09 Saline        1       3 R      R      FALSE   wt
## 4 Mn4656 2019-10-09 Saline        1       4 L      R      TRUE    wt
## 5 Mn4656 2019-10-09 Saline        1       5 R      L      TRUE    wt
## 6 Mn4656 2019-10-09 Saline        1       6 R      R      FALSE   wt
## # … with 1,114 more rows
```

Want to know more

Chapter 10, Relational data and dplyr

Tidy dataframes

The golden rules of **tidy** data frames

1. Each variable must have its own column.
2. Each observation must have its own row.

Advantages

1. You only need to learn how to process one type of data frame
2. dplyr and ggplot are designed to work with tidy data.

Tidy dataframes

You will eventually encounter data sets that are not tidy.

What is wrong with this data frame?

```
df <- tibble(country = c("Afghanistan","Brazil","China"),
       "1999" = c(745, 37737, 80488),
       "2000" = c(2666, 80488,213766))
df
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China        80488 213766
```

Tidy dataframes

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
##   <chr>        <dbl>  <dbl>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China        80488 213766
```

Some of the columns are not variables but values of a variable.

Year is a variable. 1999 and 2000 are values of the variable year.

Reshaping data frames into tidy data frames

We need to gather these columns into a new pair of variables.

```
df %>% gather('1999','2000', key = 'year', value = 'cases')
```

```
## # A tibble: 6 x 3
##   country     year   cases
##   <chr>       <chr>  <dbl>
## 1 Afghanistan 1999     745
## 2 Brazil      1999   37737
## 3 China       1999   80488
## 4 Afghanistan 2000    2666
## 5 Brazil      2000   80488
## 6 China       2000  213766
```

Now you can use `dplyr::group_by` and `dplyr::summarize` for example

Reshaping data frames into tidy data frames

Chapter 9, Tidy Data with tidyr

There is a example with a data set from the World Health Organization.

Common statistics with R

1. Linear correlation
2. t-test
3. Common errors in statistics

Why do we need statistical tests

When can we say that our results are not due to chance?

```
n=10
x <- rnorm (n = n, mean = 0, sd = 2)
y <- rnorm (n = n, mean = 0, sd = 2)
head(x,n=5)
```

```
## [1]  0.266709  2.750443  1.497430 -2.587701 -1.117542
```

```
head(y,n=5)
```

```
## [1]  3.8251055 -1.5376220  1.8157063 -0.9728810  0.6332008
```

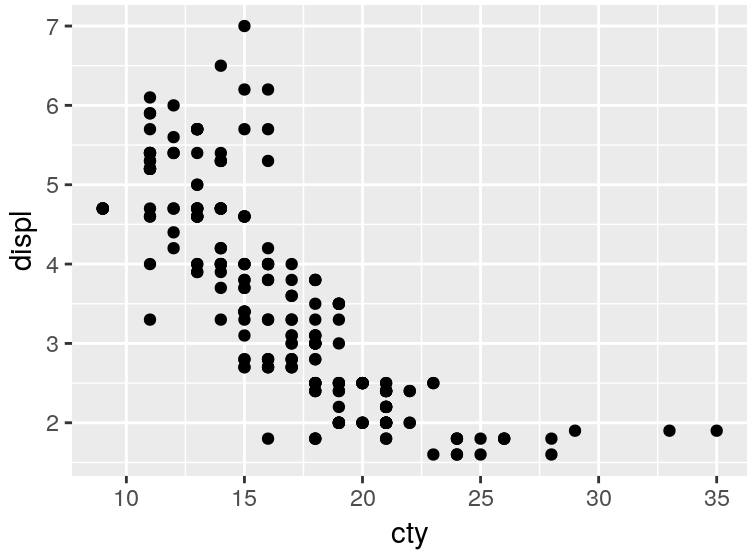
```
print(paste("Means :", round(mean(x),3),", ", round(mean(y),3)))
```

```
## [1] "Means : -0.626 ,  0.705"
```

General tip for statistics

Always plot the data. Don’t do blind statistical testing.

```
ggplot(data=mpg)+
  geom_point(mapping = aes(x=cty,y=displ))
```



Prevent the “garbage in, garbage out” effect.

Input data

Functions calculating statistics most often require **numeric vectors** as input.

If you have a data frame, you may need to extract a single column.

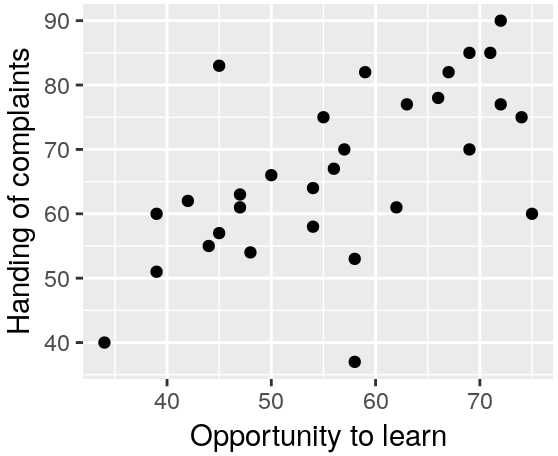
```
# classic R
mpg$displ
# or tidyverse style
mpg %>% pull(displ)
```

Linear correlation (Pearson)

Look for a linear relationships between two variables (x and y).

The typical plot for correlation analysis is a scatter plot.

```
ggplot(data=attitude)+
  geom_point(mapping = aes(x=learning,y=complaints)) +
  xlab("Opportunity to learn") +
  ylab("Handing of complaints")
```



Linear correlation

1. We will do an example **manually**
2. Then an example with the cor.test() function.
3. Then you will have a go.

Linear correlation

Let's generate data set to work with.

We will do an example *manually* to remove most of the magic from it.

```
n = 40
# x = random numbers from a gaussian distribution
x <- rnorm (n = n, mean = 0, sd = 1)
y <- 2 * x + 50 + rnorm( n = n, mean = 0, sd = 2.0)
df <- data.frame(x = x, y = y)
head(df,n=5)
```

```
##           x        y
## 1  0.2832910 53.27526
## 2  1.7477398 52.43416
## 3  1.1973573 53.86935
## 4 -0.3359747 44.42426
## 5  0.8973318 51.80975
```

It is often useful when learning to work with simulated data.

Linear correlation

Correlation coefficient (r).

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

r varies from -1 to 1.

Negative r values represent negative slopes.

1 or -1 is a perfect linear relationship.

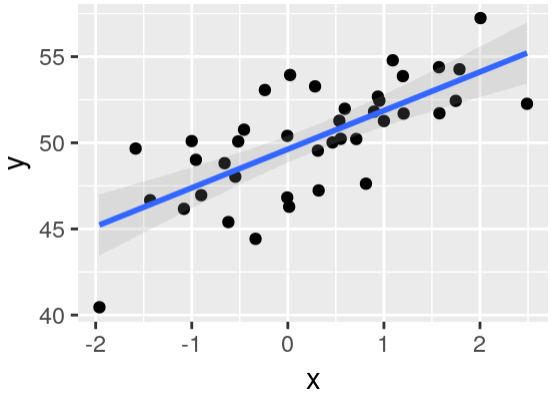
Values near 0 mean very poor correlations.

Linear correlation

```
r_value = sum( (x-mean(x)) * (y - mean(y))) /
  sqrt( sum((x-mean(x))^2) * sum((y-mean(y))^2))
print(paste("My r value:", round(r_value,2)))
```

```
## [1] "My r value: 0.71"
```

```
df %>% ggplot()+
  geom_point(mapping = aes(x=x,y=y))+
  geom_smooth(mapping = aes(x=x,y=y),method = "lm",alpha=0.2)
```



Linear correlation, degrees of freedom

The number of independent values that can vary in an analysis without breaking any constraints.

For a linear correlation: $df = n - 2$

Nowadays, I use the value of df to make sure the computer is doing what I think it is doing.

Linear correlation, p value

We now need to decide whether there is a significant relationship between x and y.

The p value tells you the probability of obtaining a specific r value (0.71) by chance.

If it is less than 5%, we claim that there is a significant relationship between x and y.

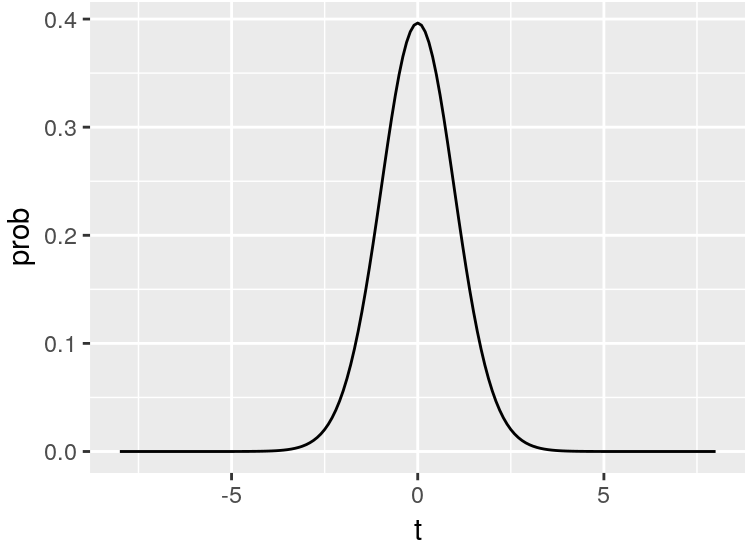
Notice that you will get this wrong once every 20 attempts.

Linear correlation, p value

$$t = \frac{r}{\sqrt{1-r^2}}\sqrt{n-2}$$

```
t_value = r_value * sqrt(n-2) / sqrt(1-r_value^2)
print(paste("t:",round(t_value,4),
            ", p:", round(1-pt(t_value,df = n-2),4)))
```

```
## [1] "t: 6.263 , p: 0"
```

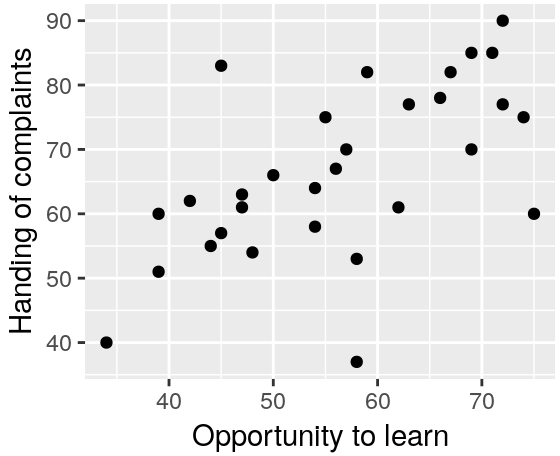


Example 1

We are done with manually calculating r, t and p. Let's see how you normally do this in R.

We will use a data set (attitude) available within R.

```
ggplot(attitude) +
  geom_point(mapping = aes(x=learning,y=complaints)) +
  xlab("Opportunity to learn") +
  ylab("Handing of complaints")
```



Example 1

All the calculation is done by `cor.test()`.

```
cor.test(attitude$learning,attitude$complaints)
```

```
##
##  Pearson's product-moment correlation
##
## data:  attitude$learning and attitude$complaints
## t = 3.935, df = 28, p-value = 5e-04
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3012258 0.7876678
## sample estimates:
##       cor
## 0.5967358
```

If $p < 0.05$, there is a linear relationship between the two variables.

Example 1

You can also save the results for later use.

```
res <- cor.test(attitude$learning,attitude$complaints)
print(paste("r:", res$estimate, res$estimate^2))
```

```
## [1] "r: 0.596735806267436 0.356093622481647"
```

```
print(paste("t:",res$statistic))
```

```
## [1] "t: 3.93504542629823"
```

```
print(paste("p:",res$p.value))
```

```
## [1] "p: 0.00050016995186579"
```

Example 2

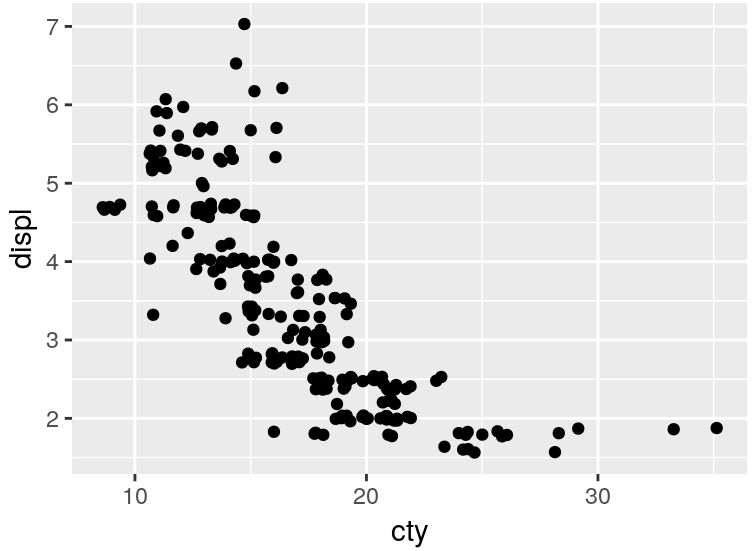
Using the `mpg` data frame, test whether there is a linear relationship between the engine displacement and the city miles per gallon.

Tip: use `?mpg` if you want a reminder of what is in this data frame.

You have 4 minutes to complete this task.

Example 2: plot

```
ggplot(data=mpg)+
  geom_point(mapping = aes(x=cty,y=displ),position="jitter")
```



Example 2: statistics

```
cor.test(mpg$displ,mpg$cty)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mpg$displ and mpg$cty
## t = -20.205, df = 232, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8406782 -0.7467508
## sample estimates:
##       cor
## -0.798524
```

Negative correlation coefficients are for negative slopes.

Assumptions for linear correlations

- The relationship is linear
- The data (x and y) are normally distributed (see shapiro.test()).
- The data points are independent of each other (not from same subject).

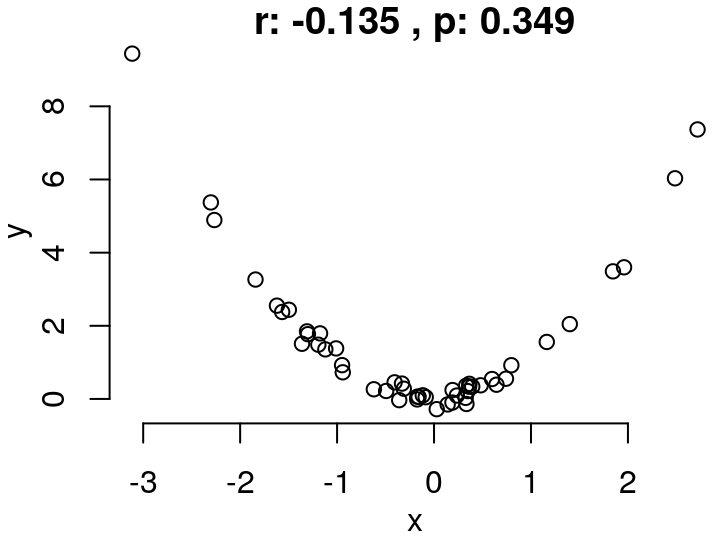
The p value has no real meaning if the assumptions of the test are violated.

Alternative: **Spearman** or **Kendall** rank correlation (method = c("pearson", "kendall", "spearman"))

Linear correlations: possible pitfalls

Things can go wrong with non-linear relationships

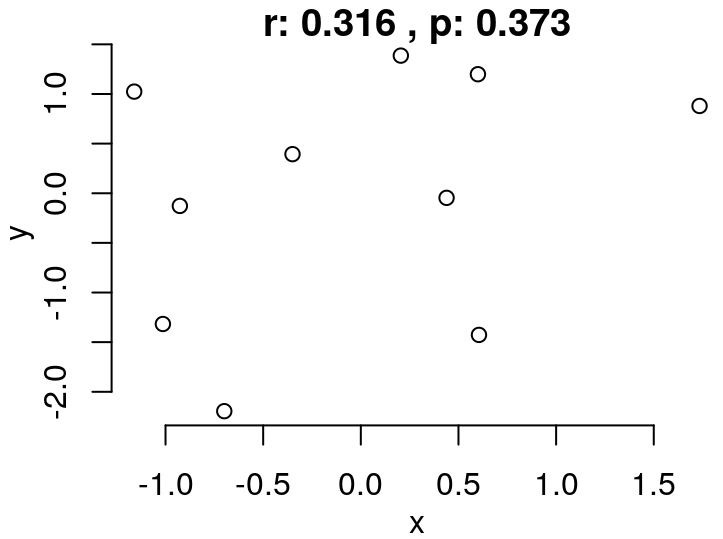
```
n=50
x <- rnorm (n = n, mean = 0, sd = 1)
y <- x^2 + rnorm(n = n, mean =0, sd = 0.2)
res <- cor.test(x,y)
plot(x,y,main=paste("r:",round(res$estimate,3),
                    ", p:",round(res$p.value,3)))
```



Linear correlations: possible pitfalls

Random data

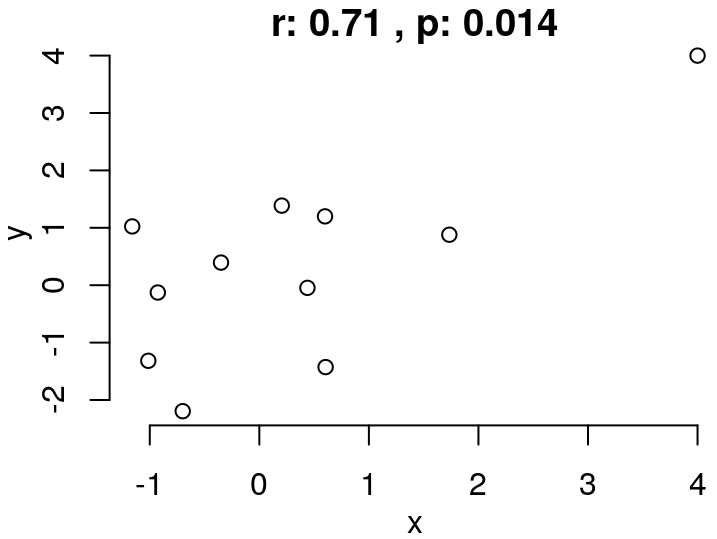
```
x <- rnorm (n = n, mean = 0, sd = 1)
y <- rnorm (n = n, mean = 0, sd = 1)
res <- cor.test(x,y)
plot(x,y,main=paste("r:",round(res$estimate,3),
                    ", p:",round(res$p.value,3)))
```



Linear correlations: possible pitfalls

Random data and **one outlier**

```
x <-c(x, 4)
y <-c(y, 4)
res <- cor.test(x,y)
plot(x,y,main=paste("r:",round(res$estimate,3),
                    ", p:",round(res$p.value,3)))
```



t-test

Test if there is a difference between

- the **mean** of one group to a known value (one-sample t test)
- the **means** of **2** groups (two-sample t test)
    - 2 independent groups
    - 2 dependent groups

Fun fact: Developed by William Sealy Gosset who worked as Head Brewer at Guinness in Dublin. He published under the pseudonym of “Student”.

One-sample t test

We want to test whether some values are significantly different from a value (e.g., 0).

More precisely, we want to know whether our data are sampled from a normal distribution with a specific mean.

Assumes that the data are normally distributed.

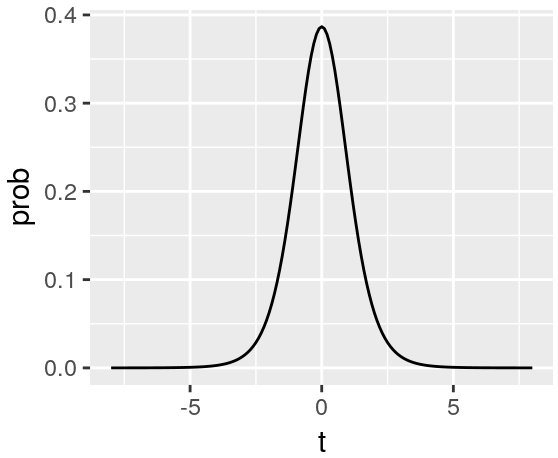
Degrees of freedom, $df = N - 1$

One-sample t test

$$t = \frac{\bar{X} - \mu}{s/\sqrt{N}}$$

where $\bar{X}$ is the mean of your sample, μ is the mean of the population, s is the standard deviation of your sample.

With t and your degrees of freedom, you can find the probability (p value) using the t distribution.



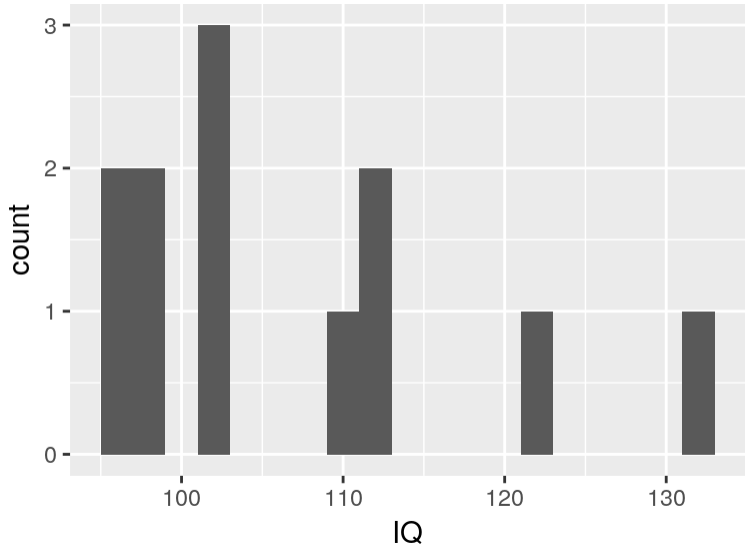
One-sample t test example

We want to know whether the IQ of a group of undergraduate students is different than that of the general population (100). We obtain scores from 12 students and perform a t-test.

One-sample t test example

Don’t forget to look at your data

```
# data entries
gs <- data.frame(IQ = c(103,123,95,132,113,102,98,97,110,102,112,98))
ggplot(data = gs) +
  geom_histogram(mapping = aes(x=IQ),binwidth = 2)
```



One-sample t test example

Test for normality with the Shapiro-Wilk test.

If p < 0.05, then your distribution is not normal.

```
shapiro.test(gs$IQ)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  gs$IQ
## W = 0.88139, p-value = 0.09131
```

One-sample t test example

Use the `t.test()` function to perform the t test.

```
t.test(x = gs$IQ,
       alternative = "two.sided",
       mu = 100)
```

```
##
##  One Sample t-test
##
## data:  gs$IQ
## t = 2.1574, df = 11, p-value = 0.05395
## alternative hypothesis: true mean is not equal to 100
## 95 percent confidence interval:
##   99.85697 114.30970
## sample estimates:
## mean of x
##  107.0833
```

Two-sample t test

Comparing the means of **2** groups.

- Two independent groups (different cells or subjects)
- Two dependent groups (same cells or subjects tested twice)

t test for independent samples

We measure the firing rate of the two types of neurons (pyramidal cells and interneurons) and want to know if their firing rates differs.

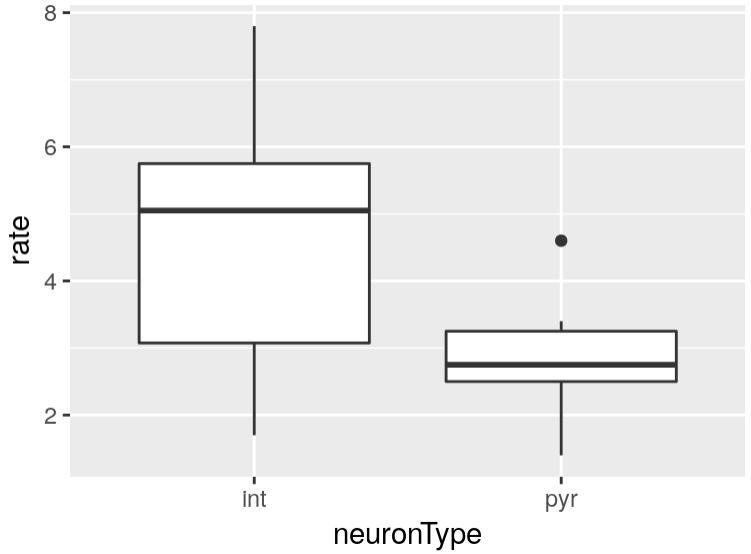
```
neurons <- tibble(neuronType = c(rep("pyr",8),rep("int",8)),
  rate = c(2.5,1.4,3.2,2.5,3.4,2.5,4.6,3.0,5.2,1.7,6.2,2.7,7.8,4.9,5.6,3.2))
head(neurons,n=5)
```

```
## # A tibble: 5 x 2
##   neuronType  rate
##   <chr>      <dbl>
## 1 pyr          2.5
## 2 pyr          1.4
## 3 pyr          3.2
## 4 pyr          2.5
## 5 pyr          3.4
```

t test for independent samples

We should always plot our data. For t-tests, a boxplot is a good idea.

```
neurons %>% ggplot()+
  geom_boxplot(mapping = aes(x = neuronType,y = rate))
```



t test for independent samples

Use 2 numerical vectors as inputs.

```
pyr <- neurons %>% filter(neuronType=="pyr") %>% select(rate) %>% pull()
int <- neurons %>% filter(neuronType=="int") %>% select(rate) %>% pull()
t.test(x = pyr, y = int,
       alternative = "two.sided", paired = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  pyr and int
## t = -2.2724, df = 9.8363, p-value = 0.0468
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.51935805 -0.03064195
## sample estimates:
## mean of x mean of y
##    2.8875    4.6625
```

t test for independent samples

Use the arguments `data` and `formula`. Simpler with tidy data sets.

```
t.test(formula = rate~neuronType,
       alternative = "two.sided", paired = FALSE, data = neurons)
```

```
##
##  Welch Two Sample t-test
##
## data:  rate by neuronType
## t = 2.2724, df = 9.8363, p-value = 0.0468
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.03064195 3.51935805
## sample estimates:
## mean in group int mean in group pyr
##            4.6625            2.8875
```

t test for independent samples

Assumptions

- The two distributions have the same variance
- The two groups are independent
- Normality

`t.test()` by default will take into consideration the unequal sample size, like in our example.

Alternative for data that are not normally distributed: Wilcoxon Rank Sum Tests `wilcox.test()`.

t test for dependent samples

We measure the firing rate of the **same** neurons in two conditions (t0 and t1) and want to know if the firing rate changed.

```
neurons <- data.frame(t0 = c(2.5,1.4,3.2,2.5,3.4,2.5,4.6,3.0),
          t1 = c(2.7,1.7,3.2,2.7,3.8,2.9,4.6,3.2))
neurons
```

```
##    t0  t1
## 1 2.5 2.7
## 2 1.4 1.7
## 3 3.2 3.2
## 4 2.5 2.7
## 5 3.4 3.8
## 6 2.5 2.9
## 7 4.6 4.6
## 8 3.0 3.2
```

Anything wrong with this data frame?

```
##    t0  t1
## 1 2.5 2.7
## 2 1.4 1.7
## 3 3.2 3.2
## 4 2.5 2.7
## 5 3.4 3.8
## 6 2.5 2.9
## 7 4.6 4.6
## 8 3.0 3.2
```

Anything wrong with this data frame?

```
##    t0  t1
## 1 2.5 2.7
## 2 1.4 1.7
## 3 3.2 3.2
## 4 2.5 2.7
## 5 3.4 3.8
## 6 2.5 2.9
## 7 4.6 4.6
## 8 3.0 3.2
```

It has several observations per row. Harder to use ggplot.

Reshape for ggplot

Here we reshape the data set.

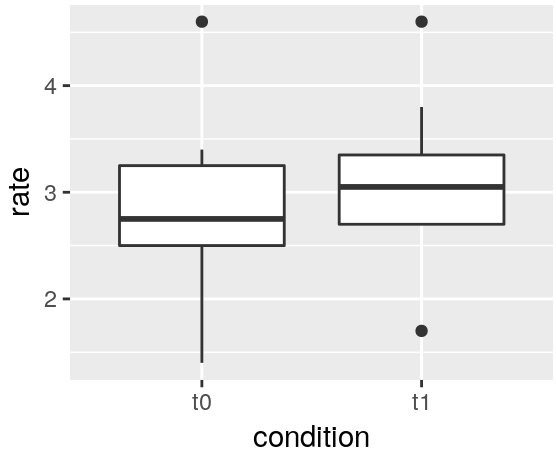
```
neurons_long <- neurons %>%
  gather(key="condition", value = "rate", starts_with("t"))
neurons_long
```

```
##    condition rate
## 1         t0  2.5
## 2         t0  1.4
## 3         t0  3.2
## 4         t0  2.5
## 5         t0  3.4
## 6         t0  2.5
## 7         t0  4.6
## 8         t0  3.0
## 9         t1  2.7
## 10        t1  1.7
## 11        t1  3.2
## 12        t1  2.7
## 13        t1  3.8
## 14        t1  2.9
```

t test for dependent samples

Now it is easier to use ggplot.

```
ggplot(data=neurons_long)+
  geom_boxplot(mapping = aes(x = condition,y=rate))
```



t test for dependent samples

For paired t test, you need to vectors of equal length

```
t.test(x=neurons$t0, y=neurons$t1,
       alternative = "two.sided", paired = TRUE)
```

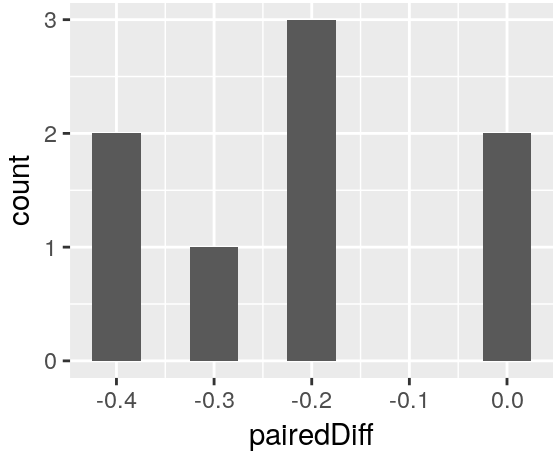
```
##
##  Paired t-test
##
## data:  neurons$t0 and neurons$t1
## t = -3.8711, df = 7, p-value = 0.006123
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.34230458 -0.08269542
## sample estimates:
## mean of the differences
##                 -0.2125
```

Significant results despite a very small effect (0.2125)

t test for dependent samples

The test works on the difference between t0 and t1.

```
df <- data.frame(pairedDiff = neurons$t0-neurons$t1)
df %>% ggplot()+
  geom_histogram(aes(x=pairedDiff),binwidth=0.05)
```



t test for dependent samples can detect smaller difference.

Reading for this week

Ten common statistical mistakes (eLife, 2019)

This article covers the most common statistical mistakes made by scientists. They are very common.

Learn to detect them and eliminate them from your work.