Computer Science

GRADUATE PROGRAMS

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

ILLINOIS COMPUTER SCIENCE

has a global reputation for developing revolutionary technology—where groundbreaking research addresses real-world problems.

EARN A PREMIER CS GRADUATE DEGREE

FROM A TOP-5 PROGRAM WITH RESEARCH IN:

Architecture, Compilers, and Parallel Computing

—

Artificial Intelligence

—

Bioinformatics and Computational Biology

—

Computers and Education

Data and Information Systems

—

Interactive Computing

—

Programming Languages, Formal Methods, and Software Engineering

Scientific Computing

—

Security and Privacy

—

Systems and Networking

—

Theory and Algorithms

COLLABORATIVE COMMUNITY

Learn from and work with some of the best CS faculty in the world.

Our internationally recognized faculty include 18 ACM Fellows, 20 IEEE Fellows, 9 Sloan Research Fellows, and 47 NSF CAREER Award winners.

CS FACTS & FIGURES

Ranked #5
U.S. News & World Report Graduate School Rankings

Birthplace of Mosaic Web Browser & LLVM Compiler Infrastructure

MS $119,978
PhD $138,536
Average Starting Salaries according to the Illini Success Survey

120+
Faculty

2,700+
Graduate Students

2,500+
Undergraduate Students

$33 Million
Research Expenditures in 2021

VIBRANT ECOSYSTEM

FOR INNOVATORS AND ENTREPRENEURS

Entrepreneurship is not just about startups, it is also a way of thinking and an approach to solving problems—a catalyst to inspiring the next generation of innovators. This mindset is an important part of our culture, is fully integrated across the curriculum, and is supported in and out of the classroom.

EXPLORE THE COMMUNITY:
entrepreneurship.illinois.edu

EXTRAORDINARY

QUALITY OF LIFE

ALL THE AMENITIES OF BIG-CITY LIVING WITHOUT THE HASSLES

Conveniently centered between Chicago, Indianapolis, and St. Louis, the University of Illinois provides an entertainment and cultural hub on par with the country's leading cities. Our campus community is home to Big 10 Division I sports; marquee theatrical, musical, dance performances; amazing festivals and fairs; and fantastic health and recreational facilities.

AN AFFORDABLE MICRO-URBAN ENVIRONMENT

The cost of living in Champaign-Urbana is 21.1% lower than the U.S. average; 214% cheaper than San Francisco, 116% cheaper than Seattle, 97% cheaper than Boston, and 40% cheaper than Atlanta. [Sperling's Best Places, 2022]

FUNDING OPPORTUNITIES

Our PhD students, with very few exceptions, are funded by a research or a teaching assistantship and/or by a fellowship. These provide a stipend and a full tuition waiver. Many of our MS students are also funded. Students may also qualify for fellowships offered by Illinois Computer Science, the College of Engineering at Illinois, the Graduate College, or apply to external funding agencies such as the NSF.

Computer Science

GRADUATE PROGRAM	ENTRY	APPLICATION DEADLINE(S)
PhD in Computer Science	Fall	December 15
MS in Computer Science	Fall	December 15
MS in Bioinformatics	Fall Spring	January 15 October 15
Professional Master of Computer Science (MCS)	Fall Spring	January 15 September 15
Professional Master of Computer Science (MCS) In Chicago	Fall Spring	April 15 October 31
Professional MCS and MCS in Data Science Online	Fall Spring Summer	May 30 October 15 February 15

Thomas M. Siebel Center for Computer Science
201 North Goodwin Avenue
Urbana, IL 61801-2302

Illinois Computer Science in Chicago
200 South Wacker Drive, 7th Floor
Chicago, IL 60606

OFFICE OF GRADUATE PROGRAMS

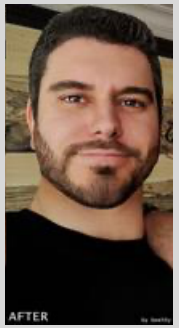
In person programs - academic@cs.illinois.edu
Online programs - online-mcs@cs.illinois.edu
go.cs.illinois.edu/GradPrograms

CONDUCTING AN INDEPENDENT-SAMPLES T-TEST

Example: **Comparing Attractiveness Ratings**

To test our hypothesis, we randomly assign 4 participants to view a filtered photo of someone and 4 participants to view an unfiltered photo of someone.

We then have them rate how attractive they think that person is on a scale from 1 to 10, with 10 being very attractive.



Filtered Photo Group		Unfiltered Photo Group	
Subject	**Attractiveness**	**Subject**	**Attractiveness**
1	9	5	5
2	8	6	6
3	6	7	5
4	9	8	4

Step 1: Check Assumptions

- **The distributions for each sample are normal** (no skew/outliers)

For this: We don't have a lot of data for this example, but in normal circumstances, you would **create a histogram of the data to check for normality**

- **The samples are independent of one another**
  - The participants in one condition are not the same participants in the other condition (there is no effect of one sample on the other)

For this: **We have to know how our sample was collected** and know that we controlled for confounding variables to ensure the two samples are collected from different participants

- **Homogeneity of variance**
  - The variance of the populations (represented by each sample) **are equal**—this means that the spread or variability in scores around the population mean is the same in both samples

For this: We need to run a **Levene's Test** (if our data is less or not normally distributed) or a **Barlett's Test** (if our data is close to or normally distributed) to test whether the variances are the same

Step 1: Check Assumptions

For this: We need to run a **Levene's Test** (if our data is less or not normally distributed) or a **Barlett's Test** (if our data is close to or normally distributed) to test whether the variances are the same

- **Homogeneity of variance**
  - The variance of the populations (represented by each sample) **are equal**—this means that the spread or variability in scores around the population mean is the same in both samples

Both **Levene's** and **Bartlett's Tests** *are* hypothesis tests. They test the null and alternative hypotheses that the variances are equal or not, respectively. (P.s., always two-tailed!)

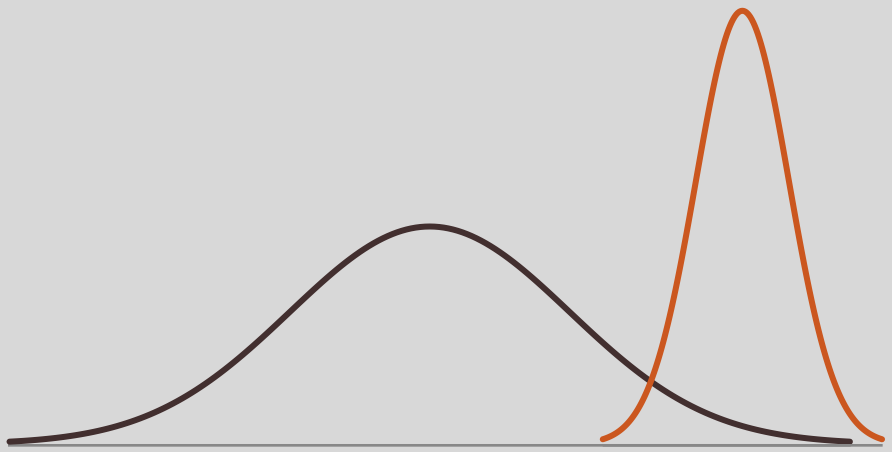
$$H_0: \sigma_1^2 = \sigma_2^2 = \ldots \sigma_x^2$$

$$H_a: \sigma_1^2 \neq \sigma_2^2 \neq \ldots \sigma_x^2$$

You will test whether you **fail to reject the null hypothesis (homogenous/equal variances)**

or

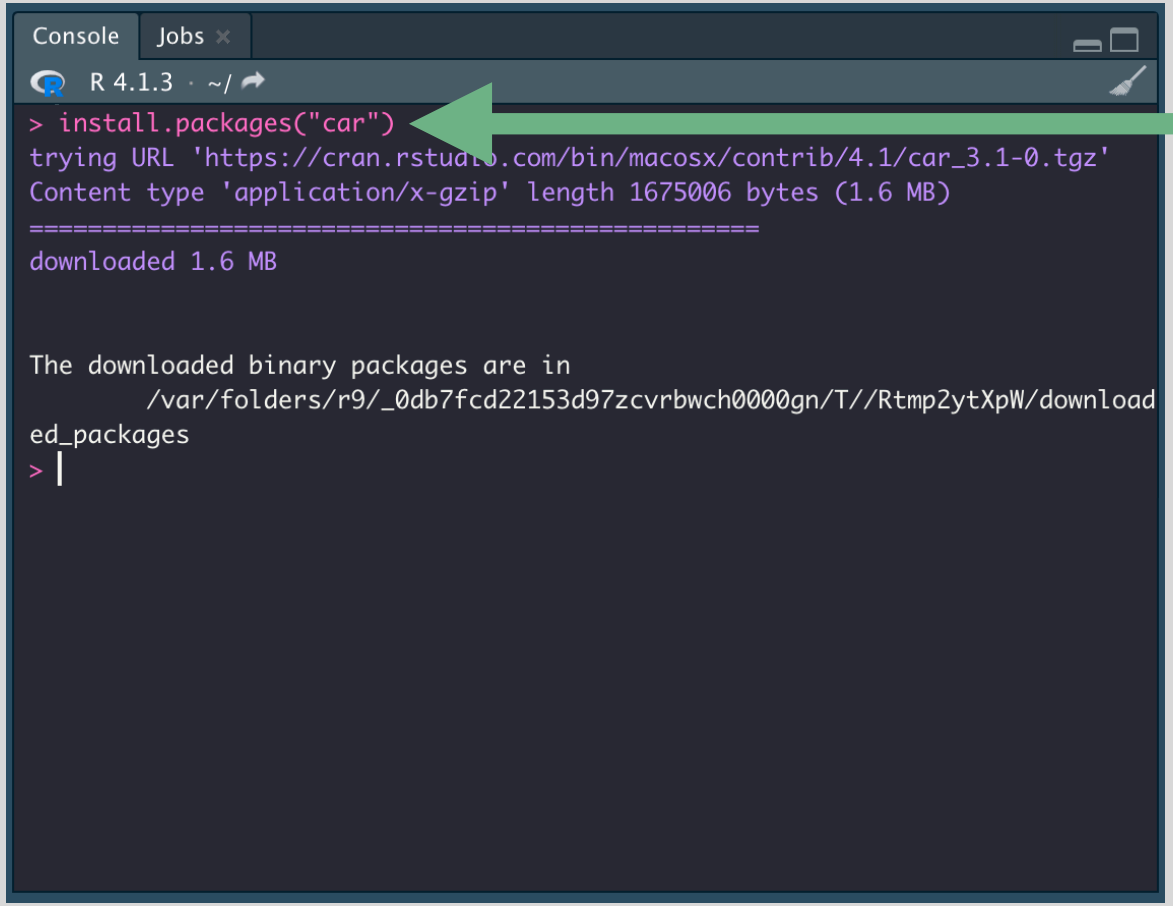
reject the null hypothesis (heterogeneous/unequal variances)



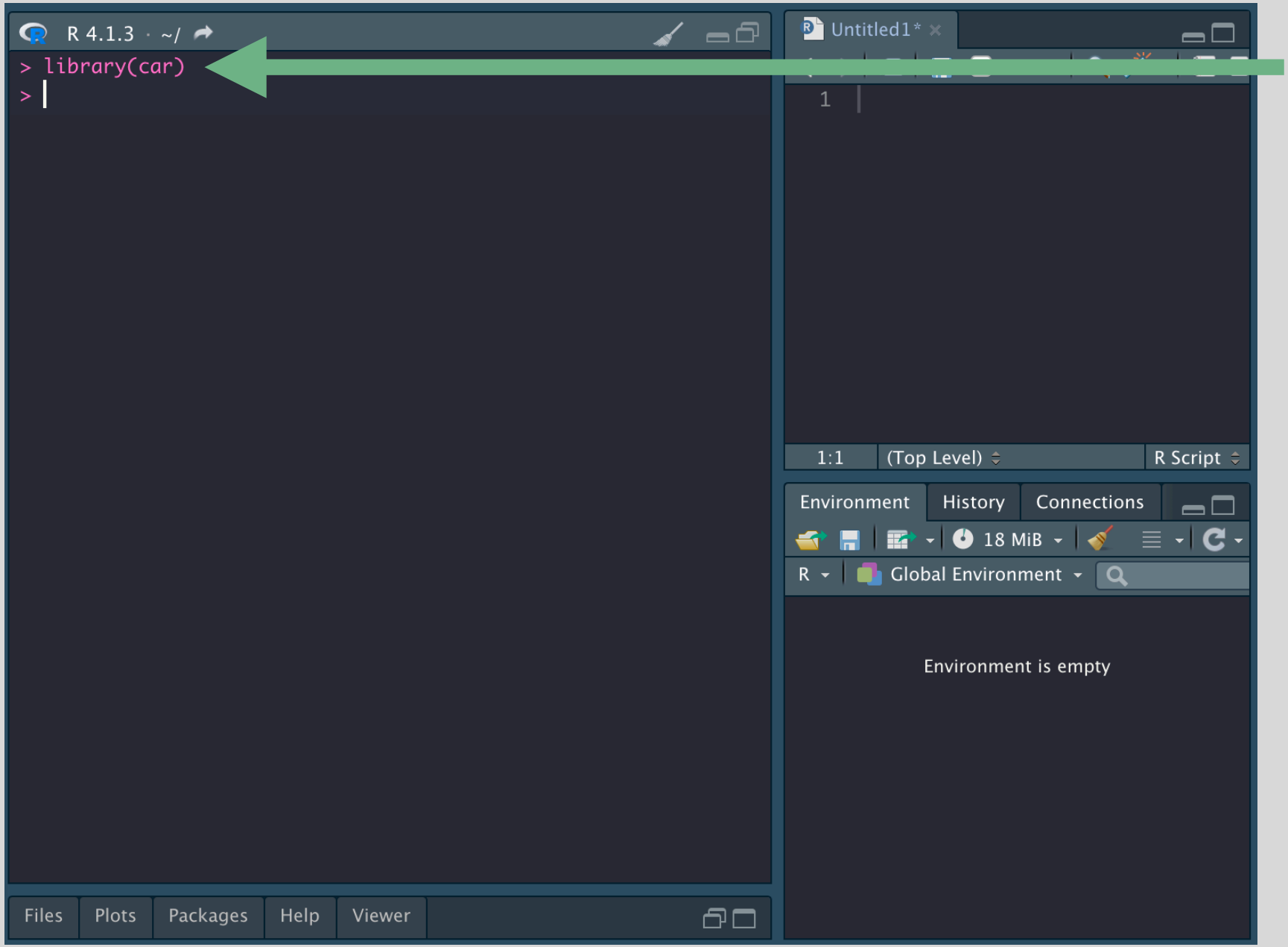
The variances of **Group 1** and **Group 2** are not equal if this were actually our distributions.

Step 1: Check Assumptions

For this: We need to run a **Levene's Test** (if our data is less or not normally distributed) or a **Barlett's Test** (if our data is close to or normally distributed) to test whether the variances are the same



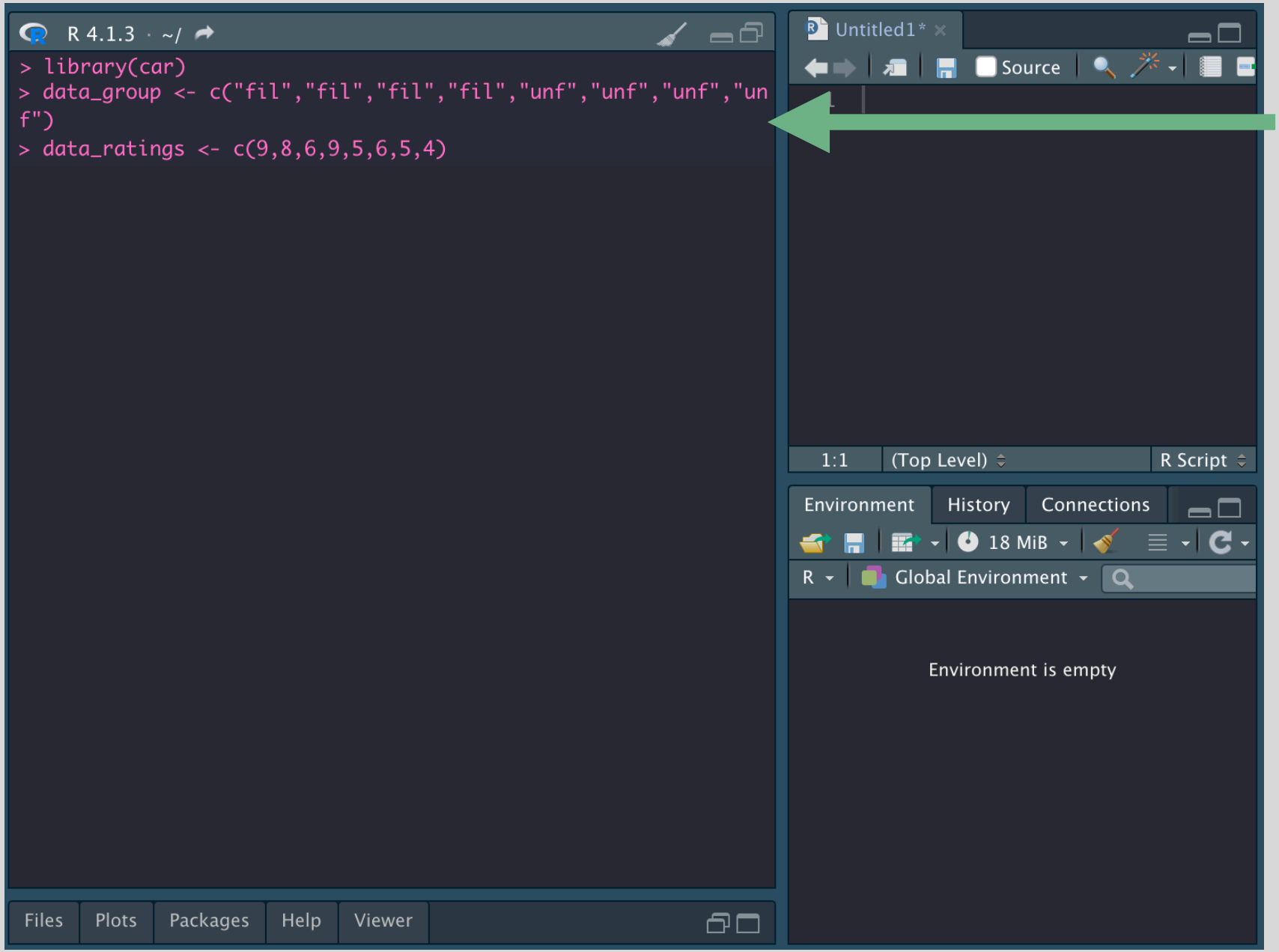
Install the **car** package in R.
Remember: You only ever need to install a package *once*.



Load the **car** package from your **library** in R. Remember: You need to do this *every* time you open RStudio and start coding a new script.

Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.



A note about **Wide** versus **Long** format for data files

Let's say we wanted to conduct a study on how happiness is affected by the weather/season. So, on four selected days over the course of one year, we asked people to rate their happiness on a scale from 1 *(really unhappy)* to 10 *(really happy)*.

Wide Format: A column for each level of the independent variable

Long Format: Multiple rows for each level of the independent variable

Wide Format

Rainy & Fresh Spring Day	Warm & Sunny Summer Day	Chilly & Brisk Fall Day	Cold & Snowy Winter Day
5	7	6	2
6	8	2	1
4	9	5	3
3	10	7	4

Long Format

Season	Happiness Rating
Spring	5
Spring	6
Spring	4
Spring	3
Summer	7
Summer	8
Summer	9
Summer	10
Fall	6
Fall	2
Fall	5
Fall	7
Winter	2
Winter	1
Winter	3
Winter	4

Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.

A note about **Wide** versus **Long** format for data files

Let's say we wanted to conduct a study on how happiness is affected by the weather/season. So, on four selected days over the course of one year, we asked people to rate their happiness on a scale from 1 *(really unhappy)* to 10 *(really happy)*.

Wide Format: A column for each level of the independent variable

Long Format: Multiple rows for each level of the independent variable

Wide Format

Rainy & Fresh Spring Day	Warm & Sunny Summer Day	Chilly & Brisk Fall Day	Cold & Snowy Winter Day
5	7	6	2
6	8	2	1
4	9	5	3
3	10	7	4

Long Format

Season	Happiness Rating
Spring	5
Spring	6
Spring	4
Spring	3
Summer	7
Summer	8
Summer	9
Summer	10
Fall	6
Fall	2
Fall	5
Fall	7
Winter	2
Winter	1
Winter	3
Winter	4

Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.

A note about **Wide** versus **Long** format for data files

Let's say we wanted to conduct a study on how happiness is affected by the weather/season. So, on four selected days over the course of one year, we asked people to rate their happiness on a scale from 1 *(really unhappy)* to 10 *(really happy)*.

Wide Format: A column for each level of the independent variable

Long Format: Multiple rows for each level of the independent variable

Wide Format

Rainy & Fresh Spring Day	Warm & Sunny Summer Day	Chilly & Brisk Fall Day	Cold & Snowy Winter Day
5	7	6	2
6	8	2	1
4	9	5	3
3	10	7	4

Long Format

Season	Happiness Rating
Spring	5
Spring	6
Spring	4
Spring	3
Summer	7
Summer	8
Summer	9
Summer	10
Fall	6
Fall	2
Fall	5
Fall	7
Winter	2
Winter	1
Winter	3
Winter	4

Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.

A note about **Wide** versus **Long** format for data files

Let's say we wanted to conduct a study on how happiness is affected by the weather/season. So, on four selected days over the course of one year, we asked people to rate their happiness on a scale from 1 *(really unhappy)* to 10 *(really happy)*.

Wide Format: A column for each level of the independent variable

Long Format: Multiple rows for each level of the independent variable

Wide Format

Rainy & Fresh Spring Day	Warm & Sunny Summer Day	Chilly & Brisk Fall Day	Cold & Snowy Winter Day
5	7	6	2
6	8	2	1
4	9	5	3
3	10	7	4

Long Format

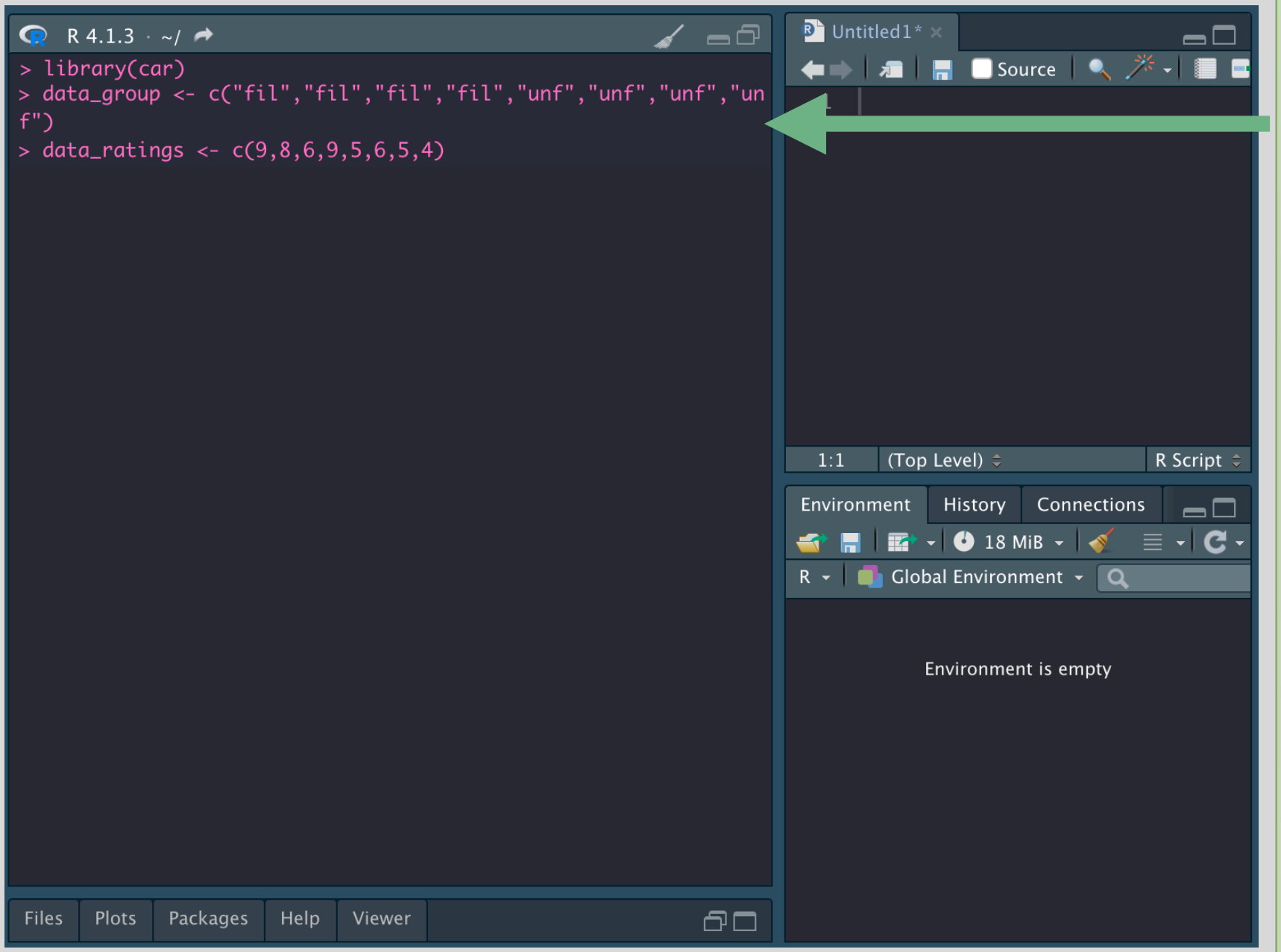
Season	Happiness Rating
Spring	5
Spring	6
Spring	4
Spring	3
Summer	7
Summer	8
Summer	9
Summer	10
Fall	6
Fall	2
Fall	5
Fall	7
Winter	2
Winter	1
Winter	3
Winter	4

Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.

Create your dataset in R.

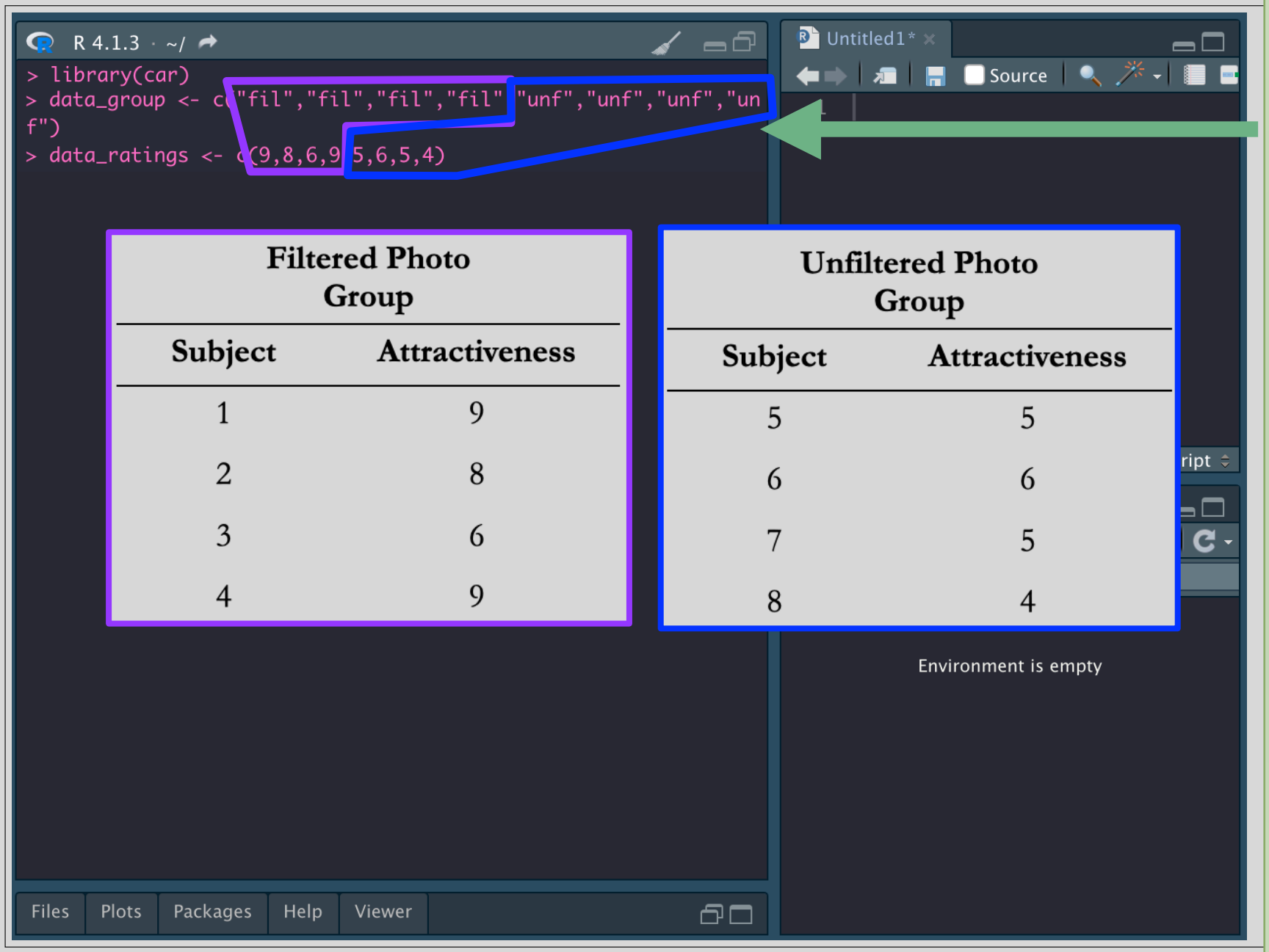
- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.
- Be careful about writing your code because you need to line up your values for your IV and DV.



Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.
- Be careful about writing your code because you need to line up your values for your IV and DV.

- I decided to assign values for the **Filtered group = "fil"** and values for the **Unfiltered group = "unf"**. The first four values listed are attractiveness ratings for Filtered and then Unfiltered.



Filtered Photo Group

Subject	Attractiveness
1	9
2	8
3	6
4	9

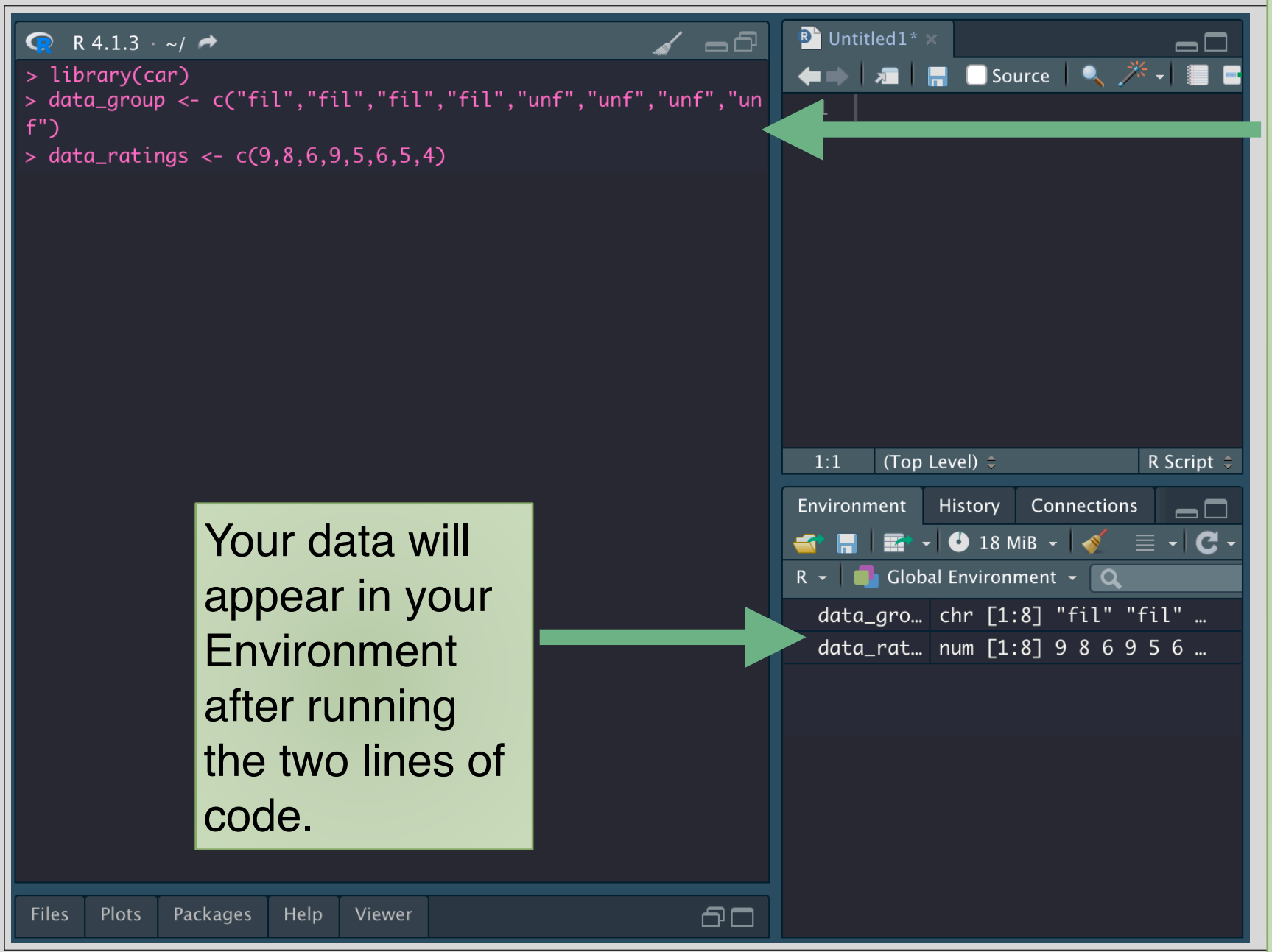
Unfiltered Photo Group

Subject	Attractiveness
5	5
6	6
7	5
8	4

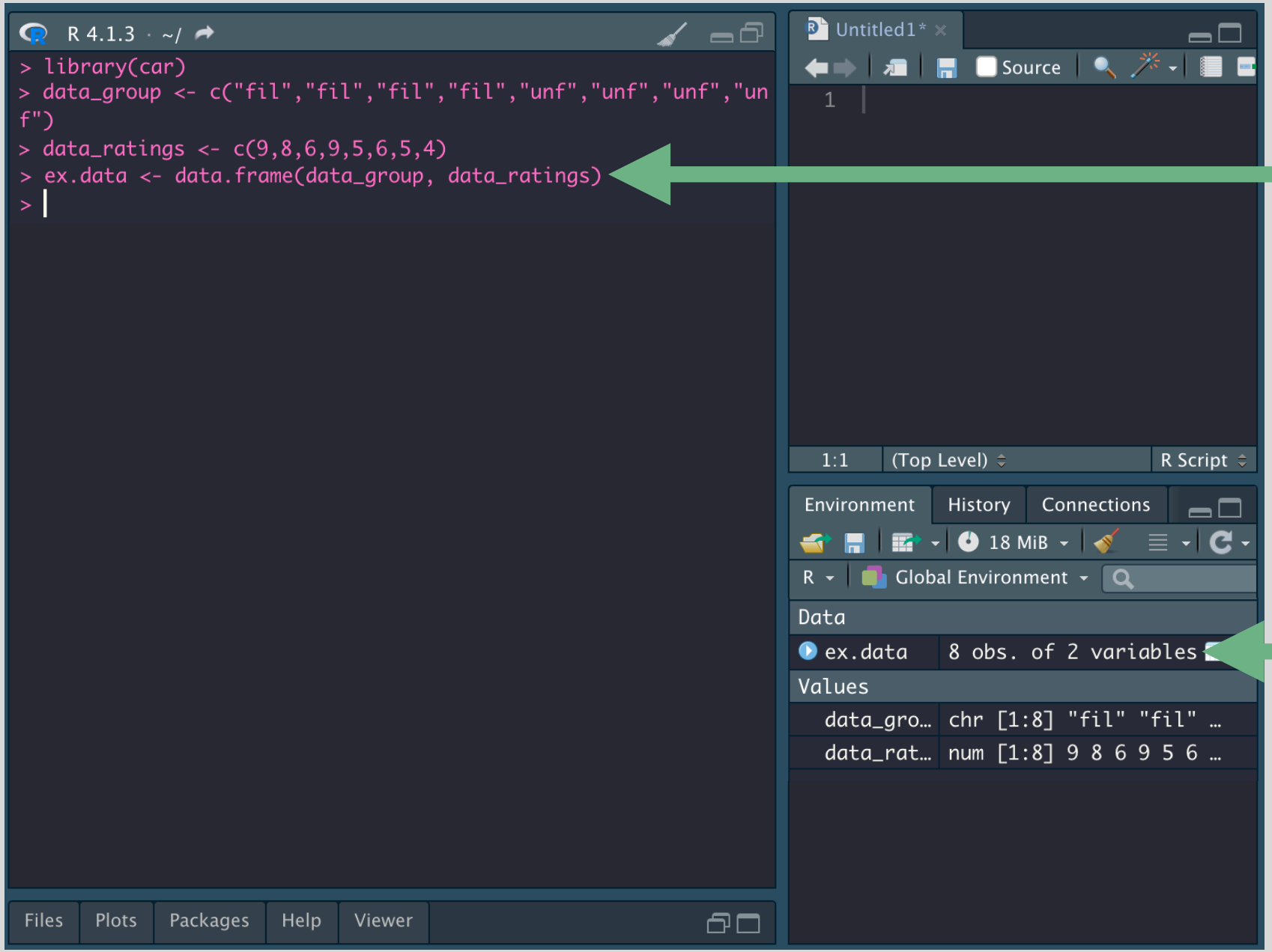
Create your dataset in R.

- You will do this just the same way as before using a name, assignment operator, and **c()**.
- **NOTE:** R analyzes data in *long* format. This means that instead of having a column for each dataset (or condition), you need to have one column for your dependent variable (across datasets) and one column for your independent variable.
- Be careful about writing your code because you need to line up your values for your IV and DV.

- I decided to assign values for the **Filtered group = "fil"** and values for the **Unfiltered group = "unf"**. The first four values listed are attractiveness ratings for Filtered and then Unfiltered.



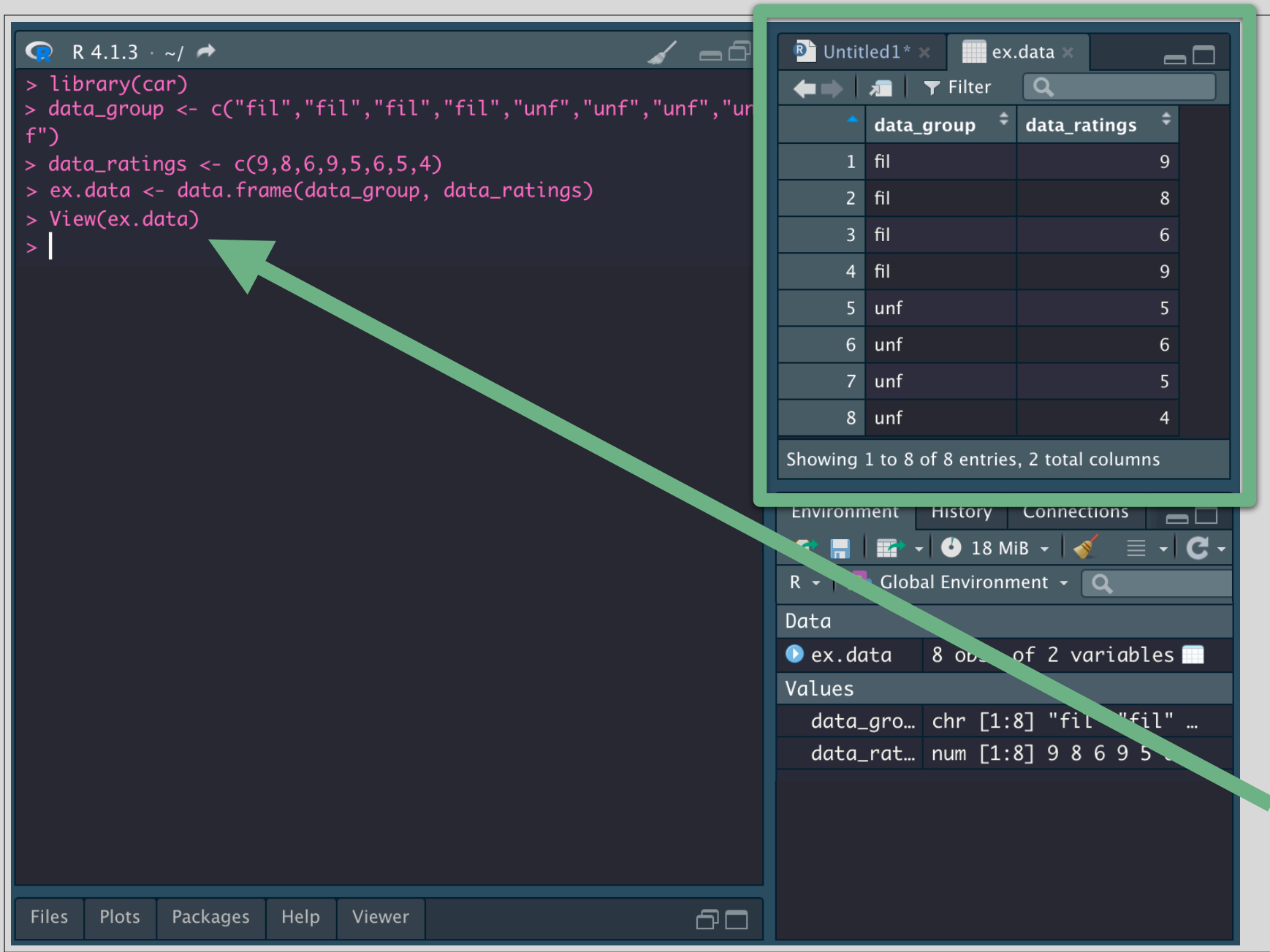
Your data will appear in your Environment after running the two lines of code.



Use **data.frame()** to create your combined data matrix.

- Type the names you assigned for your data in the parentheses in no particular order.

Your data matrix will appear in your Environment after running the line of code.

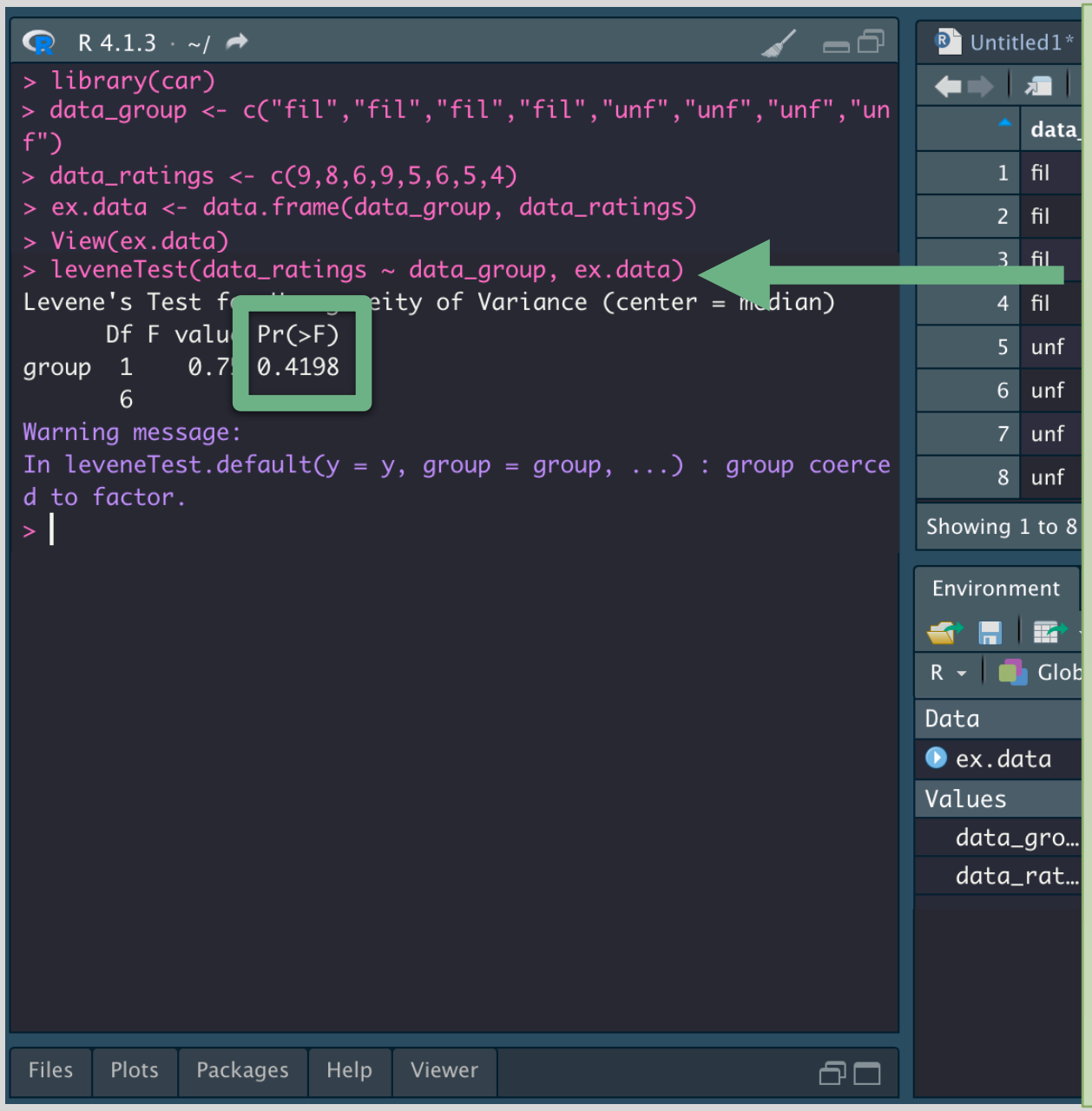


You can see now how the data is encoded into R (i.e., *long* format).

The **data_group** column corresponds to whether the score in **data_ratings** is from the filtered or unfiltered group.

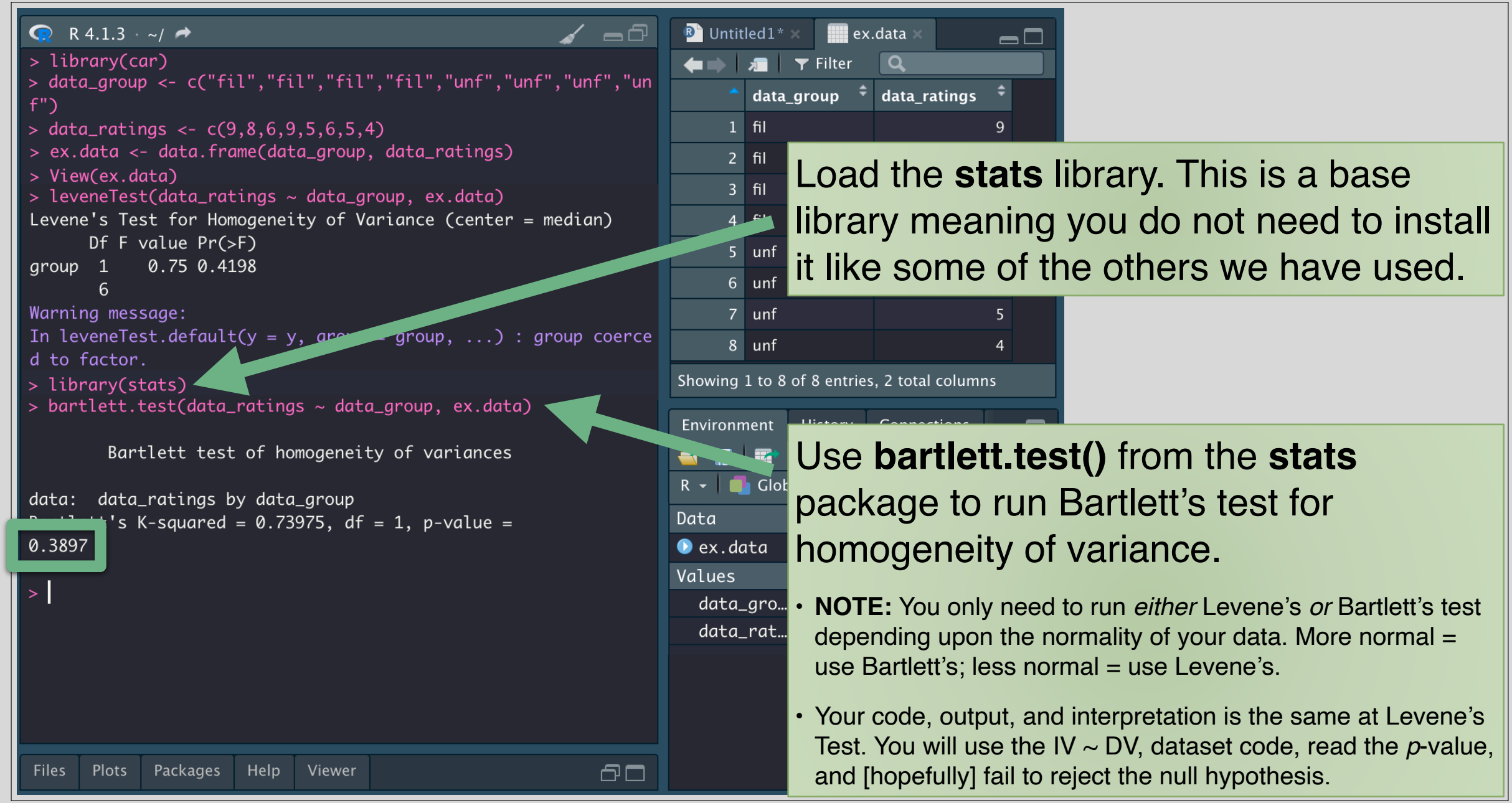
Use **View()** to see your data matrix.

- Type the name you assigned your data matrix in the parentheses.



Use **leveneTest()** from the **car** package to run Levene's test for homogeneity of variance.

- For your code, the first thing you will type in the parentheses is your dependent variable by (~) your independent variable: **data_ratings ~ data_group**. You will then use a comma and type the name of your data matrix to indicate to the function which data to run the code.
- Levene's Test is similar to a t-test in that you have a null hypothesis [that the variances for each of your samples are the same] and an alternative hypothesis [that the variances for each of your samples is different].
- To meet the assumption criteria, you need to fail to reject the null—stating that there are no differences.
- For your output, you will pay attention to the value for **Pr(>F)** otherwise known as a *p*-value.
- If your *p*-value is greater than 0.05, then you fail to reject the null meaning the variances for each of your samples are statistically equivalent (good!). If your *p*-value is less than 0.05, you reject the null meaning the variances for each of your samples are statistically different (bad!).


R 4.1.3 · ~/
> library(car)
> data_group <- c("fil","fil","fil","fil","unf","unf","unf","unf")
> data_ratings <- c(9,8,6,9,5,6,5,4)
> ex.data <- data.frame(data_group, data_ratings)
> View(ex.data)
> leveneTest(data_ratings ~ data_group, ex.data)
Levene's Test for Homogeneity of Variance (center = median)
Df F value Pr(>F)
group 1 0.75 0.4198
6
Warning message:
In leveneTest.default(y = y, group, ...) : group coerced to factor.
> library(stats)
> bartlett.test(data_ratings ~ data_group, ex.data)
Bartlett test of homogeneity of variances
data: data_ratings by data_group
's K-squared = 0.73975, df = 1, p-value =
0.3897
>
Files
Plots
Packages
Help
Viewer
Untitled1*
ex.data
Filter
data_group
data_ratings
1 fil 9
2 fil
3 fil
4
5 unf
6 unf
7 unf 5
8 unf 4
Showing 1 to 8 of 8 entries, 2 total columns
Environment
R
Data
ex.data
Values
data_gro...
data_rat...
Load the stats library. This is a base library meaning you do not need to install it like some of the others we have used.
Use bartlett.test() from the stats package to run Bartlett's test for homogeneity of variance.
• NOTE: You only need to run either Levene's or Bartlett's test depending upon the normality of your data. More normal = use Bartlett's; less normal = use Levene's.
• Your code, output, and interpretation is the same at Levene's Test. You will use the IV ~ DV, dataset code, read the p-value, and [hopefully] fail to reject the null hypothesis.

Step 2: State Hypotheses

There is **no significant difference** in means between the two samples. Implies the samples represent the **same population** of scores.

$H_0: \mu_1 = \mu_2$

$H_0: \mu_1 - \mu_2 = 0$

$H_a: \mu_1 \neq \mu_2$

$H_a: \mu_1 - \mu_2 \neq 0$

There is a **significant difference** between the two sample means. Implies the means from each sample represent **different populations** of scores

Step 3: Degrees of Freedom and T-Critical

- We are still going to use our T-Table to determine our *t*-critical values.
- Formula for the degrees of freedom is a bit different for an independent-samples *t*-test because we now take into consideration **two samples**.

$$df = (N_1 - 1) + (N_2 - 1)$$

$$df = (4 - 1) + (4 - 1)$$

$$df = (3) + (3) = \mathbf{6}$$

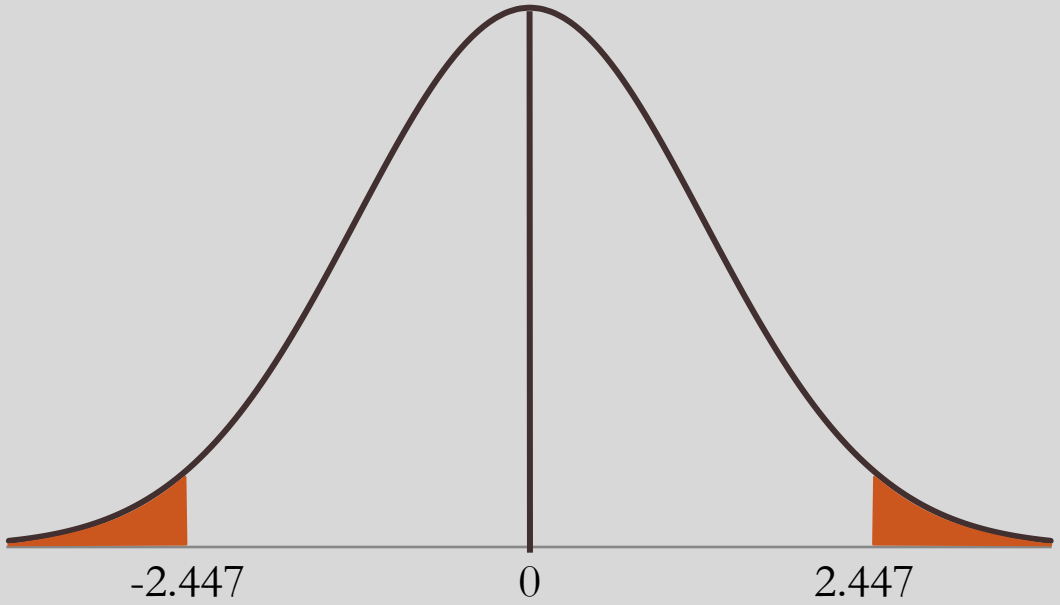
Step 3: Degrees of Freedom and T Critical

$$total\ df = 6$$

t-critical (two-tailed test at $\alpha = .05$) $= \pm 2.447$

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073

Step 3: Degrees of Freedom and T-Critical



Step 4: Calculate T-Statistic

- **Calculate the mean as usual** (sum all scores and then divide by sample size)

Filtered Photo Group	
Subject	**Attractiveness**
1	9
2	8
3	6
4	9
	8.00

Unfiltered Photo Group	
Subject	**Attractiveness**
5	5
6	6
7	5
8	4
	5.00

Step 4: Calculate T-Statistic

Then use the raw scores for each sample to calculate an **estimated population variance** for each group.

- If you remember from earlier in the semester, the variance formula is basically the same thing as the SD formula.
  - The only difference is that the variance is **missing** the square root sign.

$$\hat{\sigma}^2 = \frac{\Sigma X^2 - \frac{(\Sigma X)^2}{N}}{N - 1}$$

Calculating the Estimated Population Variance

Filtered Photo Group	
Subject	**Attractiveness**
1	9
2	8
3	6
4	9
	8.00

Unfiltered Photo Group	
Subject	**Attractiveness**
1	5
2	6
3	4
4	4
	5.00

$$\hat{\sigma}^2_{filter} = \frac{262 - \frac{(32)^2}{4}}{4 - 1}$$

$$\hat{\sigma}^2_{filter} = 2.00$$

$$\hat{\sigma}^2_{unfiltered} = \frac{102 - \frac{(20)^2}{4}}{4 - 1}$$

$$\hat{\sigma}^2_{unfiltered} = .6667$$

Pooling the Variances

When we "pool" the variances, we are basically just adding them together and finding the average between them.

$$\sigma^2_{pool} = \frac{(df_1)\hat{\sigma}^2_1 + (df_2)\hat{\sigma}^2_2}{(df_1) + (df_2)}$$

Pooling the Variances

$$\sigma^2_{pool} = \frac{(df_1)\hat{\sigma}^2_1 + (df_2)\hat{\sigma}^2_2}{(df_1) + (df_2)}$$

$$\sigma^2_{pool} = \frac{(3)2.00 + (3)\,.6667}{(3) + (3)} \Rightarrow \sigma^2_{pool} = \frac{8.0001}{6} = 1.3334$$

Step 4: Calculate T-Statistic

- After calculating the two sample means and the estimated population variance, we can plug those values into our *t*-statistic formula.

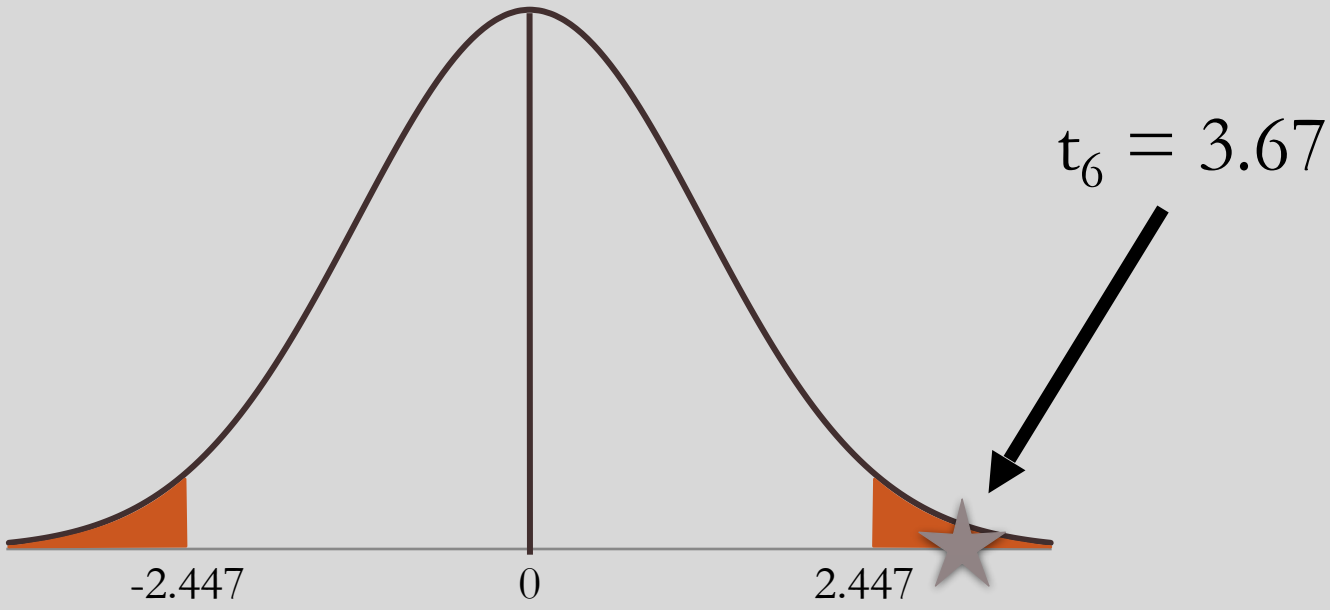
$\bar{X}_1$ = sample 1 mean
$\bar{X}_2$ = sample 2 mean
$\boldsymbol{\sigma^2_{pool}}$ = pooled estimated population variance
N = sample size (for each sample 1 and 2)

$$t_{df} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma^2_{pool}}{N_1} + \frac{\sigma^2_{pool}}{N_2}}}$$

Step 4: Calculate T-Statistic

$$t_{df} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma^2_{pool}}{N_1} + \frac{\sigma^2_{pool}}{N_2}}} \Rightarrow t_6 = \frac{8 - 5}{\sqrt{\frac{1.3334}{4} + \frac{1.3334}{4}}} \Rightarrow t_6 = 3.674$$

Step 5: Compare T-Statistic to Rejection Region



Step 6: Calculate 95% Confidence Interval

- **For a two-tailed test, we are going to want to calculate both an upper and lower confidence interval limit.**

$$\mathrm{CI}_{\mathrm{upper}} = t\ critical \left(\sqrt{\frac{\sigma^2_{pool}}{N_1} + \frac{\sigma^2_{pool}}{N_2}} \right) + (\bar{X}_1 - \bar{X}_2)$$

$$\mathrm{CI}_{\mathrm{lower}} = -\ t\ critical \left(\sqrt{\frac{\sigma^2_{pool}}{N_1} + \frac{\sigma^2_{pool}}{N_2}} \right) + (\bar{X}_1 - \bar{X}_2)$$

Step 6: Calculate 95% Confidence Interval

Upper Confidence Interval Limit

$$CI_{upper} = 2.447\left(\sqrt{\frac{1.3334}{4} + \frac{1.3334}{4}}\right) + (8 - 5)$$

$$CI_{upper} = 2.447\left(\sqrt{.6667}\right) + (3)$$

$$CI_{upper} = 1.9980 + 3$$

$$CI_{upper} = 4.9980$$

Lower Confidence Interval Limit

$$CI_{lower} = -2.447\left(\sqrt{\frac{1.3334}{4} + \frac{1.3334}{4}}\right) + (8 - 5)$$

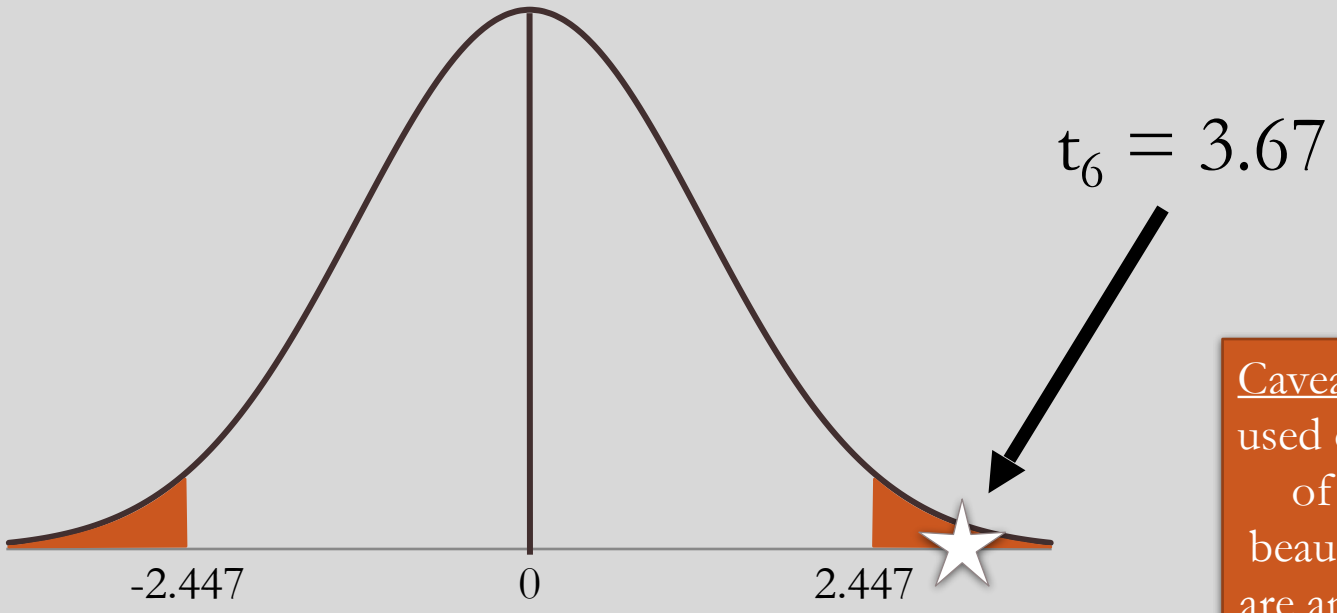
$$CI_{lower} = -2.447\left(\sqrt{.6667}\right) + (3)$$

$$CI_{lower} = -1.9980 + 3$$

$$CI_{lower} = 1.0020$$

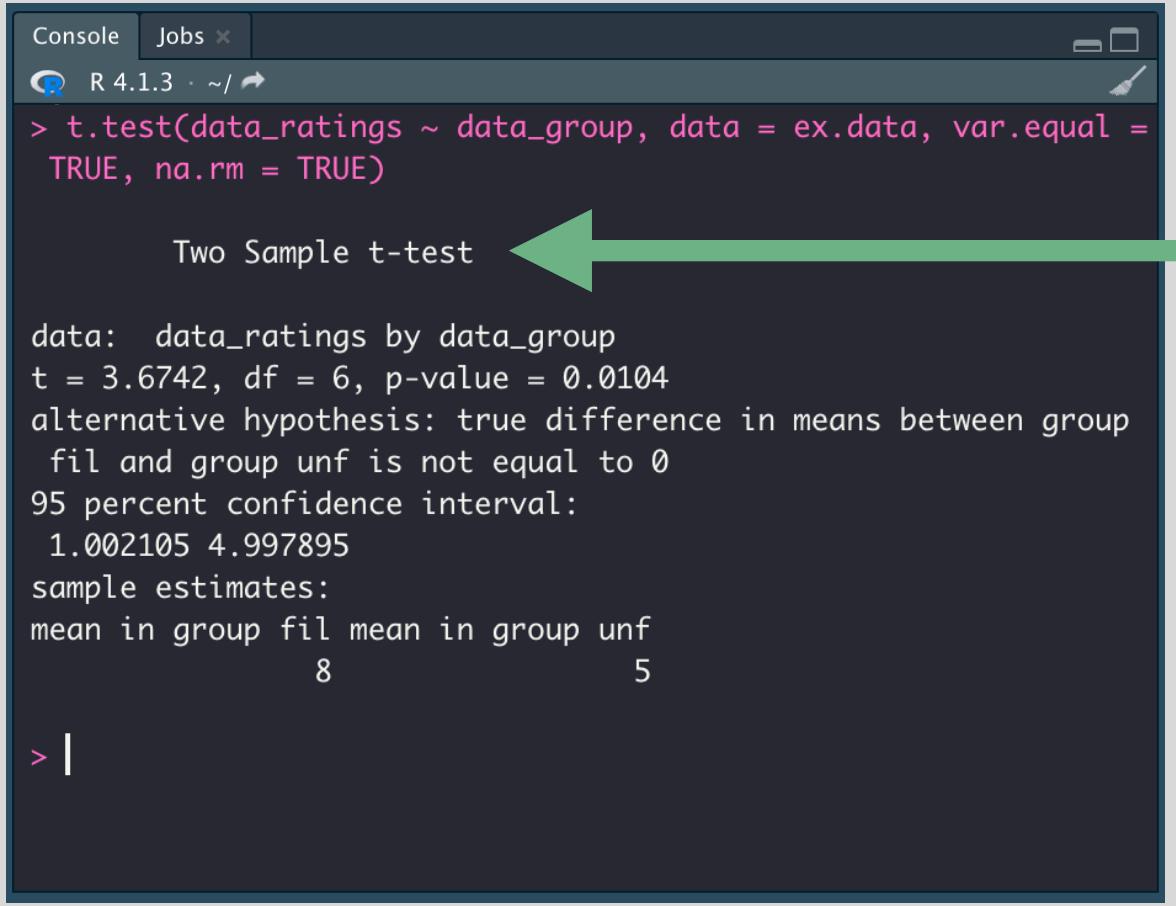
Step 7: Interpretation/Conclusion

Reject the null hypothesis: Participants who viewed filtered photos ($\bar{X} = 8$) differed significantly in their attractiveness rating compared to participants who saw unfiltered photos ($\bar{X} = 5$), $t(6) = 3.67$, $p < .05$. Participants who saw filtered photos rated them as more attractive. We can be confident the true mean difference in attractiveness rating between filtered and unfiltered photos is between 1.00 and 5.00.



Caveat: This is FAKE data used only for the purposes of this class. You are beautiful just the way you are and you do NOT need a filter to be a 10 out of 10 holy hot! level of attractiveness.

Independent-Samples T-Test in R (two-tailed)

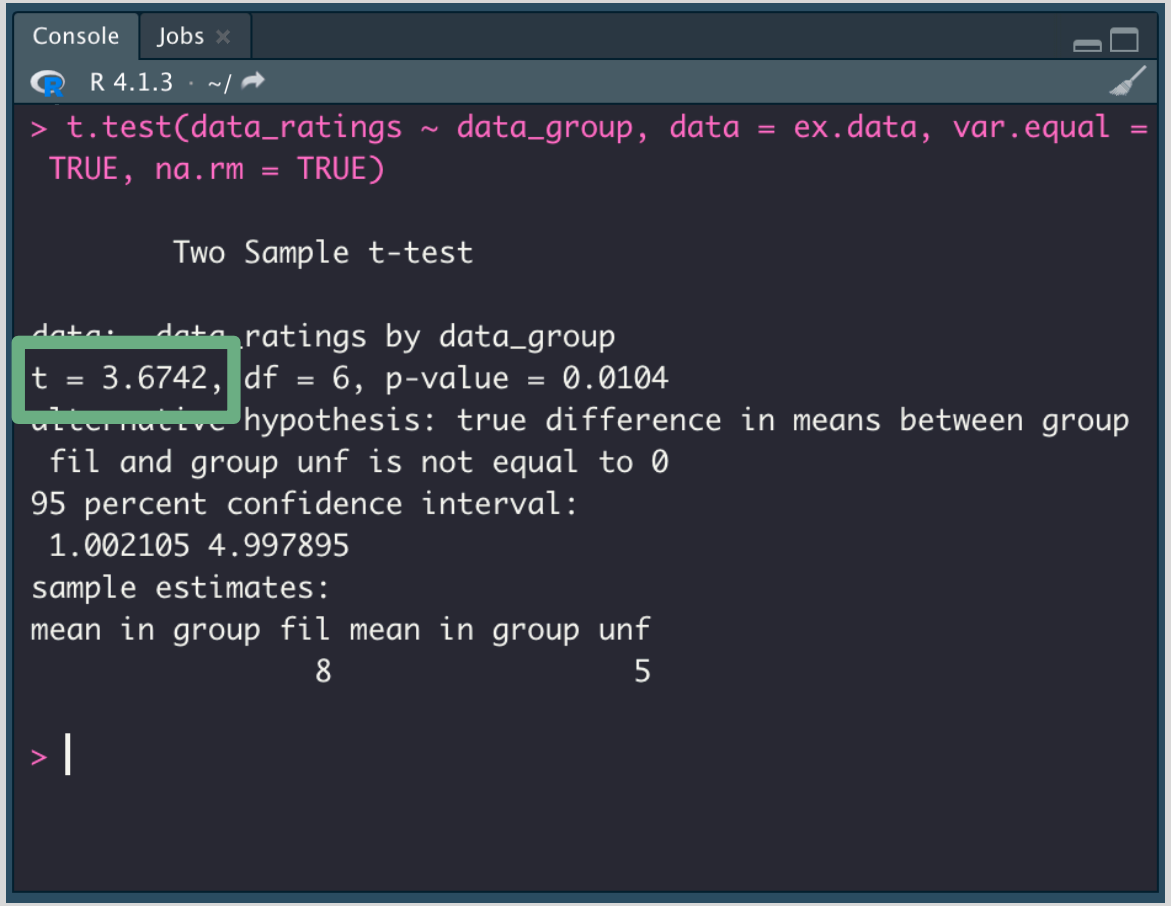


Use **t.test()** to conduct your analysis.

- The first thing you will write in your parentheses is the formula: **dependent variable ~ independent variable**.
- Then, you will include a comma and type in the name of your dataset. You will write **data =** and then the name of your data.
- Then, you will include another comma and fill in the argument **var.equal =** with either a **TRUE** (if you met the homogeneity of variance assumption) or **FALSE** (if you did NOT meet the assumption).
- Then, you will include a final comma and fill in the argument **na.rm =** with either a **TRUE** or **FALSE**. When na.rm is TRUE, the function skips over any NA values (not applicable; this means there is a missing data point). However, when na.rm is FALSE, then it returns NA from the calculation being done on the entire row or column.

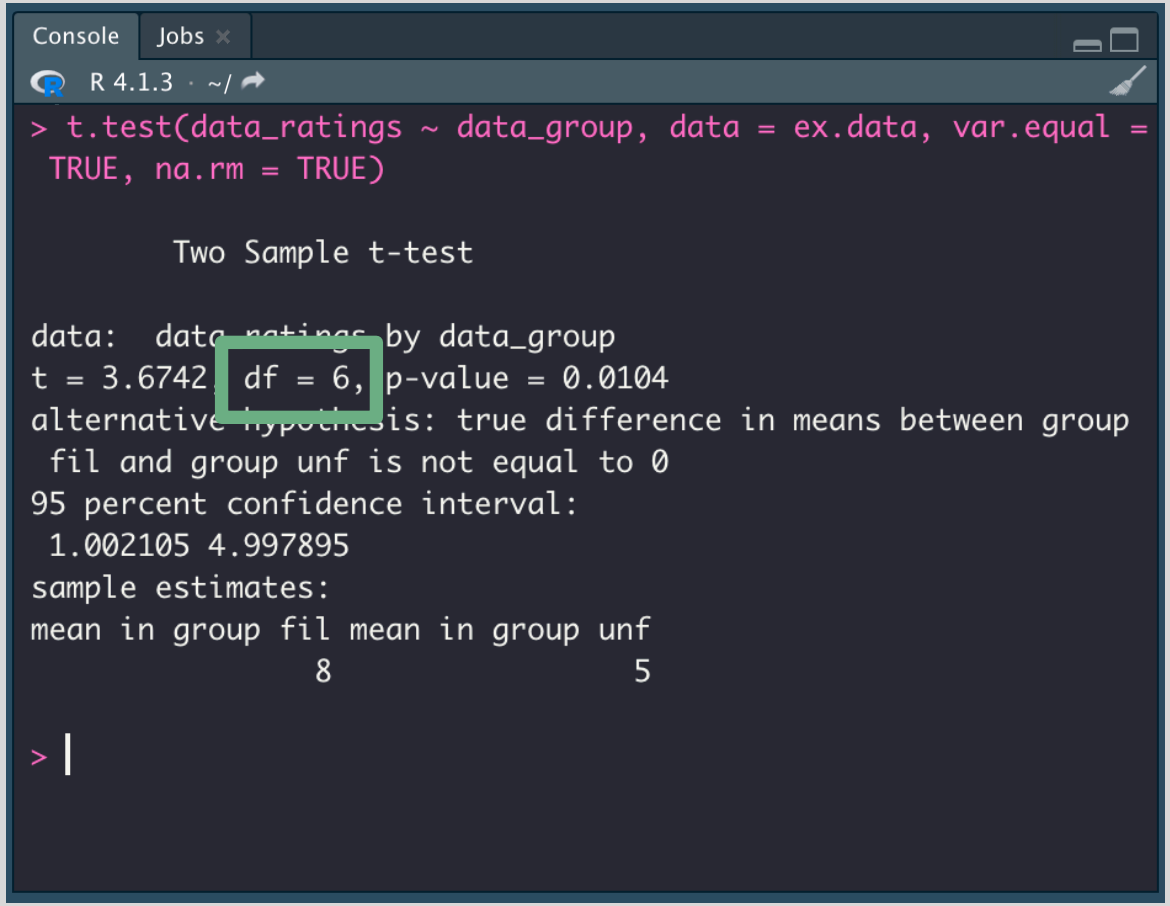
REMEMBER: If you are doing a one-tailed test, you need to include the **alternative =** argument (!!!!!!!!)

Independent-Samples T-Test in R (two-tailed)



t = *t*-statistic that we calculate to compare to the *t*-critical value

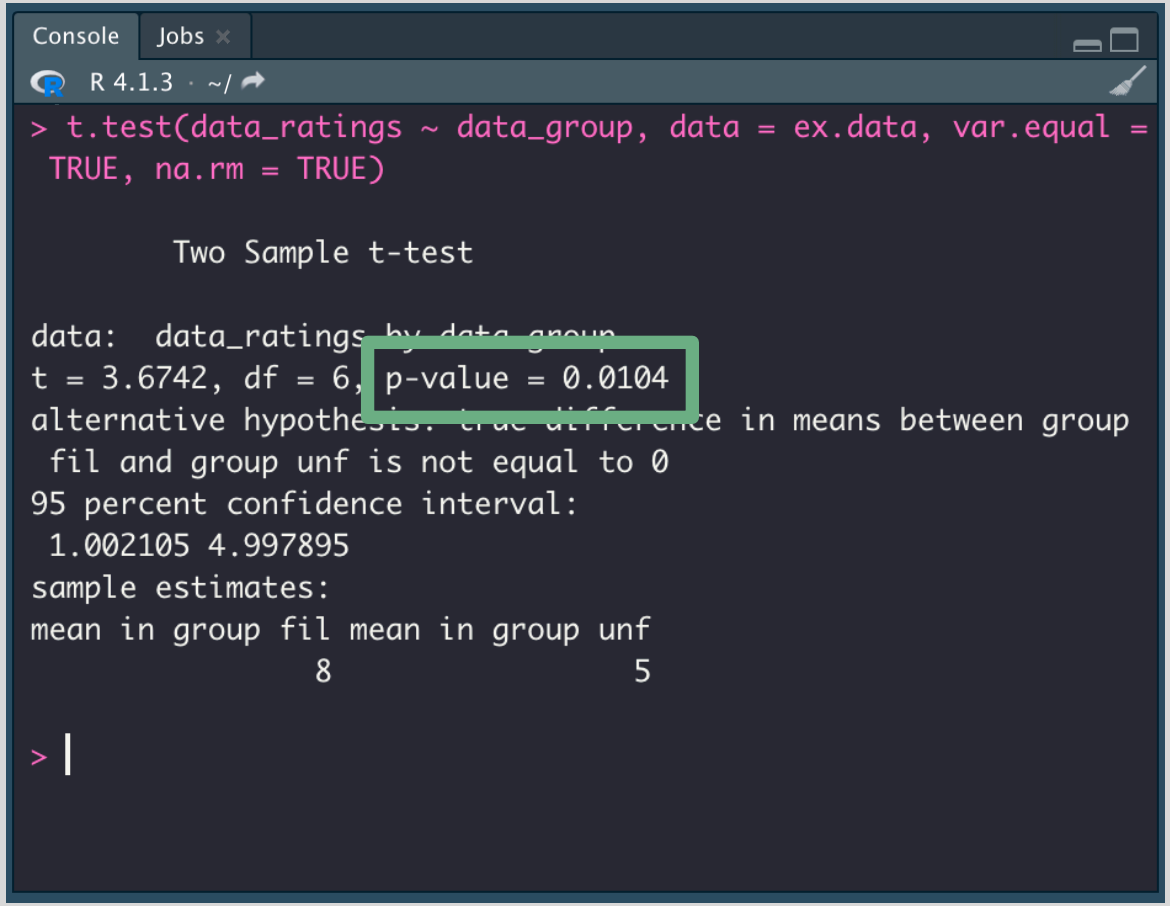
Independent-Samples T-Test in R (two-tailed)



t = *t*-statistic that we calculate to compare to the *t*-critical value

df = degrees of freedom for an independent-samples *t*-test

Independent-Samples T-Test in R (two-tailed)

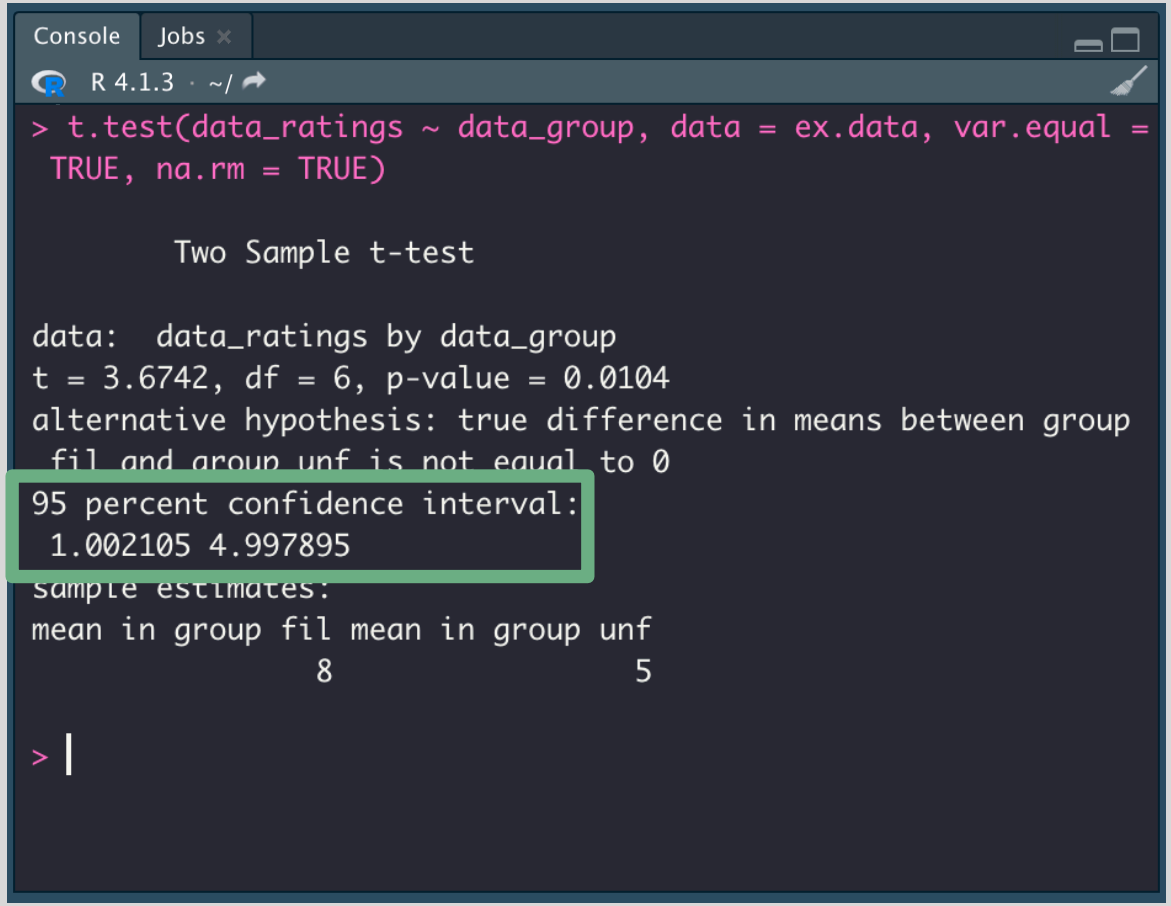


t = *t*-statistic that we calculate to compare to the *t*-critical value

df = degrees of freedom for an independent-samples *t*-test

p-value = *p*-value that we compare to 0.05; if our *p*-value is less than 0.05, we reject the null hypothesis; if our *p*-value is greater than or equal to 0.05, we fail to reject the null hypothesis

Independent-Samples T-Test in R (two-tailed)



t = *t*-statistic that we calculate to compare to the *t*-critical value

df = degrees of freedom for an independent-samples *t*-test

p-value = *p*-value that we compare to 0.05; if our *p*-value is less than 0.05, we reject the null hypothesis; if our *p*-value is greater than or equal to 0.05, we fail to reject the null hypothesis

95% confidence interval = if our CI includes 0 (zero), then we fail to reject the null hypothesis; if our CI does not include zero, we reject the null hypothesis