STAT210/410 Week 1 Workshop

Using Rstudio and Revision of the Simple Linear Regression Model

This workshop will help you revise simple linear regression and the basics of statistical analysis using R and RStudio. It starts right at the beginning, and it is designed to support students who haven't used this software before OR need a refresher on how the software works.

In this session, you will learn how to:

- Download files from the STAT210/410 website.
- Access R via RStudio.
- Open an editor to create command (script) files.
- Save your output, including graphics.
- Access a word processor (Word) to write up your results.
- Install supplementary packages, like `ggplot2`, `car` and `ggResidpanel`.

R and Rstudio can be used on a mac OS or Windows. If your operating system differs from the unit coordinators, there may be slight changes in the appearance of the software, but all should function the same way. Where instructions differ between operating systems, they will be available for both.

1 Getting started

To start your session you must be **working on a computer with R and RStudio installed**. All of the computers in the UNE computer labs are ready to go. If you are working on your own computer you must ensure you've installed **BOTH R and Rstudio**. Instructions for download are available under the Week 0 block on the STAT210/410 Mylearn page. If you are an on campus student attending the face-to-face workshop sessions yet may be working on your own computer for assignments, quizzes, and practice you'll also need to download this software at some time. Alternatively, there is a browser based version of Rstudio available. If you are going to use the Rstudio browser, refer to the recording in the unit recordings titled: "Using Rstudio in your browser" of how to set up the browser version.

Before starting RStudio, you are going to create a file directory where you will store command files, data and output files.

1.1 Creating a file directory

1.1.1 Windows

1. Go to File Explorer by clicking on the yellow folder in the task bar at the bottom of the screen.
2. Click on This PC and then choose a location that suits your file management system. If you are using a computer in the UNE lab, create a folder in your barney drive, which is identified with your name.
3. Right mouse click and select New> Folder.
4. Label the new folder STAT210 (or STAT410).

1.1.2 Mac

1. Go to Finder by clicking on the Finder app.
2. Choose a location in your file directory that suits your file management system. EG in the Documents folder. This is where we are going to make our new STAT210/410 folder.
3. Right click with the mouse and select “New folder”.
4. Rename your folder, by right clicking on it and selecting “Rename”. Give your folder an appropriate name. EG STAT210 OR STAT410.

1.1.3 Browser based Rstudio

1. In the bottom right panel, under the files tab, select “New Folder” and name the folder “data sets”.
2. Click on the new folder “data sets” to open the folder. You will need to upload all your data files into this folder.

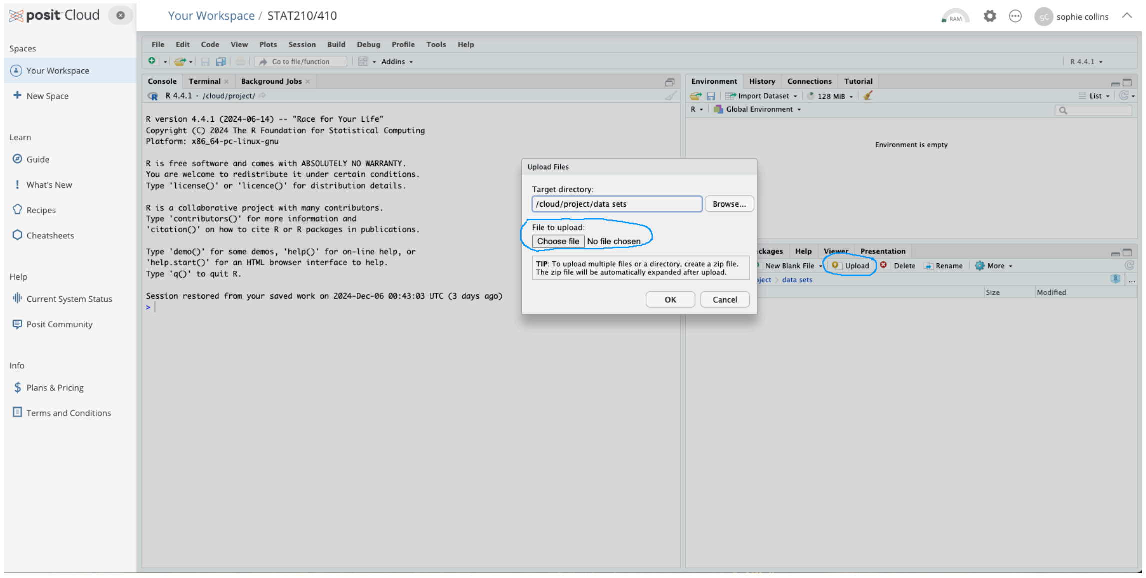
NOTE: You may still need a STAT210/410 folder on your computer if you want to download and save output from the Rstudio browser version. Follow the steps from above to do so, if you haven’t already.

1.2 Accessing Files from the STAT210/410 Mylearn

Now we will download the folder with the files off the STAT210/410 mylearn and save it to our newly made STAT210 OR STAT410 folder.

1. Go to the STAT210/410 Mylearn website via MyUNE.
2. Go to Unit Information and Resources.
3. Click on the link: “Data sets and R script files used in lectures and workshops”.
4. On the top right of that page, click on “Download folder”. This will download all the data files you will need.
5. Once the download is complete, move this folder to your STAT210 OR STAT410 folder and unzip it.
6. Rename the folder “data sets”.

NOTE: on a **Mac**, you simply double click on the zipped folder and this will work. For **Windows**, you will need to select “Extract all” from the File Explorer Window menu and follow the prompts. If you are working in the **Browser** version of Rstudio, you will need to first unzip the file on your computer before uploading the individual data files to the cloud. Select “upload” from the bottom right panel in the Rstudio browser, then “Choose file” and navigate to the data file you want to upload.



```
R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Session restored from your saved work on 2024-Dec-06 00:43:03 UTC (3 days ago)
>
```

Upload Files

Target directory: /cloud/project/data sets

File to upload: Choose file No file chosen

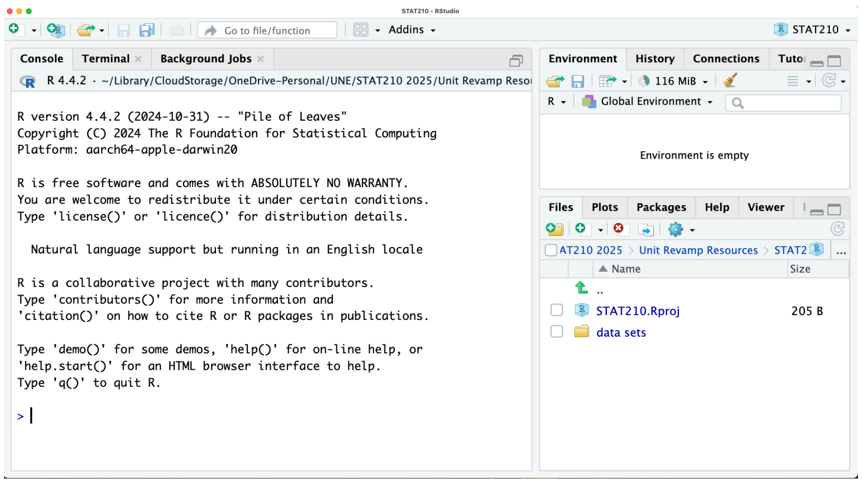
TIP: To upload multiple files or a directory, create a zip file. The zip file will be automatically expanded after upload.

posit Cloud
Your Workspace / STAT210/410
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Background Jobs
R 4.4.1 · /cloud/project/
Environment History Connections Tutorial
Environment is empty
Browse...
OK
Cancel
Upload
Delete
Rename
More

2 Introduction to Rstudio layout

You can now open Rstudio. You will need to set a project and open a new R script. If you are new to R and Rstudio or need a refresher on how to do this, the details are below. **If you are already familiar with R and Rstudio, you can skip ahead to 3.**

Once RStudio is open, the following screen will appear:



```
R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

STAT210 - RStudio
Go to file/function
Addins
STAT210
Console Terminal Background Jobs
R 4.4.2 · ~/Library/CloudStorage/OneDrive-Personal/UNE/STAT210 2025/Unit Revamp Reso
Environment History Connections Tuto
116 MiB
Global Environment
Environment is empty
Files Plots Packages Help Viewer
AT210 2025 > Unit Revamp Resources > STAT2
Name Size
STAT210.Rproj 205 B
data sets

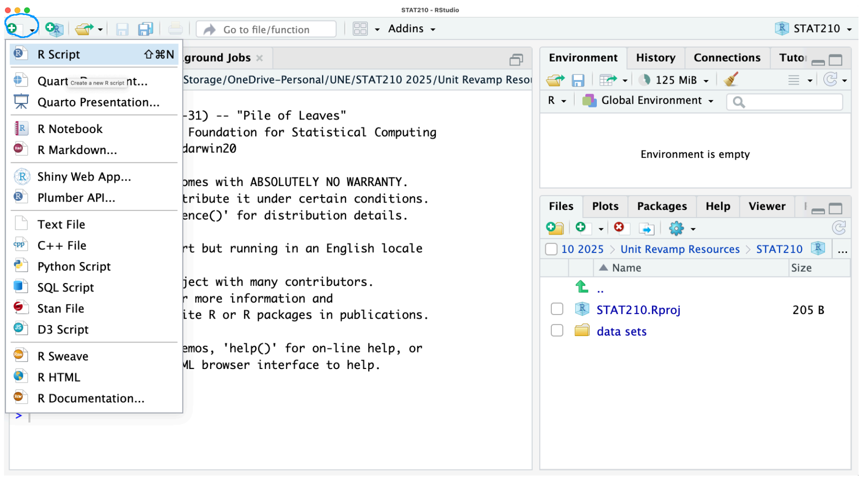
NOTE: If your screen does not look like this, you might have opened R instead of Rstudio by mistake.

Before you start, you must do two things. First, set up your RStudio project:

1. In RStudio, click on the Projects button in the top right-hand corner of the screen (it might say ’Project: None”), and select “New Project”. Choose “Existing directory”, then browse to the STAT210 *OR* STAT410 folder you just created. Then click “Create project”.
2. Now, you should see the name of your project (“STAT210” *OR* “STAT410”) in the top right-hand corner of the screen (like the screenshot above does).

3. Every time you open RStudio, you should check that you are working in that project, by looking for its name ("STAT210" *OR* "STAT410") in the top right-hand corner of the screen. If you aren't there, you can open it by clicking on the project and choosing "open project" and navigating to it, or by finding the folder in your File Explorer or Finder and double clicking on "STAT210.Rproj" *OR* "STAT410.Rproj". If you are using the browser version, you will need to login to posit and then select the project.

Second, create an R script. You can do this by selecting the white page with the plus in the top left panel and then R Script.

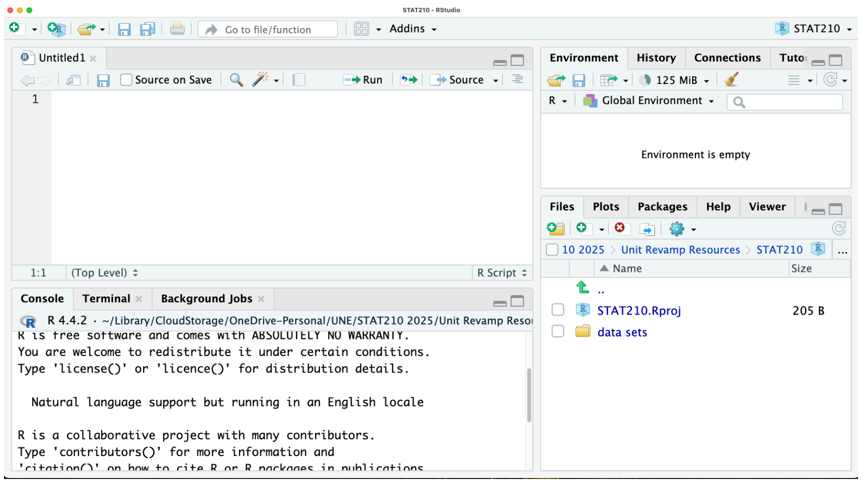


```
Storage/OneDrive-Personal/UNE/STAT210 2025/Unit Revamp Reso
-31) -- "Pile of Leaves"
 Foundation for Statistical Computing
darwin20

omes with ABSOLUTELY NO WARRANTY.
tribute it under certain conditions.
ence()' for distribution details.

rt but running in an English locale

ject with many contributors.
r more information and
ite R or R packages in publications.

emos, 'help()' for on-line help, or
ML browser interface to help.
```

Save your script with a sensible name, e.g., "STAT210 Workshop 1.R".

2.1 Understanding the layout

Now, your RStudio should look like this:



```
R 4.4.2 · ~/Library/CloudStorage/OneDrive-Personal/UNE/STAT210 2025/Unit Revamp Reso
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

The key areas you'll use are:

1. The **Script** window on the top left is where you will be writing (or cutting and pasting) your commands
2. The **Console** window on the bottom left is where see the commands you have run and their outputs
3. The **Environment** tab in the upper right window is where you can see the details of data sets you've loaded or variables you've created
4. The **Plots** tab in the lower right window is where your graphics will appear

When you work in RStudio, you should always type or paste your code into your script in the top-left hand corner. To run a line of code, you can put your cursor on that line or select it, and click ‘Run’ at the top of the script window, or use a shortcut, typically Ctrl+Enter for Windows or Cmd+Enter for Mac (hover over the run button to see the shortcut). When you do this, you will see the code you have run appear in the console in the bottom left panel, with the output below that.

3 Introduction to R Computing

Lets get started. Make sure you have set your project and started a new R script. Name your R script something sensible.

You can start using R following the examples below. There is also a consolidated R Resources page (https://turing.une.edu.au/~statshare/index.html) created by our stats team you can access that has some useful resources.

You can use R as a calculator

Type each of the following at the > prompt in the console, hitting enter after each line. Or type these into the script window and press the button ”run”.

```
3*4
```

```
3**5
```

```
3^5
```

```
sin(pi/4)
```

Some common calculation commands in R:

`+` plus

`-` minus

`/` divided by

`*` multiply by

`**` OR `^` to the power of

`sqrt()` square root

`log()` natural log

`log10()` log base 10

You can create variables and perform operations on them

Try the following, noting the use of `<-` to assign values.

Create a variable named x with numbers 1 to 10 contained within.

```
x<-1:10
x
```

Create a variable named y with the log numbers of x contained within.

```
y<-log(x)
y
```

Calculate a mean for x and for y.

```
mean(x)
```

```
## [1] 5.5
```

```
mean(y)
```

```
## [1] 1.510441
```

Other sample statistic commands include:

`sd(x)` to calculate a standard deviation of x

`var(x)` to calculate a variance of x

`summary(x)` to calculate the 5 point summary for x (min, Q1, median, Q3, max). Note that this also includes the mean too.

`length(x)` to find the total number of observations in x.

`quantile(x, 0.25)` to find the 25th quantile (Q1) of x. Note that Q3 is the 75th quantile.

Practice using the commands above by calculating the standard deviation, median and IQR for x and y. Remember that the IQR can be calculated by:

$$IQR = Q3 - Q1$$

▶ **CHECK**

4 Simple linear regression (SLR)

In this example we will **fit a simple linear regression model, obtain diagnostic plots, and use the model to predict values.**

We will use the following example to cover these concepts:

- Create a script file that will store all commands, rather than executing commands one at a time interactively.
- The `options(digits=)` command sets the number of significant figures in your output.
- Reading data into a data frame from a file using `read.csv()` .
- The linear model is fitted using the `lm()` command.

- Information is extracted from the `lm()` command using the functions `anova()`, `summary()` and `confint()`.
- `ggplot2` will be used to produce high quality graphics to include in reports and assignments.
- The function `resid_panel(model.lm)` produces residual/diagnostic plots for the linear model model.lm.
- Use `predict()` to predict mean and individual responses and find CI's and PI's for those predictions.

4.1 Example: Fungus

Laetisaric acid is a compound that holds promise for the control of fungus disease in crops. The data set `acid.csv` contains the results of growing the fungus, *Pythium ultimum*, in various concentrations of laetisaric acid. There are two variables in the dataset:

- `acid` : the concentration of laetisaric acid (µg/ml)
- `fungus` : Radius of the fungus growth after 24hrs (mm)

Source: Statistics for the Life Sciences, (5th edn), M. L. Samuels, J. A. Witmer, & A. A. Schaffner (2016), p. 533

4.1.1 Initial settings and importing data

First, set the number of significant figures for your output and turn off the significant figure stars with:

```
# set the number of significant figures for output
options(digits=3, show.signif.stars=F)
```

Next, import the `acid.csv` data:

```
# Read data from data file (acid.csv) and store in R in a data frame (acid.df)
acid.df<-read.csv("data sets/acid.csv",header=T)
```

NOTE: If you have followed the set up steps above correctly, you should have set your project file to the STAT210 or STAT410 folder and you should have a folder in the STAT210 or STAT410 folder called "data sets", where the `acid.csv` file is stored. If you get an error that looks like this:

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'acid.csv': No such file or directory
```

1. Check you have spelled everything in your code correctly
2. Make sure you have the data folder downloaded and saved in the STAT210/410 folder as "data sets".
3. Check you have your STAT210 or STAT410 project open in Rstudio.

Once the data has been read (imported) into Rstudio, you can check the names of the variables with:

```
# list names of variables
names(acid.df)
```

```
## [1] "acid"   "fungus"
```

```
# list first 3 lines of data frame
head(acid.df,3)
```

```
##   acid fungus
## 1    0   33.3
## 2    0   31.0
## 3    3   29.8
```

4.1.2 Exploratory analysis

Before starting any statistical inference, we should do some exploratory analysis of our data set. Primarily an exploratory plot. You could make this plot in base R, but we will instead introduce a plotting package called `ggplot2`.

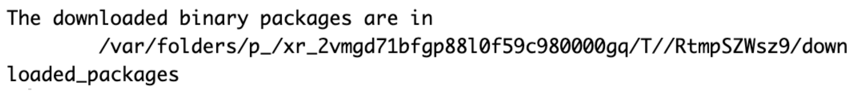
Packages are extensions that contain data sets and new functions which can be used to allow the use of a wider range commands. In this unit, we will use a few different packages. The `ggplot2` and `ggResidpanel` package that we will look at this week and the `GGally` package that we will look at next week contain new functions primarily concerned with the creation of plots. Packages need to first be installed into R and then loaded each time you want to use the functions.

First, you will need to install `ggplot2` with:

```
#install ggplot2.
install.packages("ggplot2")
```

NOTE: you only need to install the package once.

A bunch of red text will run, which is normal. Once the package is finished installing, the final output in the console should look something like this:



```
The downloaded binary packages are in
        /var/folders/p_/xr_2vmgd71bfgp88l0f59c980000gq/T//RtmpSZWsz9/down
loaded_packages
```

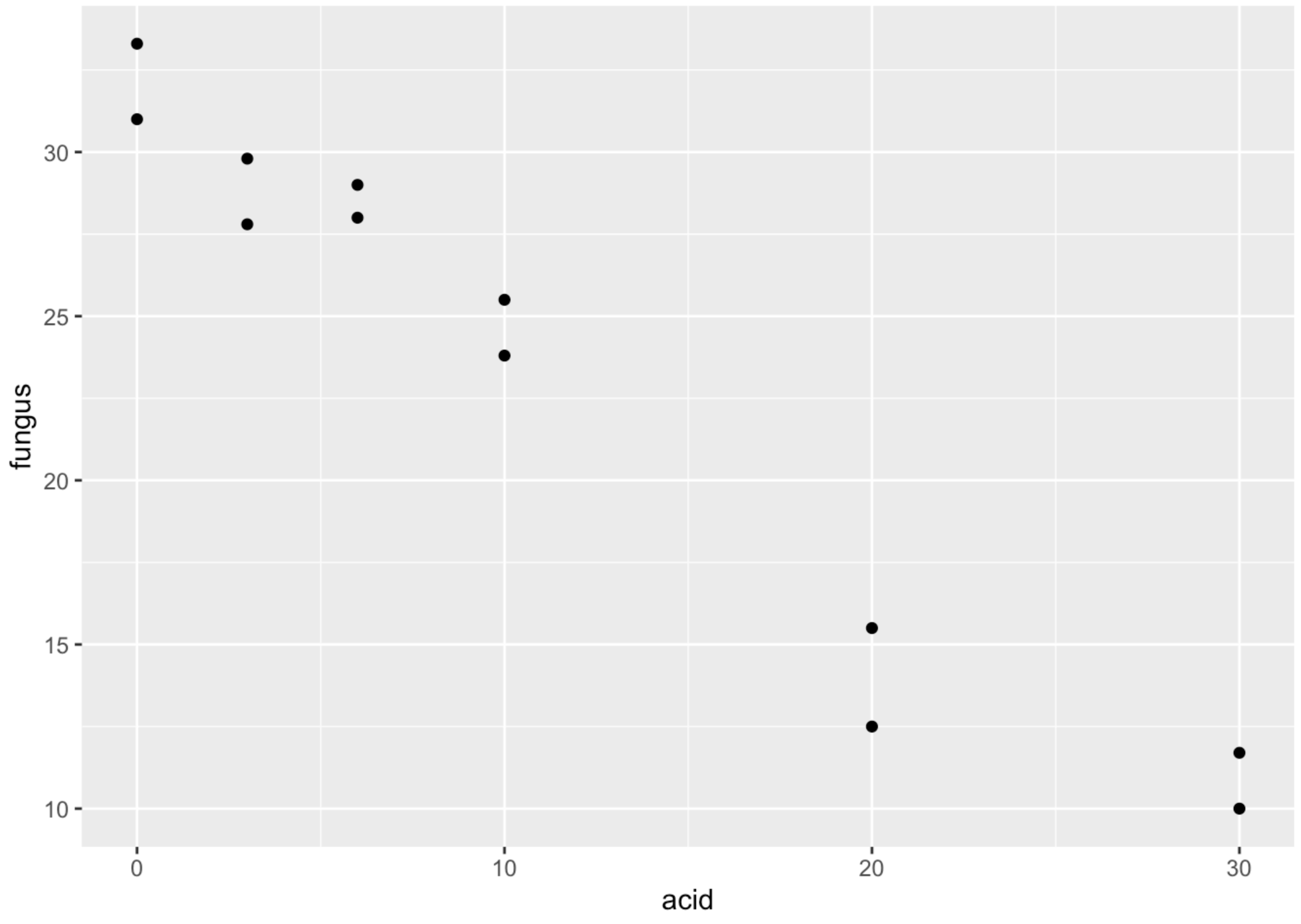
You can then load the `ggplot2` package with:

```
#load ggplot2 package.
library(ggplot2)
```

NOTE: you will need to load the package for each session you want to use a `ggplot2` function.

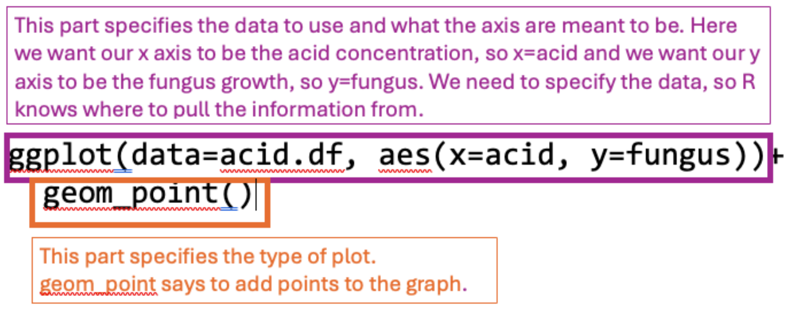
Once the package is loaded, we can start using the new commands from the `ggplot2` package. Let's start with a simple scatterplot showing a relationship between fungus growth and acid concentration:

```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point()
```



`ggplot` works by adding separate layers depending on what you specify to the plot. Each layer is added with a `+`. In this example we have 2 layers. The first layer gives us the base of the plot, eg what the axis should be. The second layer adds the points to the plot. If you want to test this, just run the first layer and see what you end up with!

Lets break down the code:



This part specifies the data to use and what the axis are meant to be. Here we want our x axis to be the acid concentration, so x=acid and we want our y axis to be the fungus growth, so y=fungus. We need to specify the data, so R knows where to pull the information from.

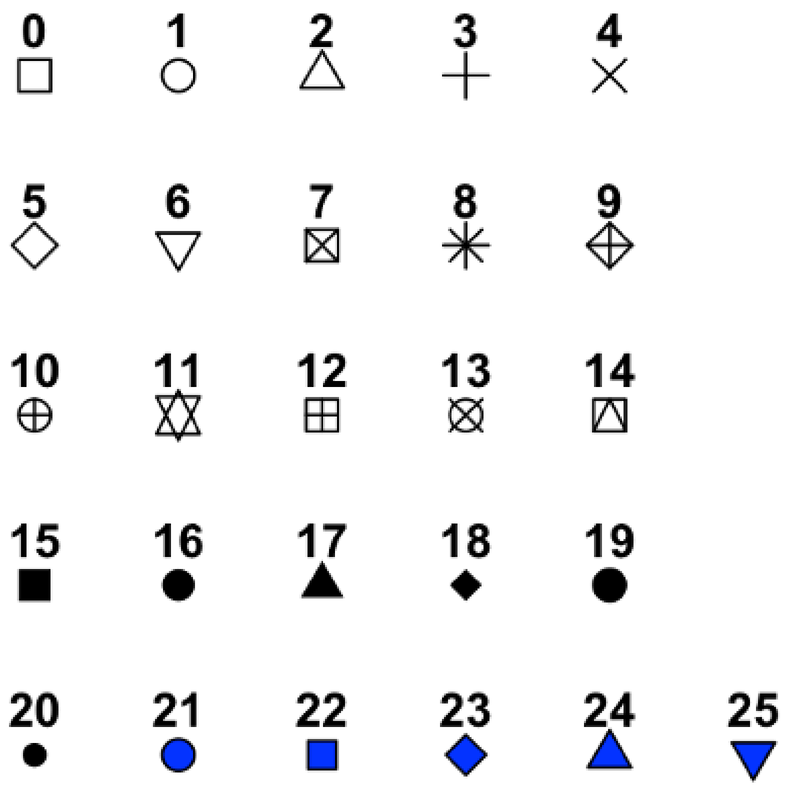
```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point()
```

This part specifies the type of plot. geom_point says to add points to the graph.

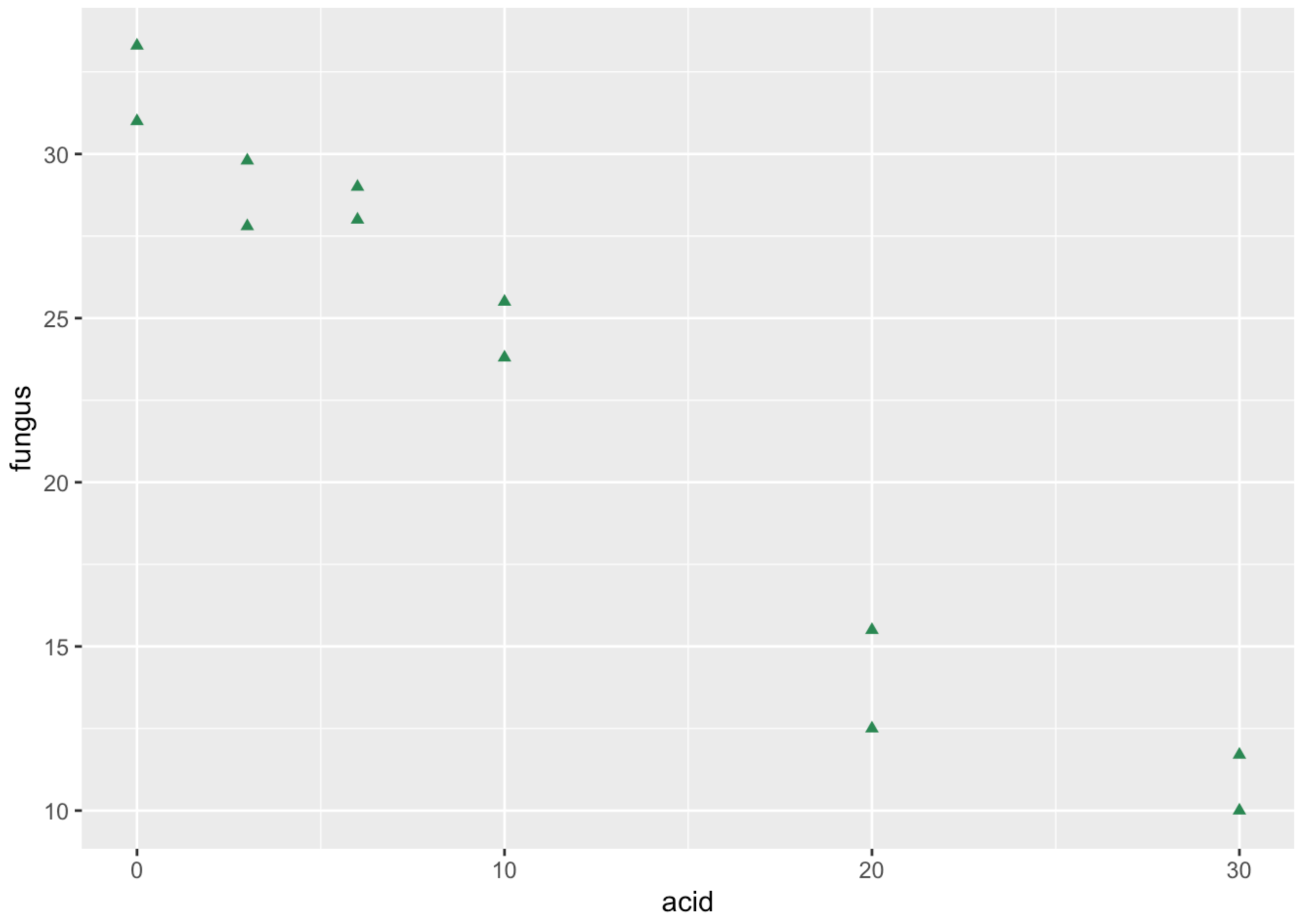
We can change the colour and shape of our plots using a number of different commands. Some common ones you might want to use are:

- `pch =` changes the type of point
- `lwd =` changes the width of the line
- `linetype =` changes the type of line
- `col = "colour"` changes the colour of the points. **NOTE:** change colour to your colour choice.

Here we will **change the symbol and colour of the points.** Refer to the symbol chart below for the different symbol options and the R colour chart (https://mylearn.une.edu.au/pluginfile.php/4817512/mod_resource/content/1/Rcolor.pdf) in mylearn for some colour options.

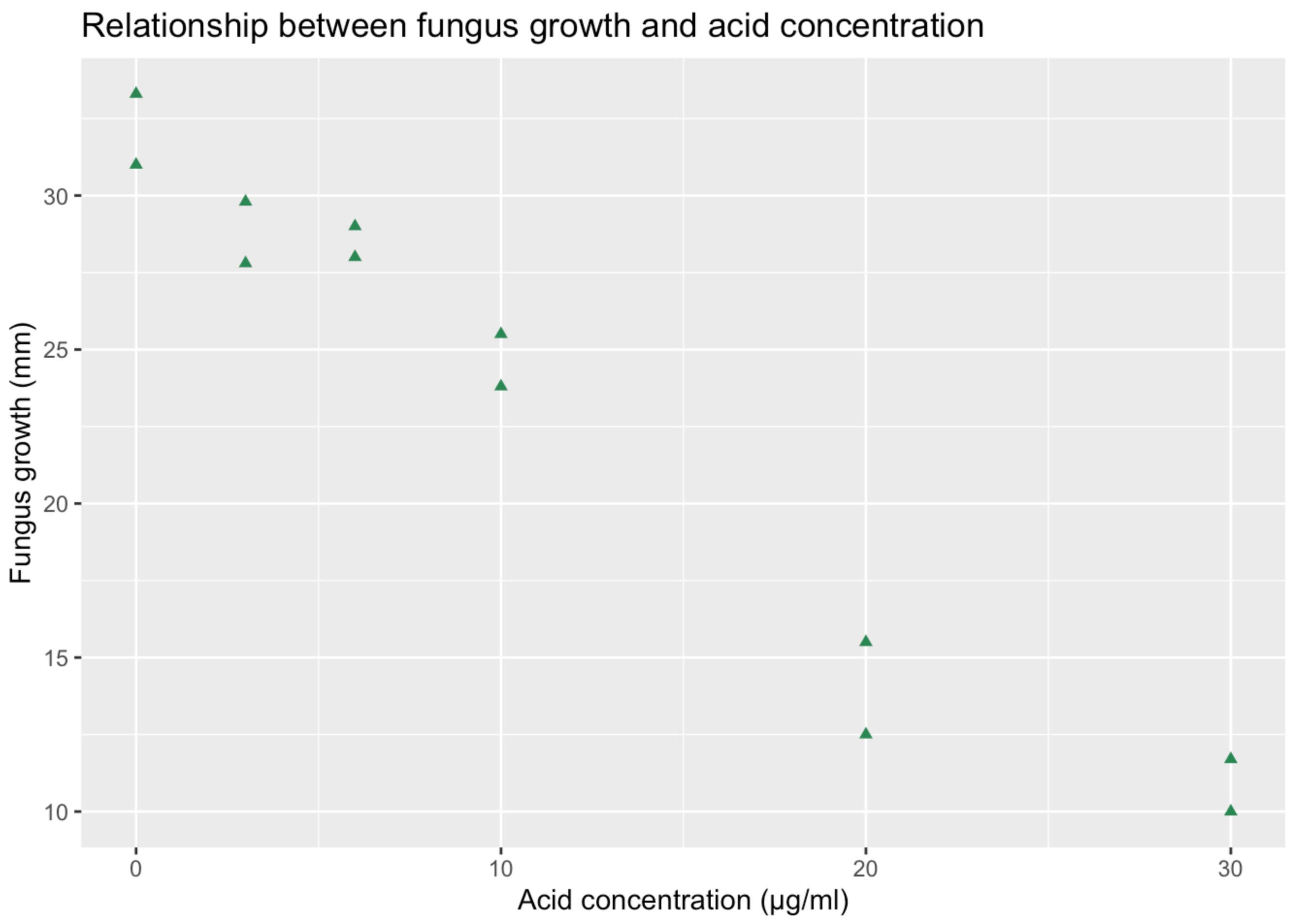


```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point(pch=17, col="seagreen")
```



To change our axis labels, we need to add another layer:

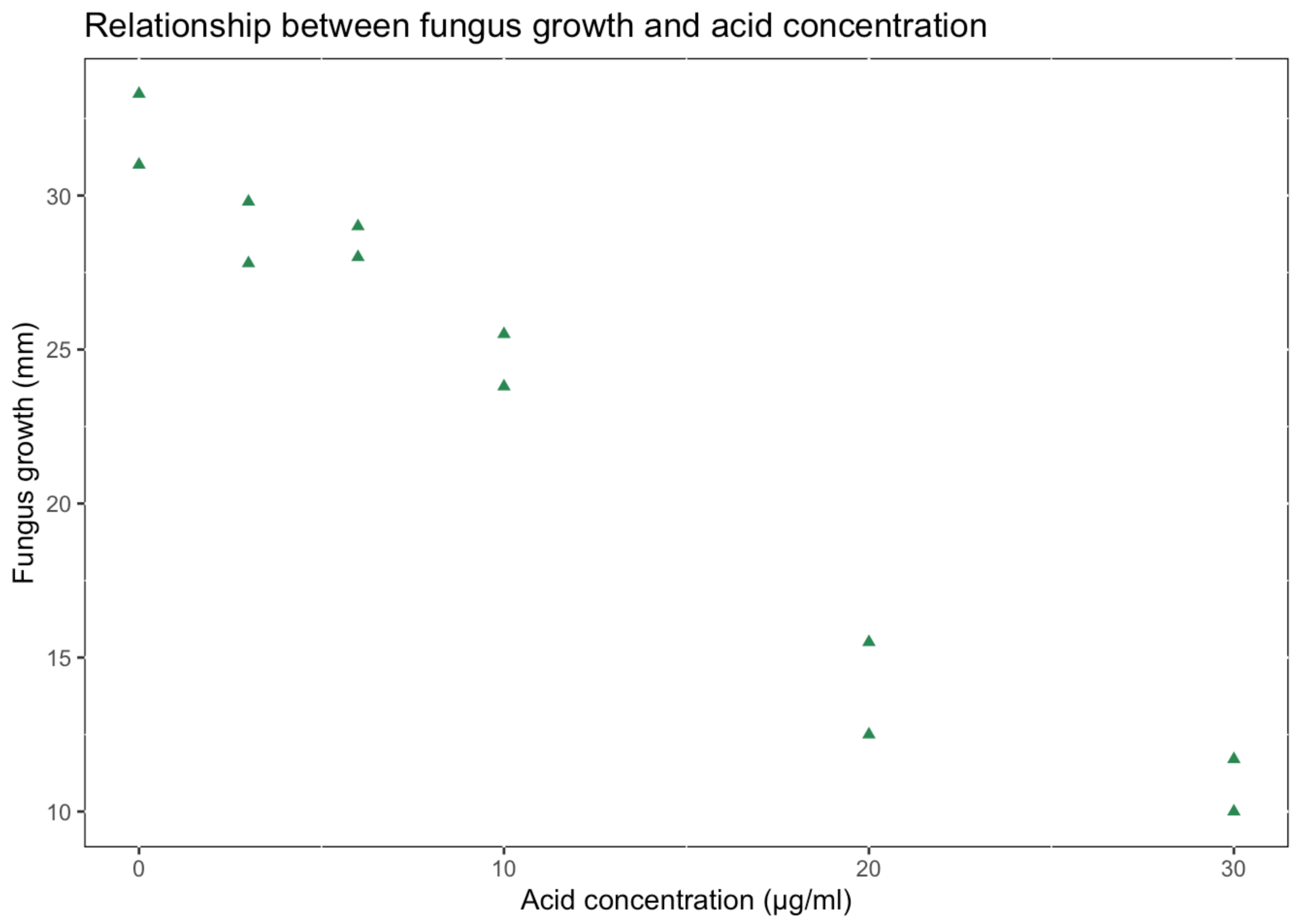
```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point(pch=17, col="seagreen")+
  labs(title="Relationship between fungus growth and acid concentration",
       y="Fungus growth (mm)",
       x="Acid concentration (μg/ml)")
```



The `labs()` command changes the names of the labels. The `x=` changes the x axis label, the `y=` changes the y axis label and the `title=` places a heading above your graph.

I personally don’t like the grey background, so I usually remove this with another layer:

```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point(pch=17, col="seagreen")+
  labs(title="Relationship between fungus growth and acid concentration",
       y="Fungus growth (mm)",
       x="Acid concentration (µg/ml)")+
  theme(panel.background = element_rect(fill = "white", colour = "black"))
```



The `theme()` command allows you to change aspects of the plot, like the background colour, thickness of the axis lines, appearance of the axis labels and axis ticks, placement of legends and other modifications. Check out this resource (https://ggplot2.tidyverse.org/reference/theme.html) if you want to explore more.

Now that we have made our plot, we should discuss features of the plot; namely the relationship between `acid` and `fungus` . Remember when discussing a relationship between numerical variables, you should address 3 points:

1. If the relationship appears linear (or non-linear).
2. The strength of the relationship (weak, moderate, strong, etc).
3. The direction (positive or negative)

Take a moment to **describe the relationship between `acid` and `fungus` .**

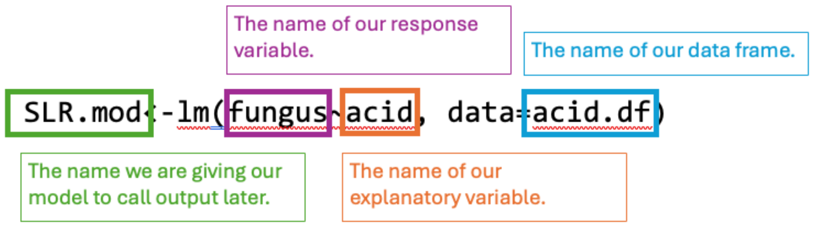
▶ **CHECK**

4.1.3 Fit an SLR model and look at output

Now that we have determined that our relationship does look linear between `fungus` and `acid` , we should fit a linear model. The first step is to define the linear model with:

```
SLR.mod<-lm(fungus~acid, data=acid.df)
```

Lets have a closer look at this code. `SLR.mod` is the name we are going to create to store all the information from the model. The `lm()` command says to create a linear model. We also need to specify the variables in our model and the data that we want to use.


The name of our response variable.
The name of our data frame.
The name we are giving our model to call output later.
The name of our explanatory variable.

```
SLR.mod<-lm(fungus~acid, data=acid.df)
```

Call output from the model using:

```
# p-value for the addition of acid concentration as a predictor for fungus growth.
anova(SLR.mod)
```

```
## Analysis of Variance Table
##
## Response: fungus
##           Df Sum Sq Mean Sq F value  Pr(>F)
## acid       1    733     733     195 6.9e-08
## Residuals 10     38       4
```

```
# summary output. Gives r-squared value, coefficients of the model and p-value of predic
tors.
summary(SLR.mod)
```

```
##
## Call:
## lm(formula = fungus ~ acid, data = acid.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.285 -0.907  0.491  1.292  2.414
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.7816     0.8327    38.2  3.6e-12
## acid         -0.7498     0.0537   -14.0  6.9e-08
##
## Residual standard error: 1.94 on 10 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.946
## F-statistic:  195 on 1 and 10 DF,  p-value: 6.89e-08
```

```
#gives confidence intervals for the coeficients.
confint(SLR.mod)
```

```
##                2.5 % 97.5 %
## (Intercept) 29.926  33.64
## acid        -0.869  -0.63
```

Using the output above, **write your equation for the model.**

Remember the general model equation for a fitted SLR is defined as:

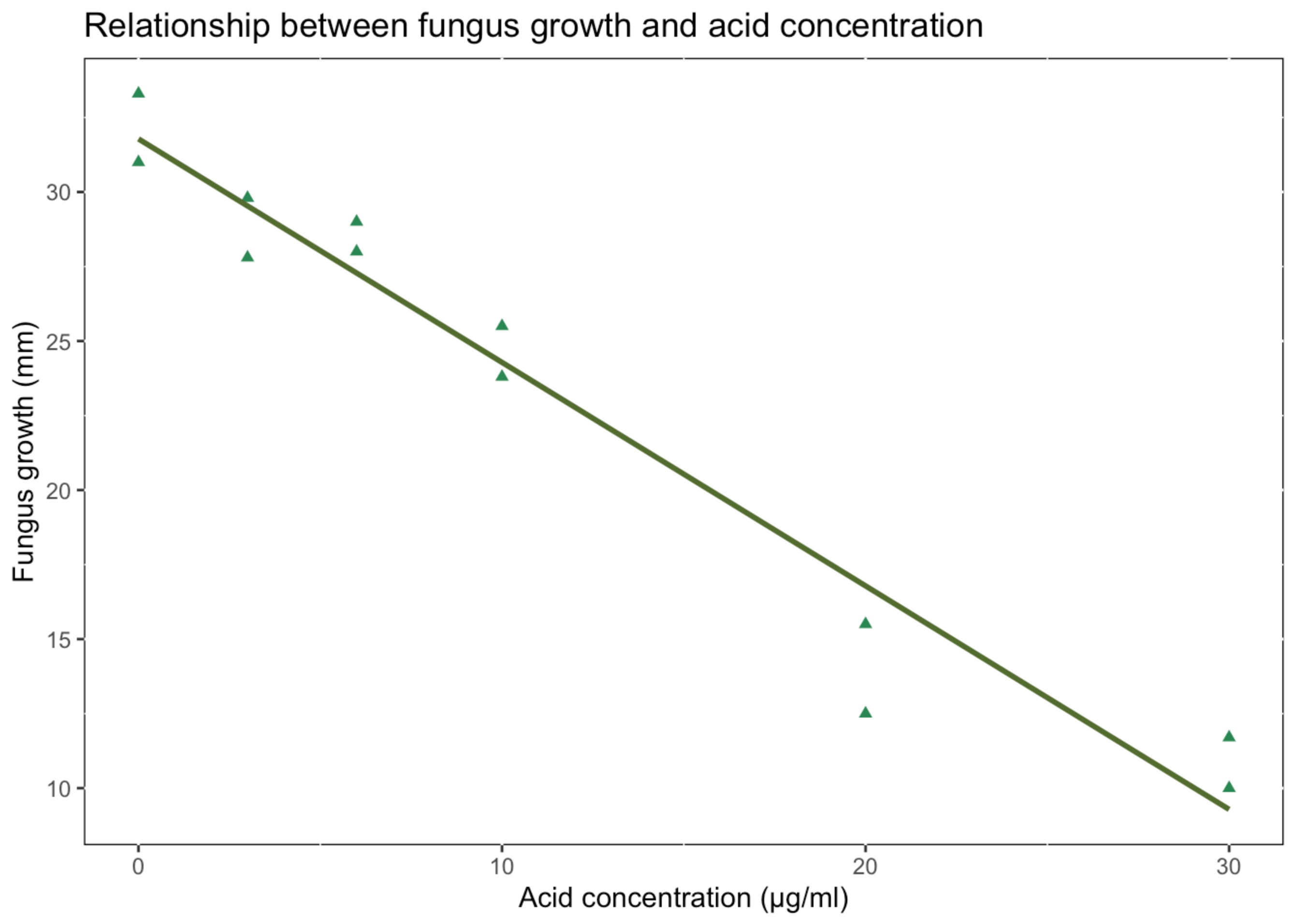
$$\hat{y} = b_0 + b_1 x$$

▶ **CHECK**

Lets create a plot with a line representing our model equation added.

```
ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point(pch=17, col="seagreen")+
  labs(title="Relationship between fungus growth and acid concentration",
       y="Fungus growth (mm)",
       x="Acid concentration (µg/ml)")+
  theme(panel.background = element_rect(fill = "white", colour = "black"))+
  geom_smooth(method="lm", col="darkolivegreen", se=F)
```

The command `geom_smooth()` adds a line. By default, this line will be applied using Loess (a non-parametric method that only STAT410 students will look at in their advanced reading topic). To specify we want to use SLR instead, we include `method="lm"` . The `col=` changes the colour of the line and `se=F` removes the confidence interval bands. If you want to add the confidence interval bands, simply change to `se=T` .



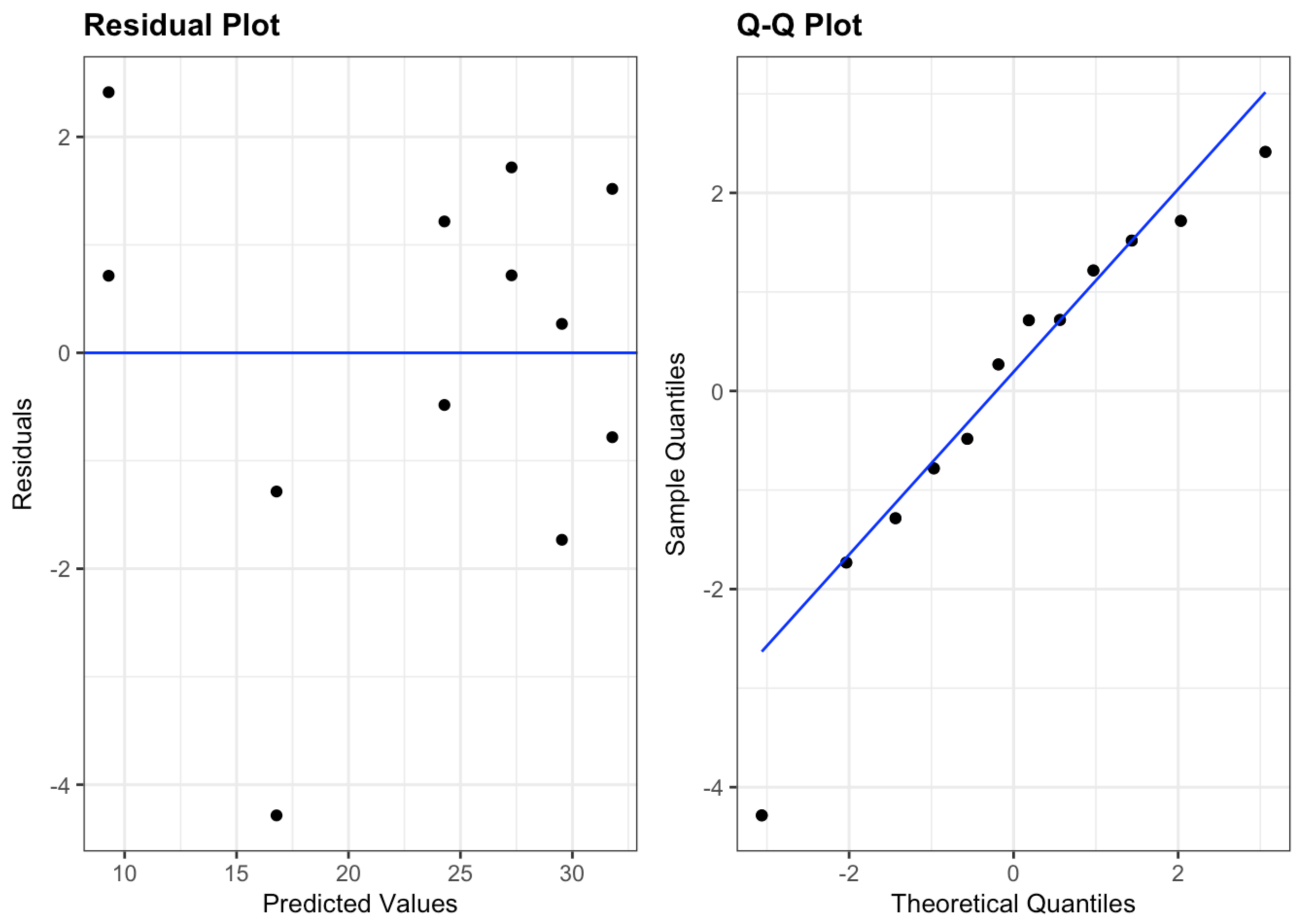
4.1.4 Check the Assumptions of the SLR

Before we get too far into interpreting the rest of the output from our model, we should check the assumptions of our model to make sure that an SLR is the appropriate model choice. To do this we will create a number of diagnostic plots using the `resid_panel()` function in the `ggResidpanel` package:

NOTE: You will need to first install the `ggResidpanel` package.

▶ **HINT**

```
library(ggResidpanel)
resid_panel(SLR.mod, plots=c("resid","qq"))
```



There are 4 assumptions that need to be checked for an SLR:

1. Independence of the observations
2. A mean centered on 0 (in STAT100 or QMER100 you might have heard it referred to as the relationship being linear).
3. Constant variance of the residuals
4. Normally distributed residuals

The second, third and fourth condition can be checked using the diagnostic plots above and are generally expressed as:

$$\epsilon \sim N(0, \sigma^2)$$

Independence can be assessed based on the data collection method.

We will look at some other types of diagnostic plots in later weeks, but for now we will focus on the residual plot and the Q-Q plot.

Use the diagnostic plots to **check that the conditions for this model have been satisfied.**

▶ **CHECK**

Determining if assumptions have been met can be tricky initially, especially for models created using a small number of observations. In our example, both of our diagnostic plots look fine. However, if you require extra support for your decision, you can run formal hypothesis tests with p-values on the residuals to check for normality using a Shapiro-Wilks test and to check for constant variance using the Breusch-Pagan test. The command `shapiro.test()` will conduct a Shapiro-Wilks test and the command `ncvTest()` in the `car` package will conduct a Breusch-Pagan test.

NOTE: you will need to first install and then load the `car` package into R in order to use `ncvTest()`.

▶ **HINT**

For both the Shapiro-Wilks and Breusch-Pagan test, the null hypothesis is that the residuals are normally distributed and have constant variance respectively. Keep in mind that you can never confirm a null hypothesis, only say that you do not have enough evidence (at this point!) to reject it.

```
# test for normally distributed residuals
 shapiro.test(SLR.mod$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  SLR.mod$residuals
## W = 0.9, p-value = 0.4
```

The p-value ($p = 0.4$) is >0.05, indicating that the residuals are normally distributed.

```
#test for constant variance
ncvTest(SLR.mod)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.67, Df = 1, p = 0.2
```

The p-value ($p = 0.2$) is >0.05, so the residuals have constant variance.

NOTE: There is some controversy surrounding the use of these hypothesis tests by themselves, so should only be used in support of the decision based on your plot. You should not use these tests without the diagnostic plots. The paper by Shatz (2023) (https://link.springer.com/article/10.3758/s13428-023-02072-x) presents some cautionary examples.

4.1.5 Use the model for predictions

Now that we have our model and it is an appropriate choice for the relationship we are investigating, we can use our model for prediction.

Here we **predict the mean fungal growth when acid concentration is equal to 15 µg/ml.**

The code below gives us our fitted prediction along with the upper and lower confidence bands for a mean response.

```
# predict the mean fungal growth and
# the 95% CI when acid conc. = 15
# Note use of new=data.frame to indicate values of X
preds.CI<-predict(SLR.mod, new=data.frame(acid=15), interval="confidence",level=0.95)
preds.CI
```

```
##    fit  lwr  upr
## 1 20.5 19.2 21.8
```

Note the part of the code `new=data.frame(acid=15)` . This specifies the new value that we want to make the prediction for. In this case we are predicting fungal growth when acid concentration is 15µg/ml. If we wanted to predict for a different value of acid concentration, we would simply change 15 to the new value of interest.

If we are interested in an individuals response, rather than a mean response we change `interval="confidence"` to `interval="prediction"` :

```
preds.PI<-predict(SLR.mod, new=data.frame(acid=15), interval="prediction",level=0.95)
preds.PI
```

```
##    fit lwr upr
## 1 20.5  16  25
```

NOTE: The fitted value is the same in both predictions as this is the point on the fitted line for the given x value (`acid` =15). However, the interval bands are wider for the prediction interval to allow for the error of predicting for a single individual.

Write interpretations of both the confidence interval and prediction interval for fungus growth when acid concentration is equal to 15 µg/ml.

▶ **CHECK**

4.1.6 Write an informative conclusion

Finally, we should **write a conclusion detailing our main findings from our output.**

So far we have only looked at the coefficents from our model to define our model equation. Other output we should look at include:

- The global utility p-value to assess the overall usefulness of the model.
- The p-value for b_1 to see if `acid` is a significant predictor of `fungus` .
- The r^2 to see how much of the variability in `fungus` is explained using a model with `acid` as the predictor and,
- The confidence intervals for b_1 to give us a range of where the true value of β_1 might lie (with 95% confidence). Remember b_1 is the sample estimate of our population parameter β_1.

First lets check the global utility p-value with the `anova()` output.

```
## Analysis of Variance Table
##
## Response: fungus
##           Df Sum Sq Mean Sq F value  Pr(>F)
## acid       1    733     733     195 6.9e-08
## Residuals 10     38       4
```

The p-value ($p = 6.9X10^{-8}$) is <0.05, so the model has good global utility and is useful in modelling fungus growth.

Next lets look at the p-value for `acid` with the `summary()` output.

```
##
## Call:
## lm(formula = fungus ~ acid, data = acid.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.285 -0.907  0.491  1.292  2.414
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.7816     0.8327    38.2  3.6e-12
## acid         -0.7498     0.0537   -14.0  6.9e-08
##
## Residual standard error: 1.94 on 10 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.946
## F-statistic:  195 on 1 and 10 DF,  p-value: 6.89e-08
```

The p-value for `acid` ($p = 6.9X10^{-8}$) is <0.05, so `acid` is a useful predictor of `fungus`.

Now we can interpret the r^2 using the same `summary()` output. The r^2 value is 0.951 which indicates that 95% of the variability in `fungus` is explained by the model that contains `acid`.

Finally, we can interpret the confidence interval for b_1 with the `confint()` command.

```
##              2.5 % 97.5 %
## (Intercept) 29.926  33.64
## acid        -0.869  -0.63
```

The confidence interval for `acid` is (-0.869, -0.63), so we can say, with 95% confidence, when acid concentration increases by 1 µg/ml, fungal growth decreases between 0.63 and 0.87 mm on average.

We could scale this interpretation up to make it more meaningful. If we multiply all the numbers in the interpretation by 10, we could say, with 95% confidence, when acid concentration increases by 10 µg/ml, fungal growth decreases between 6.3 and 8.7 mm on average. This is more meaningful as 6.3 and 8.7 mm is generally easier for your reader to understand than 0.63 and 0.87 mm.

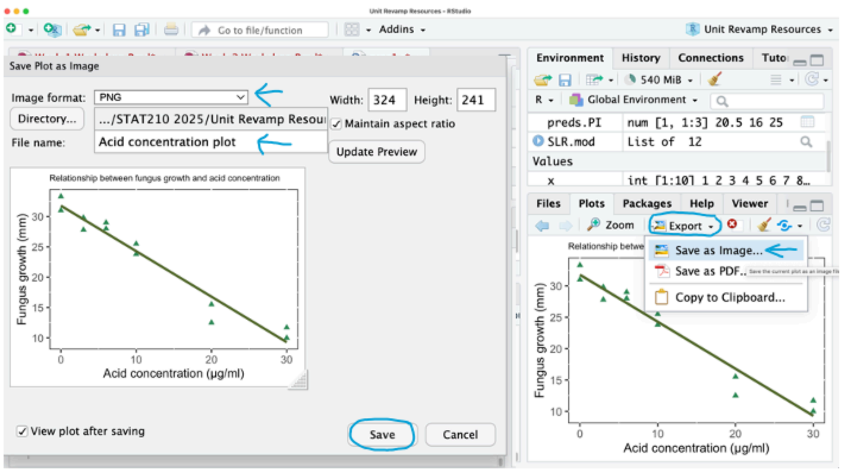
NOTE: The negative values in the confidence interval indicate a *decrease* in fungal growth. If the confidence interval values were positive, this would indicate an *increase*.

5 Saving plots and compiling your output

5.1 Saving plots

1. To save the scatterplot, ensure that the scatterplot is the visible plot in the bottom right hand window of R Studio by clicking on the right arrow at the top of the Plots window to scroll through your plots. Then click on Export > Save Plot as Image…, select a file type (e.g., PNG or JPG) and give the file a name: Acid concentration plot.png, say. This will be automatically saved to the working directory.

Similarly save the diagnostics plots. Select Export > Save Plot as Image… select a file type (e.g., PNG or JPG) and give the file a name.



Save Plot as Image

Image format: PNG

Directory... .../STAT210 2025/Unit Revamp Resou

File name: Acid concentration plot

Width: 324 Height: 241

✓ Maintain aspect ratio

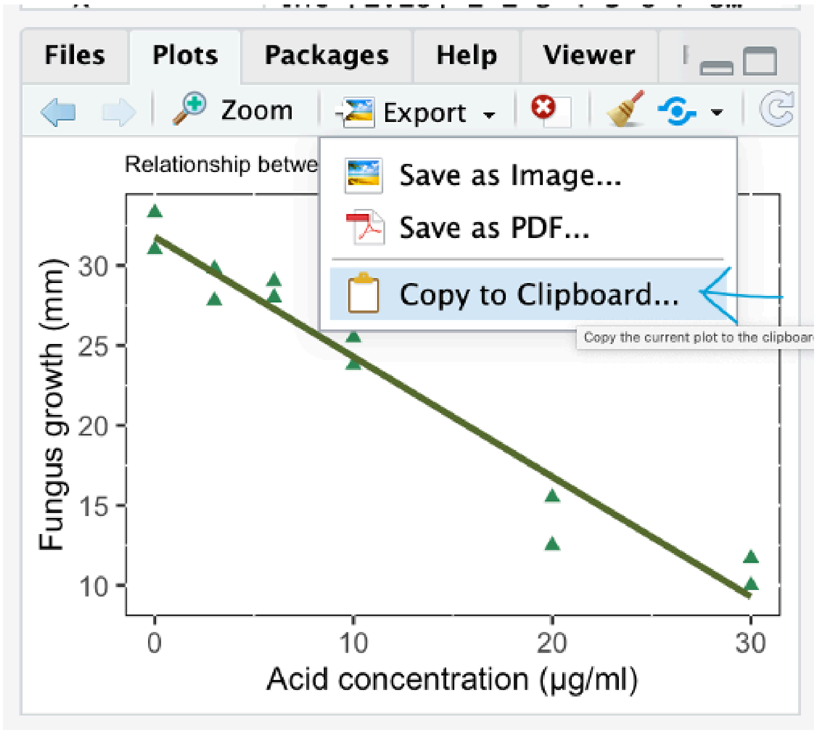
Update Preview

✓ View plot after saving

Save Cancel

```
preds.PI   num [1, 1:3] 20.5 16 25
SLR.mod    List of  12
Values
x          int [1:10] 1 2 3 4 5 6 7 8…
```

2. You can also save the plot to the clipboard and simply paste it directly into a WORD document: Select Export > Copy to Clipboard… and then click on Copy Plot.



3. Another way to export your plots is to use the `ggsave()` command.

```
ggsave("acid diagnostic plot.png")
```

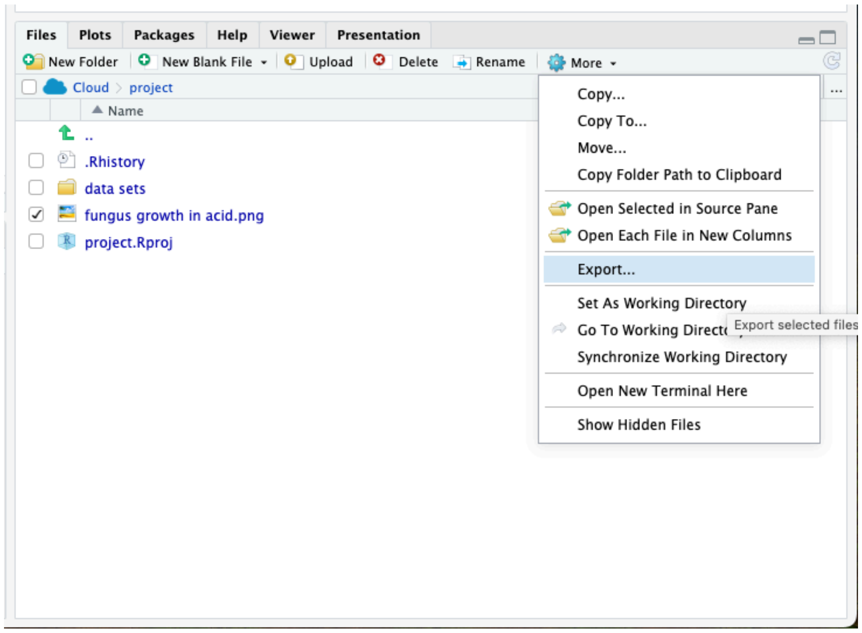
By default, the command will save the last plot you produced. You can also specify the plot to save by assigning plots to a name using the `<-` . In the example below, I have assigned my scatterplot of `acid` and `fungus` to `p` and then saved `p` in the `ggsave()` command.

```
p<- ggplot(data=acid.df, aes(x=acid, y=fungus))+
  geom_point(pch=17, col="orange3")+
  labs(title="Relationship between fungus growth and acid concentration",
       y="Fungus growth (mm)",
       x="Acid concentration (µg/ml)")+
  theme(panel.background = element_rect(fill = "white", colour = "black"))+
  geom_smooth(method="lm", col="brown", se=F)
ggsave("fungus growth in acid.png", plot = p)
```

The plot should now be saved in your STAT210 or STAT410 folder. If you wanted it to be saved somewhere else, you can also specify where using the `path=` function. Here I have told R to save my plot in the data sets folder (Remember that by default R will "look" in your project folder).

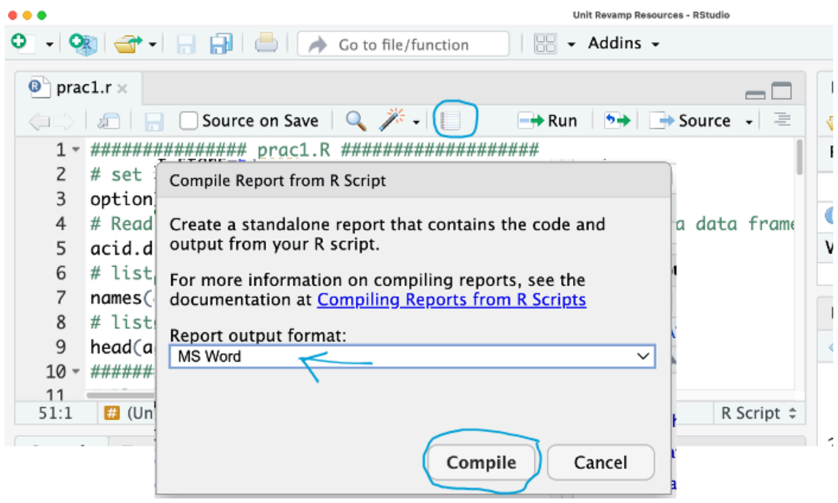
```
ggsave("acid diagnostic plot.png", path="data sets")
```

If you are working in the browser version, you can still save your plot and copy to clipboard in the same way as above. If you save your plot though, you will need to export it to your computer in order to use it. To do so, once the file is saved, tick the box next to it, select "More", "Export…" and then "Download". The file will then be downloaded to your downloads folder, where you can move it to wherever you like (eg your STAT210 or STAT410 folder!).



5.2 Producing a Word document

If your script file is running without error then you can click on Source - next to Run - at the top right of the script window. If you get an error then you need to make corrections ensuring that you do not have any incomplete or erroneous commands. Once your script file can be "Sourced" without error, then you can send all of your output to a WORD document by clicking on the page icon. Select the MS Word format and click on Compile.



```
############### prac1.R ##################
# set
option
# Read                                   a data frame
acid.d
# list
names(
# list
head(a
######
```

Compile Report from R Script

Create a standalone report that contains the code and output from your R script.

For more information on compiling reports, see the documentation at Compiling Reports from R Scripts

Report output format: MS Word

Compile Cancel

NOTE: This will not be a suitable format for assignment submission (refer to presentation guidelines) but it is a useful reference document. You may wish to use this output as a starting point for an assignment- editing before final submission.

6 Extra Example: Timber Volume of Black Cherry Trees

If you have time!

The `trees` dataset in R contains measurements of the diameter (`Girth` in inches), `Height` (ft) and timber `Volume` (cubic ft) of timber in 31 felled black cherry trees.

Source: Forest Mensuration. H. A. Meyer (1953). Penns Valley Publishers, Inc.

Your task is to **fit a simple linear regression to predict the timber `Volume` of cherry trees using either `Girth` or `Height` as explanatory variables.**

As part of fitting your model:

- Do some exploratory analysis using `ggplot()` .
- Fit an SLR using the `lm()` command and write the model equation. You might also want to make a plot including your linear model.
- Check assumptions of your model using the `resid_panel()` command.
- Write a meaningful conclusion using the output from the `anova()` , `summary()` and `confint()` commands.
- Predict mean timber `Volume` when `Height` of the cherry trees are 83 ft. Alternatively, predict mean timber `Volume` when `Girth` of the cherry trees are 9 inches.

NOTE: Your final prediction will depend on which model you chose to fit initially. For extra practice, you could fit both!

▶ **CHECK**

7 You’re Finished!

Have a funny cartoon for your efforts!

WHERE ARE YOU OFF TO, GARY?
SOME SCIENTISTS STUCK THIS TRACKER THINGIE ON ME, SO I'M SWIMMING TO BELGIUM, JUST TO MESS WITH THEM.
HA! CLASSIC GARY!
TOM GAULD for NEW SCIENTIST