

Guide to creating a server for online *R* experiments using *psychTestR*

1	INTRODUCTION.....	2
1.1	PSYCHTESTR	2
1.2	R.....	2
1.3	INFLUENCING GUIDES	3
1.4	SERVERS.....	4
1.5	OPERATING SYSTEMS	5
2	CREATING YOUR AWS SERVER.....	6
3	ACCESSING YOUR SERVER	9
3.1	OSX INSTRUCTIONS	9
3.1.1	<i>Installing Homebrew to your Mac computer</i>	<i>9</i>
3.1.2	<i>Accessing the Ubuntu server in OSX.....</i>	<i>9</i>
3.2	WINDOWS 10 INSTRUCTIONS.....	12
4	INSTALLING R AND OTHER PACKAGES ONTO YOUR SERVER	14
4.1	INSTALLING R	14
4.2	INSTALLING LIBRARIES REQUIRED FOR THE DEVTOOLS PACKAGE – OSX USERS.....	15
4.3	INSTALLING LIBRARIES REQUIRED FOR THE DEVTOOLS PACKAGE – WINDOWS 10 USERS	15
4.4	INSTALLING THE PACKAGES DEVTOOLS, SHINY, AND PSYCHTESTR.....	16
4.5	INSTALLING SHINY-SERVER	17
4.6	CHECKING YOUR SHINY-SERVER WEB URL	17
5	TROUBLESHOOTING	19
5.1	POSSIBLE ERRORS WHEN INSTALLING SHINY.....	19
5.2	ACCESSING THE LOGS FOR TROUBLESHOOTING	19
5.3	POSSIBLE ANTICONF ERRORS WHEN INSTALLING PACKAGES SUCH AS DEVTOOLS	20
6	ACKNOWLEDGEMENTS	21

1 Introduction

This guide was created by Anthony Chmiel, a member of the Western Sydney University MARCS Online Platforms Committee. The committee was formed in response to the impact of COVID-19 on data collection, and was aimed to help staff and students at Western Sydney University continue with data collection during social isolation. This guide has been shared online with the hope that it will also be of use to external researchers.

By the end of this guide you will have created a free server with the latest version of the free program *R* installed on it, as well as several *R* packages. You will be able to use this server to run many online experiments through apps or interfaces that are designed for use with the *R* web platform *shiny*. Specifically, this guide aims to show you how to install one particular *R* package called *psychTestR*, which can be used to create your own experiment interfaces for online use.

Before starting this guide, please read the following introductory sections. Importantly, this guide will **not** show you how to create an experiment interface within *psychTestR* or a similar package. To learn how to do this, I recommend that you follow the links in Section 1.1.

1.1 psychTestR

psychTestR was created by Peter M. C. Harrison, from the Max Planck Institute for Empirical Aesthetics. *psychTestR* can be downloaded for free from Peter's 'github' website (<https://github.com/pmcharrison/>). You will also find it useful to visit the *psychTestR* website (<https://pmcharrison.github.io/psychTestR/>). The *psychTestR* website contains an overview of the package, a guide on how to create online experiments within the interface, and examples of experiments that you can demo, download, and modify if you choose. Once you install a standalone version of *R*, I recommend that you download the *psychTestR* package and also open the Readme file within.

See, also Peter's overview paper (<https://psyarxiv.com/dyar7/download>). *psychTestR* has been used in large-scale experiments such as the current version of the Gold-MSI music aptitude test (<https://www.gold.ac.uk/music-mind-brain/gold-msi/>). For further information, see publications on the Gold-MSI *Melodic Discrimination Test*¹, *Beat Perception Test*², and *Mistuning Perception Test*³.

1.2 R

Before you begin this guide, if you do not have any experience with the program *R* then I recommend that you learn the basics. First, you will want to download the latest version of the software (which at the time of writing is 3.6.3). You can download *R* for free here (<https://www.r-project.org/>). I also recommend you download *R Studio*, which makes use of *R* easier (<https://rstudio.com/products/rstudio/download/>).

¹ Harrison, P.M.C., Collins, T. & Müllensiefen, D. Applying modern psychometric techniques to melodic discrimination testing: Item response theory, computerised adaptive testing, and automatic item generation. *Sci Rep* **7**, 3618 (2017). <https://doi.org/10.1038/s41598-017-03586-z>

² Harrison, P.M.C., Müllensiefen, D. Development and Validation of the Computerised Adaptive Beat Alignment Test (CA-BAT). *Sci Rep* **8**, 12395 (2018). <https://doi.org/10.1038/s41598-018-30318-8>

³ Larrouy-Maestri, P., Harrison, P.M.C. & Müllensiefen, D. The mistuning perception test: A new measurement instrument. *Behav Res* **51**, 663–675 (2019). <https://doi.org/10.3758/s13428-019-01225-1>

I recommend Barton Poulson's two-hour introductory video, available on Youtube (https://www.youtube.com/watch?v=_V8eKsto3Ug) as it will give you a strong basis from which to work from. Poulson's website (<https://bartonpoulson.com>) and Youtube channel have great resources to help get you started. I recommend that you download the standalone version of *R* first, and then download the packages *devtools*, and then *psychTestR* by separately running the two commands below. You may find it best to learn how to use *psychTestR* in the standalone version of *R* before you begin using it via a server.

```
install.packages("devtools")
devtools::install_github("pmcharrison/psychTestR")
```

From here you may wish to demo some of the Gold-MSI interfaces that have been created through *psychTestR*, to get a feel for them and how to access the system logs. Two of the Gold-MSI interfaces you can try out are the *Beat Perception Test*, also known as “cabat” and the *Melodic Discrimination Test*, also known as “mdt”.

To demo these two programs visit the sites:

cabat - <http://shiny.pmcharrison.com/cabat-demo>

mdt - <http://shiny.pmcharrison.com/mdt-demo>

To install either of the above packages, you would run the below command(s) in *R* after installing *devtools*. If *psychTestR* has not yet been installed, the below commands will automatically install it.

```
devtools::install_github("pmcharrison/cabat")
devtools::install_github("pmcharrison/mdt")
```

By downloading the separate Github files for each of these two packages from Peter's Github website (above), you can open the *R* Readme file and learn how to set each package up, access the logs and saved data, and so forth.

As is evident from the two Gold-MSI links provided above, *psychTestR* experiments provide ease of use for participants as they are only required to follow a URL; there is no set up required on their end. However, while the process of installing *psychTestR* in standalone *R* is straightforward, **there are many extra steps to be able to run *psychTestR* on a server (which enables you to run the experiments online)**. The rest of this guide outlines these steps.

1.3 Influencing guides

Sections of this guide have been taken from existing guides on similar topics, such as Charles Bordet's “The ultimate guide to deploy a Shiny app on AWS”⁴. While a number of steps remain similar to Bordet's guide, certain elements have been updated such as:

⁴ <https://www.charlesbordet.com/en/guide-shiny-aws/#>

1. Automatically installing the latest version of *R*, instead of a now-outdated version
2. Inclusion of some work-arounds that were found when trying to install certain packages such as *devtools* and *shiny* onto a server, and a troubleshooting guide at the end
3. Inclusion of additional content, such as *psychTestR*
4. A more linear structure, particularly aimed at people with no prior experience.

With this said, you may wish to also read Bordet's guide as it can be useful, particularly in regards to understanding **why** each step is taken.

1.4 Servers

In this particular guide, we will be using Amazon Web Services, or *AWS* (<https://aws.amazon.com/>) to host our server. I chose to use AWS based on Bordet's guide, and also as AWS have many server options to choose from including one in Australia (specifically, based in Sydney). The MARCS Online Platforms Committee have reported that regarding the Ethics Approval Process this may be beneficial for Western Sydney University (WSU) researchers, as the process strongly favours data that is stored in Australia instead of overseas storage. With this said, you will be able to find many other hosting alternatives. Based on your own research (and also the nature of the data you are collecting, as Australian storage is likely only to be a beneficial step for data considered personal) you may wish to choose an alternative server. In such a case the steps you take will likely be similar, but not identical to those I describe within the AWS website, although all steps within your server (once created) should be identical.

Western Sydney University researchers: I strongly encourage you to contact the Ethics Approval Committee early on in relation to whether or not your data should be stored in an Australian server. Additionally, as of June 2020 the MARCS Technical Team are investigating the feasibility of running *psychTestR* on a provided WSU server, which would have no associated cost. This feasibility is still to be determined.

Researchers from other institutes are encouraged to share this guide with their own IT staff, in the hopes that they might be able to provide an adequate server for hosting.

AWS provide a type of server (referred to as an "instance") that allows 12 months of free service, as long as you stay within the boundaries of their 'free tier'. The free server is an EC2, t2.micro instance. Please note at sign up AWS requires you to supply a valid credit card. If you breach the terms of the free tier use, you will incur a cost starting at around \$5-\$10 USD per month (although it increases from this cost). It is your responsibility to read and understand the terms of the free tier usage, and to monitor your usage and your billing. See the terms here (<https://aws.amazon.com/free/>)

At one point in this guide (Section 4.4) you will need to temporarily increase your t2.micro instance to a t2.small instance to install the package *devtools* and *shiny*, and then downgrade the server back to t2.micro once these installations are complete. This is because some of the required packages such as *dplyr* require a minimum of 2GB RAM (memory), whereas the t2.micro instance only has 1GB RAM. After installation of these packages you can return your instance to the free tier t2.micro instance. In testing we have noticed no detrimental effects to subsequently running *psychTestR* experiments on the t2.micro instance, although if



you run into problems in the future this may be the cause – for example, if your experiments require use of the package *dplyr*. Note that to install and run *R*, your server needs a minimum of 1GB of RAM. Many of the free alternatives to AWS contain <500MB of RAM. In my tests of multiple servers, in which I upgraded and subsequently downgraded all servers back to t2.micro, I incurred a total cost of less than 20 cents USD. However, a full year of running a t2.small server would cost approximately \$240 USD.

1.5 Operating systems

This guide was first written from experience working on various Apple computers, ranging from OSX El Capitan 10.11.6 to OSX Mojave 10.14.6. I had no issues creating a server on El Capitan, although at the time of writing at least one of the OSX programs used in this guide (*Homebrew*) does not officially support OSX versions before High Sierra. As such, I recommend using OSX Mojave or newer.

A guide for Windows 10 users has also been included, although this operating system has received far less road-testing than OSX. Apart from two Sections (3.1/3.2 and 4.2/4.3) the commands are identical for OSX and Windows users.

2 Creating your AWS server

Please note that the steps in Section 2 are almost identical to a section in Charles Bordet's guide, and Bordet's guide includes several screenshots. If you have trouble, visit the guide and look at the screenshots for additional help.

First, visit the Amazon web site (<https://aws.amazon.com/>) and create an AWS account. As noted earlier, you will need to provide a valid credit card in case you incur costs, although if you follow this guide correctly you should incur no costs (with that said, please check the disclaimer in Section 1.4, and understand that it is your responsibility to read and understand the AWS terms and to regularly check your billing summary). You will also need to supply a phone number to verify your account.

Once you have created and verified an account, sign in as a root user and type "EC2" into the "Find Services" search bar. This is the type of service that we will be using. You will be taken to the EC2 page. The first thing you will want to do is to decide in which country your data will be saved. This is shown in the top right of the screen, next to your username. My default was set to Ohio, although for WSU researchers I recommend changing this to an Australian server (see Section 1.4 for details concerning WSU Ethics Guidelines). Once your location is set, find the "Instances" tab on the left side of the screen and click the menu button labelled "Instances". The term instance refers to a server; if we have three servers then AWS will refer to this as three instances.

On the new page that opens up, click "Launch Instance", through which will create our server (instance). You are welcome to search which type of server you would like to create as all have different positives and negatives, although my recommendation (and Bordet's recommendation) is the Ubuntu 18.04 Server. For almost all cases, I will recommend the '64-bit (x86)' option. Make these selections, and then click the "Select" button that corresponds to that server.

On the next page we will decide what type of server we want to create. The only server that is eligible for the free tier is the t2.micro, and so that is what we will be selecting here. This server should run most *shiny* and *psychTestR* packages, although it only contains 1GB of memory. As detailed above in Section 1.4 we will run into an issue later on when trying to install the *R* package *devtools*, as this package requires at least 2GB of memory, although we will just temporarily move our server from a t2.micro to a t2.small for that installation. However, if you run into problems in the future (such as if you want to run *R* packages such as *dplyr* that require greater memory) then the t2.micro may not be an appropriate server for you. Select the t2.micro server, and click on "Next: Configure Instance Details". On the following page we will not make any changes, so click "Next: Add storage". You can have up to 30GB of storage on your server, and this server can run multiple packages, so my advice is to change the storage size from the default of 8GB to 30. Click "Review and Launch", followed by "Launch".

The first time we launch an instance, we must create a file known as a "Key pair". The Key pair, which is a .pem file, is kept on your computer, and without this file you cannot access the server. In the dropdown menu that states "Choose an existing key pair" select "Create a new key pair". Name your Key pair and then download it. Try not to make the name too long

as we will need to use this in code later. **Keep this .pem file safe** as without it you will need to re-create your server, and preferably store it in an easy location to access via commands. In this example I will be placing the file into my Documents folder although you are welcome to choose a different location. On the following screen, I recommend that you create Billing alerts, in case you incur costs. When we return to the “Instances” page you can see that our newly created server is now visible. You are free to name the server by clicking in the blank area below “Name”. You can also see the location of the server, and importantly the IP address of your server under *IPv4Public IP*.

N.b. The following step can be tricky to follow. You may find it easiest to also refer to screenshots in Charles Bordet’s guide, Section 4.

We will now make a small but important change to the server. At the bottom of the Instance dashboard, you should see a tab called “Description”. On the right side of this tab you should see a line labelled “Security Groups”, and next to this a link labelled “launch-wizard”. Click on “launch-wizard”, and you will be taken to a new page. On this new page find the “Inbound rules” tab down the bottom of the page, and click it. Once clicked, a new area should open below “Inbound Rules” (if not, look for three squares to the right of “Inbound Rules” that maximize/minimize this section of the page – these are highlighted in Figure 1). By clicking the filled in square you should be able to now see an expanded section below. Click “Edit inbound rules” and on the following page click “Add rule”. A new line labelled “Custom TCP” will be created. We want to edit this new line so that the port range is “3838”, and in the blank dropdown menu with a magnifying glass, select “0.0.0.0/0”. A screenshot (Figure 2) shows how it should look if correctly set up. Remember to click “Save rules”, and then click the “Instances” button on the left of the screen to return to our Instances Dashboard.

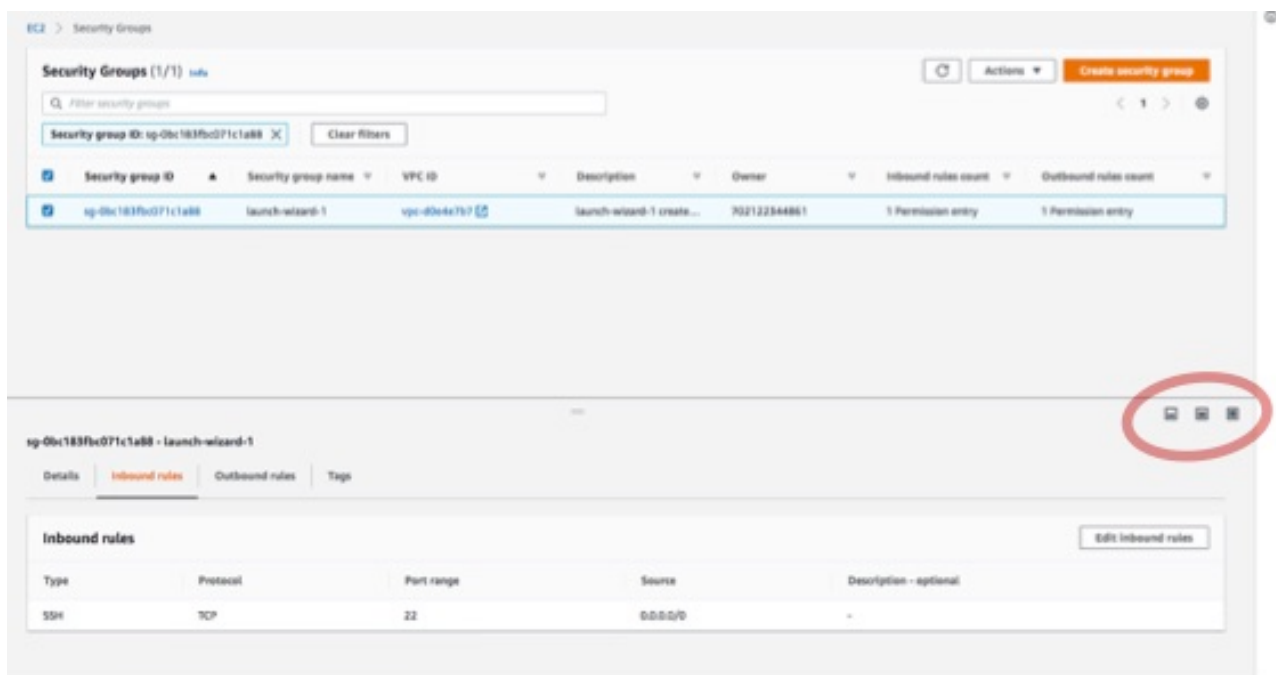


Figure 1. The three minimize/maximize buttons have been circled in this screenshot. If you can’t see the extra options at the bottom of the screen, click the right-side buttons.

EC2 > Security Groups > sg-0bc183fbc071c1a88 - launch-wizard-1 > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info		
SSH	TCP	22	Custom	<input type="text" value="Q"/>	<input type="text" value="0.0.0.0/0"/>	<input type="button" value="Delete"/>
Custom TCP	TCP	3838	Custom	<input type="text" value="Q"/>	<input type="text" value="0.0.0.0/0"/>	<input type="button" value="Delete"/>

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Figure 2. This figure details the correct setup for the inbound rules. If you don't follow this step, you won't be able to access your webpage once it is created (Section 4.6).

3 Accessing your server

3.1 OSX instructions

3.1.1 *Installing Homebrew to your Mac computer*

Before we access our server in OSX, we will install a useful piece of software called *Homebrew* that we will need later on. In OSX we will open the Terminal, which is a console window. You can do this by either 1) tapping “command” and “space bar” together, and typing “Terminal” into the spotlight search bar, or 2) by clicking Go -> Utilities -> Terminal (“Go” is accessed from the top of the desktop screen as long as Finder is selected). From here, copy/paste or type in the code depicted below, although copy/paste is the recommended approach. Note that you will **not** copy/type the “\$ ” at the start of this code. \$ is used to denote a new line of code (a new command). Once the code is entered, press the Enter key.

```
$ /usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

You will be prompted to enter your computer password. Once *Homebrew* is installed, be sure to close the Terminal window (or to quit Terminal entirely).

N.b. As the above code has been split across two lines, sometimes an issue can occur after copying and pasting it to Terminal. I include the same code below, in a smaller font; I suggest using this smaller font code for copying and pasting.

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

3.1.2 *Accessing the Ubuntu server in OSX*

In Section 2 we created an AWS instance. Now we will access this server. Start by returning to your Instances dashboard on the AWS site. Click on the “Connect” button⁵. A window will open titled “Connect to your instance”. You can view an example of this window in Figure 3. Importantly, you want to select the line of text that appears below “Example: ” (the line of text beginning with “ssh”), and copy it to your clipboard. Make sure that the text is copied exactly. We will need to use this line of text every time that we want to login to our server, so you may also wish to save this a text document for ease of access.⁶

⁵ This guide assumes you have not created multiple servers. If you have, you will need to make sure you select the server you want to connect to/make changes to by using the blue selection tool in the left-most column of the AWS instance dashboard.

⁶ Note that the text you copy here may change if you stop and then restart your server, or if you change the instance type. You may need to update the code in your text document from time to time.

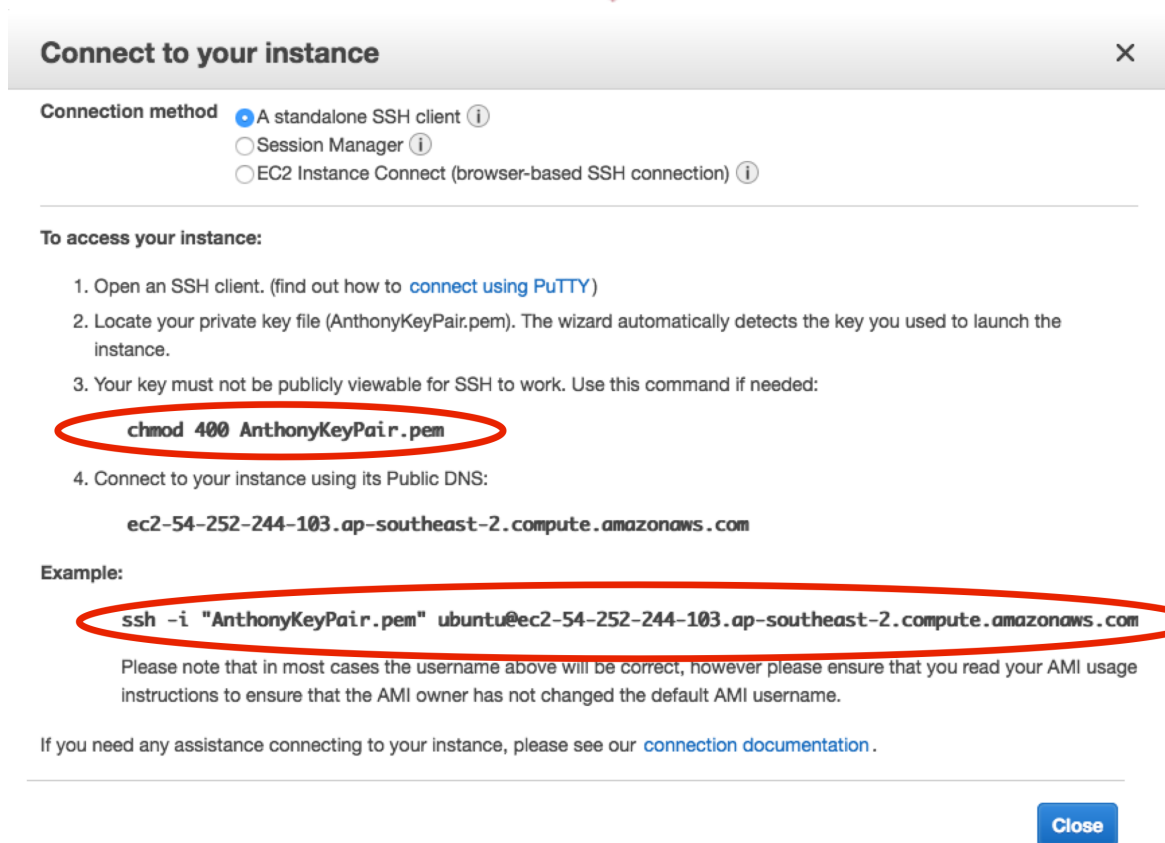


Figure 3. Connection window, depicting the example text that will be used to connect to our server in OSX.

The box below this paragraph contains three separate lines of code. Each line is to be run separately in Terminal. That is, that line of text is typed or pasted into Terminal, and then you press the enter key. You need to do this separately for each line of code in this guide. For the first line of code, “Documents” refers to whatever folder (directory) you stored your Key pair file in. “cd” refers to “Change directory”, meaning that in this case we are moving to the location of the folder where your Key pair is located. We will be using the “cd” command again so take note of it.

The second line of code is the primary text you copied from the AWS “Connect to your instance” window. Don’t use the example shown below, but instead copy your own text. The third line will only be required the first time that you access your server. This text has also been copied from the AWS “Connect to your instance” window” (Figure 3). Your text will be different, based on whatever you have named your Key pair file.

```
$ cd Documents/  
$ ssh -i "AnthonyKeyPair.pem" ubuntu@ec2-54-252-244-103.ap-southeast-  
2.compute.amazonaws.com  
$ chmod 400 AnthonyKeyPair.pem
```

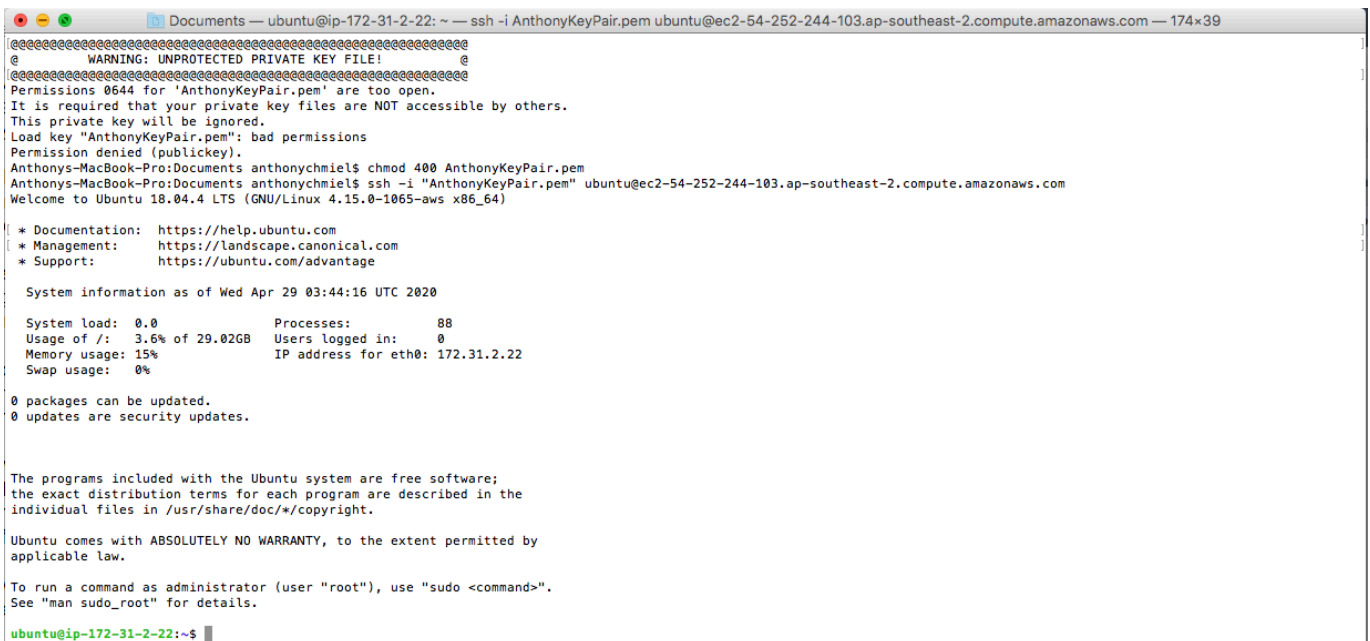
The first time you access the server you will need to provide permission. You will likely receive a message saying “Warning: Unprotected File Key”. Don’t worry, this is normal. This is where you run the third line of code, to enable access to the .pem file. Once the third line of code has been entered, you will need to run the second line of code again, and this time it will work. You have now successfully logged into your server! You know you have correctly done this step if the beginning of your new line is green, and refers to “Ubuntu@” followed by an IP address (see Figure 4 for my example).

NOTES: Remember that **you do not need to copy the “\$ ” from the beginning of each line of code.** For example, for line 1 just copy “cd Documents/”

For line 1, remember that “Documents” should refer to whatever folder you have placed the .pem file in. It does not have to be Documents, and can even be a folder within other folders (e.g., Documents/Folder1/NewFolder/), but the simpler the better.

For lines 2 and 3, remember to use your own codes from the AWS site instead of mine.

From now on, every time you want to access your server just enter the first two lines of code into Terminal, and you will log into your Ubuntu server.



```
Documents — ubuntu@ip-172-31-2-22: ~ — ssh -i AnthonyKeyPair.pem ubuntu@ec2-54-252-244-103.ap-southeast-2.compute.amazonaws.com — 174x39
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@      WARNING: UNPROTECTED PRIVATE KEY FILE!                                  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'AnthonyKeyPair.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "AnthonyKeyPair.pem": bad permissions
Permission denied (publickey).
Anthony-MacBook-Pro:Documents anthonychmiel$ chmod 400 AnthonyKeyPair.pem
Anthony-MacBook-Pro:Documents anthonychmiel$ ssh -i "AnthonyKeyPair.pem" ubuntu@ec2-54-252-244-103.ap-southeast-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-1065-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Apr 29 03:44:16 UTC 2020

System load:  0.0               Processes:    88
Usage of /:   3.6% of 29.02GB   Users logged in:  0
Memory usage: 15%              IP address for eth0: 172.31.2.22
Swap usage:   0%

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-2-22:~$
```

Figure 4. Once you have successfully logged into your Ubuntu server for the first time through OSX Terminal, it should look like this – it should show “Ubuntu@” followed by an IP address (in green).

3.2 Windows 10 instructions

N.b. These steps are almost identical to those listed in Charles Bordet's guide. Bordet's guide also contains screenshots that you may wish to use. Note that in Windows, we will not need to install *Homebrew* (which is the first step in the OSX Guide). While OSX users are able to access their server through *Terminal* (which is built into the operating system), Windows users will need to download an SSH client. Based on Charles Bordet's guide, we will use the free program *Putty* for this (<https://putty.org/>) although there are other clients you can use.

In Section 2 we created an AWS instance. Now we will access the server. Start by returning to your Instances dashboard on the AWS site. Click on the "Connect" button. A window will open titled "Connect to your instance". You can view an example of this window in Figure 5. We will need to copy the circled text that appears under Step 4 (your text will be different to that shown in the Figure).

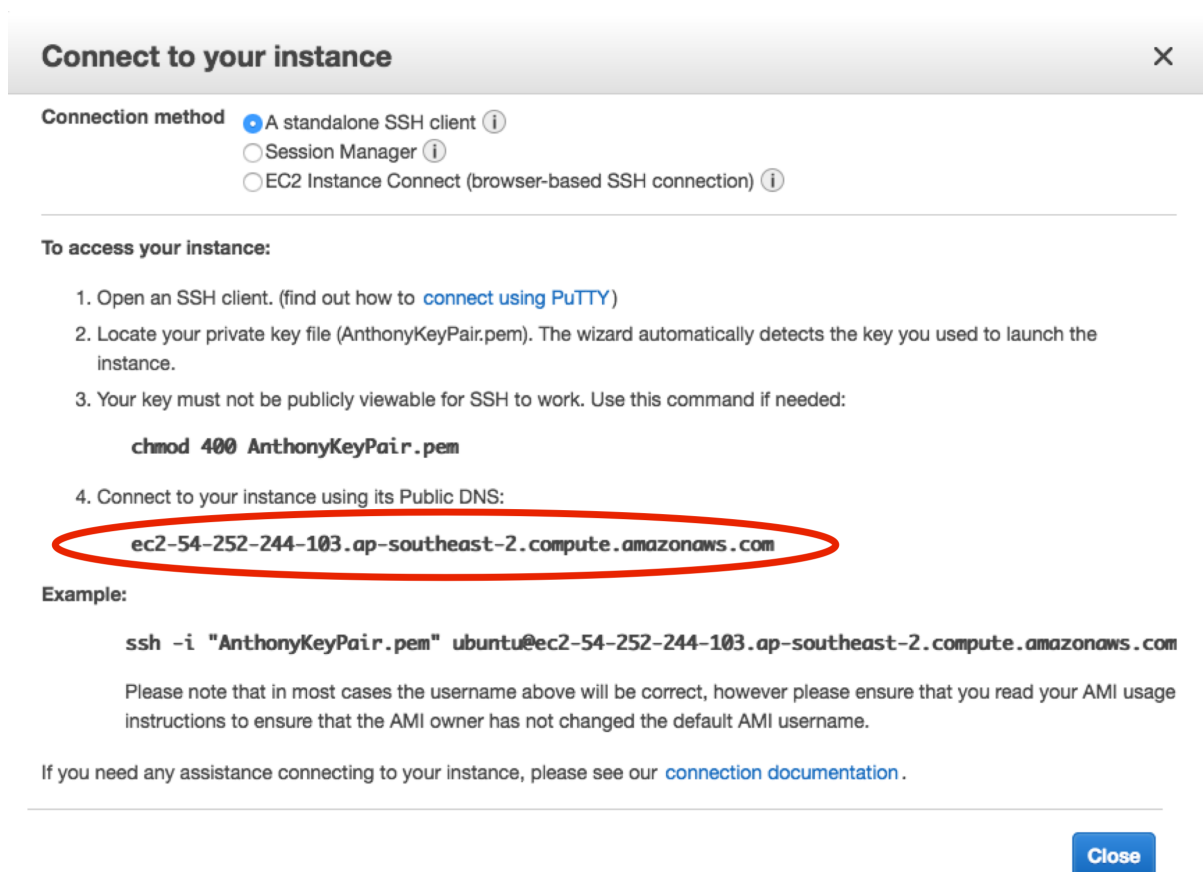


Figure 5. Connection window, depicting the example text that will be used to connect to our server in Windows 10.

Download *Putty* and install it. Two programs will be installed: *PuTTYgen* and *PuTTY*. If you can't find either program, search for them via the start menu. Open *PuTTYgen* and click the "Load" button. You will need to navigate to your Key pair (the .pem file). Note that it may not show up unless you select "show all file types". Once the file is loaded, ensure that



“RSA” is selected. Now save this as a private key file – this is the same as your .pem file but it will now be a .ppk file. Look at Charles Bordet’s screenshots if you have trouble.

Next, run *PuTTY*. In the first window, paste the text you copied from the AWS connection window (circled text in Figure 5) into the box labelled “Host name”. You also need to check that the “Port” is set to 22, and that “SSH” is selected. Next click “Connection” -> “SSH” -> “Auth” in the menu on the left side of the program. Look at Charles Bordet’s screenshots if you have trouble. You will find an option to Load the ppk file. When this is done, click “Open”. A console window will appear. Type “ubuntu” (without apostrophes) and select “yes” when prompted. You have now accessed your server!

Every time you want to access your server in future, you will need to open *PuTTY* and check that the settings are correct, before selecting “Open”.

Note: This guide assumes you have not created multiple servers. If you have, you will need to make sure you select the server you want to connect to/make changes to by using the blue selection tool in the left-most column of the AWS instance dashboard.

Note: The text you copy from the AWS Connection Window may change if you stop and then restart your server, or if you change the instance type. You may need to update the code in your text document from time to time

4 Installing *R* and other packages onto your server

When installing programs, packages, and libraries to your server, keep an eye out for error messages – these are helpful for troubleshooting. In case of errors, try to copy and paste the installation text in its entirety to a document as this will be important in trying to fix the error. Except for Sections 4.2 and 4.3, the commands in Section 4 should be identical for OSX and Windows 10.

4.1 Installing *R*

This section is identical for both OSX and Windows 10 users. Within your Ubuntu server, separately run the below commands. Again, remember not to copy “\$ ” (which denotes a new command) but only the text that occurs after this

```
$ sudo apt install apt-transport-https software-properties-common

$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9
$ sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu
bionic-cran35/'

$ sudo apt update
$ sudo apt install r-base

(for the above command, you will also need to type “Y” when prompted)

$ sudo apt install build-essential
```

As in Section 3.1.1, two of the above commands have been split across two lines, and sometimes an issue can occur after copying and pasting them. I include the same two commands below, in a smaller font; I suggest using these smaller font codes for copying and pasting.

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E298A3A825C0D65DFD57CBB651716619E084DAB9

sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu bionic-cran35/'
```

From here, you can check which version of *R* has been installed by using the command. At the time of writing, the latest version is 3.6.3 (“Holding the Windsock”).

```
$ R --version
```

To open *R* and then run commands from within *R* you need to use the below command. *Remember this command, as you will need it regularly.*

```
$ sudo R
```

From within *R*, you can now install packages with a command similar to the two shown below. To install other packages not listed, just replace “stringr” or “rmarkdown” with the name of any other package.

Every time you want to perform a task from within *R* (such as installing a package to *R*) you will need to make sure you run the command “sudo R” first. This is like opening up the application. Note: DO NOT install the packages *shiny* or *devtools* yet.

```
$ install.packages("stringr")  
$ install.packages("rmarkdown")
```

Once you have installed the packages *stringr* and *rmarkdown*, we will need to quit *R*. To do this type ‘q()’ without the apostrophes. There is no need to save the workspace image (so when prompted, type ‘n’ and ‘enter’ to quit). We are returned to the home directory of our Ubuntu server.

4.2 Installing libraries required for the devtools package – OSX users

Do not follow these steps if you are using Windows 10.

One of the most useful packages in *R* is *devtools*, and we will need this to help install *psychTestR*. Issues arise when trying to install *devtools* onto an Ubuntu server, although thanks to us installing *Homebrew* earlier, we should be able to work around this problem. The specific errors show up as “ANTICONF” errors during installation, that point to a number of missing libraries. Note that your earlier install of *Homebrew* was not to the Ubuntu server, but to your computer itself. This was intentional; *Homebrew* can now help us install the missing libraries to the Ubuntu server.

We will want to manually install the missing libraries separately. These libraries are *brew*, *openSSL*, *libssl-dev*, *libcurl4*, and *xml2*. To do this, separately run the commands below.

```
$ sudo apt install linuxbrew-wrapper  
$ brew update  
$ brew install openssl@1.1  
$ sudo apt-get update -y  
$ sudo apt-get install -y libssl-dev  
$ sudo apt-get install -y libcurl4-openssl-dev  
$ sudo apt-get install libxml2-dev
```

NOTE: you will be prompted to enter “control and D” when updating brew.

Be patient, especially for updating brew and installing openssl. Wait until the console returns to the usual ‘Ubuntu@’ followed by an IP address before you move onto the next command. If you run into an error, see Section 5.3.

4.3 Installing libraries required for the devtools package – Windows 10 users

Do not follow these steps if you are using OSX.

One of the most useful packages in *R* is *devtools*, and we will need this to help install *psychTestR*. Issues arise when trying to install *devtools* onto an Ubuntu server, although we should be able to work around this problem. The specific errors show up as “ANTICONF” errors during installation, that point to a number of missing libraries. We will want to manually install separate libraries. These libraries are *openSSL*, *curl*, *libcurl4*, *libssl-dev*, and *xml2*. To do this, run the commands below. Note that you will be prompted to enter ‘Y’ for the ‘upgrade’ command. If you run into an error, see Section 5.3.


```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install openssl
$ sudo apt install curl
$ sudo apt install libcurl4
$ sudo apt install libssl-dev
$ sudo apt-get install libxml2-dev
```

Be patient, as some of these may take time. Wait until the console returns to the usual ‘Ubuntu@’ followed by an IP address before you move onto the next command. If you run into an error, see Section 5.3.

4.4 Installing the packages devtools, shiny, and psychTestR

At this stage, we will need to temporarily upgrade our AWS instance. We will aim to do this as quickly as possible, as for each hour that server is upgraded you will be charged around 10 cents to your card that was used on sign up. The reason we are upgrading is that *devtools* will not install on a t2.micro server. I also recommend installing the package *shiny* at this point for two reasons. First, *shiny* sometimes (but not always) has installation issues on a t2.micro server. This is presumably also due to the lower available RAM. I have had minimal issues installing on a t2.small server. With that said, as there can be occasional issues installing shiny on a t2.small server, the alternative is to use *devtools* to install *shiny*. This might seem convoluted, but it is an effective workaround, detailed in Section 5.1.

To upgrade your instance in AWS, head to the instances dashboard. Remember that if you created more than one instance, you will have to select which instance you want to make changes to, using the left most column to select the instance. Before we upgrade the instance, we must temporarily stop it. This is done through the “Actions” menu. Click “Actions” -> “Instance State” -> Stop”. (On a related note, if you ever want to delete an instance, you would instead click “Terminate” in this menu, and after a few hours the instance will disappear. Terminated instances cannot be restarted, so you may wish to back up your server once it is completed).

After you have stopped the instance, it will take a few minutes for it to completely stop. Use the refresh button in the top right-hand corner to check the progress. In the same “Actions” menu we can now select “Instance settings” and then “Change Instance Type”. Here you can change your server from a t2.micro to a t2.small. Use the “Actions” -> “Instance State” -> “Start” buttons to now start your server again.

It will take a few minutes for your server to restart. Access your server by using the first two lines of code from Section 3.1.2 (get used to this step, as you will need to do it every time you want to access your server). It is possible that by stopping, upgrading, and re-starting your instance the code in the “Connect to your server” window will now be slightly different, so don’t rely on the code you pasted earlier, when first doing the steps in Section 3.

Now we will install the two packages *devtools* and *shiny* to R, as well as *psychTestR*:

```
$ sudo R
$ install.packages("devtools")
$ install.packages("shiny")
$ devtools::install_github("pmcharrison/psychTestR")
```

The above commands should work, although if there is an installation error with *devtools*, see Section 5.3, and if there is an installation error with *shiny*, see Section 5.1.

It is a good idea to check that *shiny*, *devtools*, and *psychTestR* are installed by running the following command from within *R*:

```
$ installed.packages()
```

IMPORTANTLY don't forget to now return to the AWS instance dashboard and return your server to a t2.micro instance (stop the server, downgrade, and start the server) or you will incur costs for every hour that it stays as a t2.small instance.

Also make sure to check your billing summary. If you have done this quickly, your costs should only be 1 cent, or even nothing!

4.5 Installing shiny-server

Make sure that you have quit out of *R*. To install *shiny-server*, it is best to navigate to the following web page (<https://rstudio.com/products/shiny/download-server/>) and to select your server type (Ubuntu, unless you have opted for a different instance type).

After clicking Ubuntu, scroll to the bottom of the page where you will see a code box containing three commands. I paste these commands below, and they will likely be identical or very similar to the code on the website, but it is best to copy these directly from the website to make sure you are installing the latest version of *shiny-server*. These three commands are run separately, and are run in your Ubuntu server but **not** in *R*. At the time of writing, the codes are as follows:

```
$ sudo apt-get install gdebi-core
$ wget https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-
1.5.13.944-amd64.deb
$ sudo gdebi shiny-server-1.5.13.944-amd64.deb
```

4.6 Checking your shiny-server web URL

If all steps so far have gone correctly, you will be able to type your own URL into a browser and see the generic *shiny-server* interface. To try this, go to your AWS instance dashboard and copy the IP Address listed under *IPv4Public IP*. In a new browser page copy your IP address into the url box, followed by a colon and then 3838, and press enter to see your new website! You are welcome to also put `http://` at the beginning of your URL.

As my IP address for this test server was 13.211.32.156, my URL ended up being:
`http://13.211.32.156:3838`

If you have correctly installed *shiny-server*, and correctly saved the inbound server rule in Section 2, then when you follow your URL you should be able to see a webpage titled “Welcome to Shiny Server”. You can see an example of the webpage in Figure 5. On the right-hand side of the page there should be two interactive plots. If the top plot is not

showing, then you have had an error when installing the package *shiny* within *R*. If you cannot see the bottom plot, then you have had an error when installing the *rmarkdown* package within *R*.

If *psychTestR* successfully installed (which you can check with the ‘`installed.packages()`’ command) then you are now ready to go to Peter Harrison’s websites, and start running and creating *psychTestR* experiments! **Best of luck!**

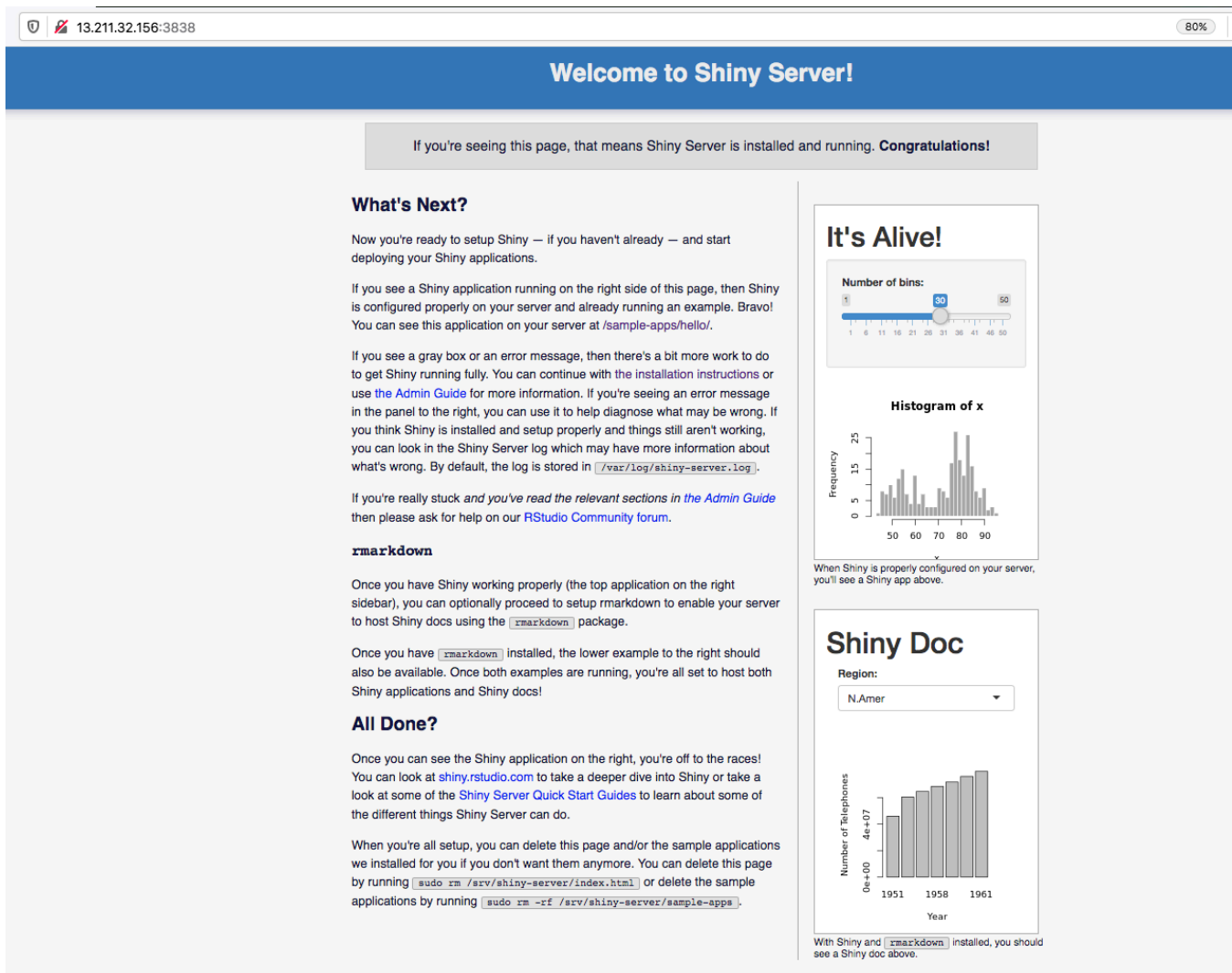


Figure 5. If you successfully install *shiny*, *shiny-server*, and *rmarkdown*, and also enable the inbound server rule in Section 2, your URL should look like this.

5 Troubleshooting

5.1 Possible errors when installing shiny

On occasional installations of *shiny* via the `install.packages("shiny")` command, *shiny* will encounter an error and subsequent attempts to install the package will show an error for one of the packages relating to *shiny*. This can be for a number of packages such as *dplyr* or *Rcpp*; in my example (Figure 6) it happened to be for *dplyr*. To fix this error, we first need to remove the directory for the problematic package(s). To do this, copy the below code, although substitute `'dplyr'` for the name of whatever package is causing the error. If multiple packages are causing errors, you will need to run separate commands for each package.

```
$ sudo rm -rf /usr/local/lib/R/site-library/00LOCK-dplyr
```

```
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (R.utils)
ERROR: failed to lock directory '/usr/local/lib/R/site-library' for modifying
Try removing '/usr/local/lib/R/site-library/00LOCK-dplyr'
Error: Failed to install 'psychTestR' from GitHub:
  (converted from warning) installation of package 'dplyr' had non-zero exit status
> █
```

Figure 6. A possible error that may occur when trying to install *shiny*. Follow the steps in this section to work around this error.

Once you have deleted the problematic directory/directories with the above code, try to install *shiny* by using the below command instead of the typical installation command (make sure *devtools* has already been successfully installed):

```
$ devtools::install_github("rstudio/shiny")
```

5.2 Accessing the logs for troubleshooting

When installing a package on your server, if you encounter an error then the first place to start investigating is in the messages listed for that installation (such as in Figure 6). If you cannot figure out the issue here, try looking in the server logs. To access the server logs, make sure you are logged into your Ubuntu server, but are not within *R*, and use the commands:

```
$ cd /var/log/shiny-server
$ ls
```

What we have done above is change directory to where the *shiny-server* logs are kept, and then the `ls` command is to list items. When I do this, I could see two items, although if you are testing you will likely have many items within this directory.



My two items were labelled “hello-shiny-20200501-060908-37975.log rmd-shiny-20200501-060908-35913.log”

To view one of these logs, use the command “sudo tail ” before the log title, such as in my example below:

```
$ sudo tail rmd-shiny-20200501-060908-35913.log
```

Press enter, and you will see the log. Note that it is normal for the log to end in “Execution halted”. There may be some useful information in the log, or there may not be. Also note that *psychTestR* contains its own log files, which are accessed through the interface itself (discussed briefly in Section 1).

5.3 Possible ANTICONF errors when installing packages such as devtools
When attempting to install packages such as *devtools*, you may find that the installation produces an error and that within the installation text an “ANTICONF error” is highlighted. You may have more than one ANTICONF error in the same attempt at a package installation. This means that a required library cannot be found. Our attempt to install extra libraries was aimed to solve this issue although it is possible that you are missing another library, or that one of the libraries in Section 4.2/4.3 was not successful. In such a case, try and identify all the missing libraries via the installation text files, and use the supplied library commands in Section 4.2/4.3 with the name of your missing package(s) substituted in, with separate commands. Some trial and error may be the best place to begin. You may also search online for existing ubuntu packages and libraries to help locate the correct command for that missing library.

6 Acknowledgements

I would like to extend my gratitude to Peter M. C. Harrison from the Max Planck Institute for Empirical Aesthetics, Frankfurt, for creating *psychTestR*, and for help in identify and solving issues that arose in the initial creation of this installation guide, as well as Daniel Müllensiefen.

Additionally, I would like to thank my colleagues in the MARCS Online Platforms Committee (which was formed in response to Covid-19) and the MARCS Technical Staff for their assistance with several matters, as well as my colleagues in the MARCS Active Minds Music Ensemble for help with beta testing of *psychTestR* experiments on my servers.

Dr Anthony Chmiel
The MARCS Institute for Brain, Behaviour and Development
Western Sydney University, Sydney, Australia
Active Minds Music Ensemble, and MARCS Online Platforms Committee

This guide was also developed for use in an ARC Project, *Maintaining active minds and bodies through older adult music education*

<https://www.westernsydney.edu.au/marcs>