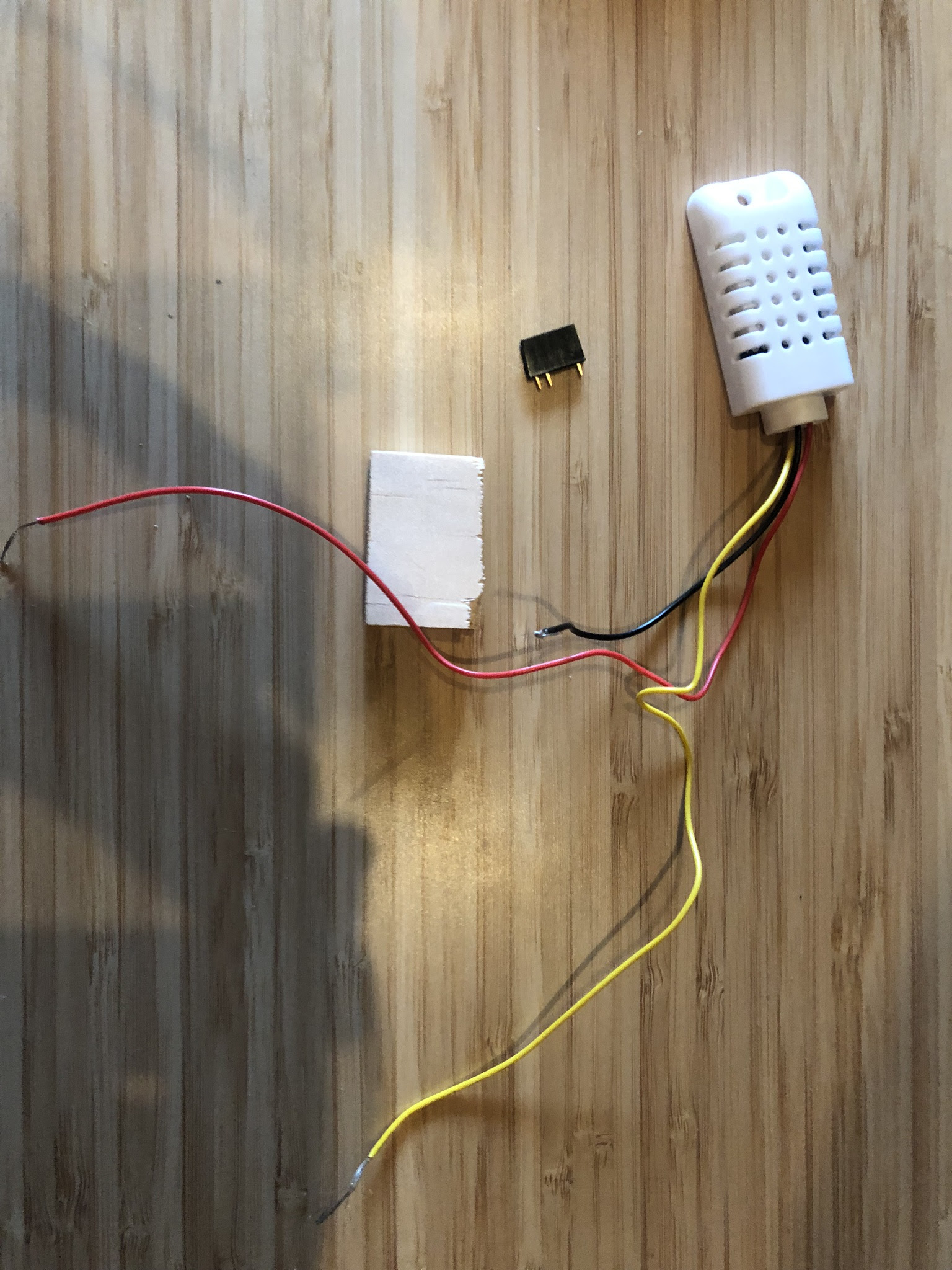
This tutorial is going to cover how to set up a temperature/humidity sensor with  
a raspberry pi. You will learn how to setup the sensor, a MYSQL server, and  
connect to the databse remotely in R. I will also do exploratory  
data analysis in R with the stored readings. A little bit of familiarity with linux, mysql servers,  
soldering skills, and R is helpful but not mandatory. The materials required  
are:

* Raspberry Pi with standard setup (SD card, case, etc.)
* Adafruit AM2302 (wired DHT22) temperature-humidity sensor
* Soldering Iron
* Female Pin Headers
* Small piece of wood



Materials to be used

*Materials to be used*



Final Product

*Final Product*

I mostly followed the tutorial found [here](https://www.instructables.com/id/Raspberry-Pi-Temperature-Humidity-Network-Monitor/).  
The majority of the work in this post is not using R. Instead of rebuilding  
everything, I wanted to build off of content that has made and sensor readings  
are handled a bit better by a low level language (C is used here).

**Install wiringPi**

We are going to start with the assumption that you have already set up the  
raspberry pi and soldered the sensor to the GPIO. An excellent repository already  
exists at <http://wiringpi.com/>. It provides a C interface  
to the GPIO that will prevent us from having to write any of the low level code.  
I’m going to SSH into my headless pi and install the wiringPi program with the  
following commands. In the examples, you can build and run a program to check  
if you are getting good readings.

git clone git://git.drogon.net/wiringPi

cd wiringPi

./build

cd examples

make rht03

./rht03

You should see readings from the sensor now. Type CTRL+c to quit the  
program.

**Set up a MYSQL Server**

In this step we are going to install MYSQL server and create some security  
around it. When you see ‘username’ or ‘password’ those are meant for you to  
replace with your own credentials. Instead of only using root to access  
the server with elevated priviliges, I am going to grant all privileges to  
a different user, but only when on the ‘localhost’, Essentially, you need to  
remote in currently to access the server with your ‘user’ credentials. Last,  
log back into the server with your new user identity for the next step.

sudo apt-get install mysql-server

sudo apt-get install default-libmysqlclient-dev

sudo mysql\_secure\_installation

sudo mysql

GRANT ALL PRIVILEGES ON \*.\* TO 'username'@'localhost' IDENTIFIED BY 'password';

\q

mysql -u 'username' -p

**Set up a Database**

Now, let’s create a database and a table with the time in UNIX integer time and  
two other columns for the sensor readings that we want to record.

create database Monitoring;

use Monitoring;

create table Climate (ComputerTime INTEGER UNSIGNED,Temperature DECIMAL(5,1), Humidity DECIMAL(5,1));

**Run the program to read and write data**

You will need to download this file:

[th.c](https://cdn.instructables.com/ORIG/FV1/SE8A/HOSLXFJA/FV1SE8AHOSLXFJA.c)

Change the 'root' and 'password'  
credentials to match the user that you set up earlier. You will need to copy  
over the Makefile and change some flags so that the program knows where to  
find some of the drivers that it needs.

cp wiringPi/examples/Makefile ~/raspberrypi/monitor/Makefile

sudo nano ~/raspberrypi/monitor/Makefile

Add the following lines to the file:

INCLUDE = -I/usr/local/include,/usr/include/mysql

LDFLAGS = -L/usr/local/lib,/usr/lib/arm-linux-gnueabihf -lmysqlclient -lpthread -lz -lm -lrt -ldl

Compile the program:

make ~/raspberrypi/monintor/th

You will now run the program that you altered and compiled. Use the & to  
run the program continuously in the background. The program will write the  
temperature and humidity every 60 seconds to the database.

./raspberrypi/monitor/th &

**Set up R**

This step is optional but good to have for troubleshooting. Later, I will be  
connecting remotely on my laptop instead of working in R on the raspberry pi.

sudo apt-get install r-base

sudo su - -e 'install.packages("DBI", "RMariaDB")'

**Check Database Connection and Query**

After a couple hours, you should have a good amount of data. I’m going to be  
connecting from my laptop so I’ll need to set up my user credentials with  
privileges to access over my LAN which to cover all ip addresses that get  
assigned use the % (wildcard) at the end of 192.168.1 (your LAN is  
already set to use this ip address numbering system).

GRANT ALL PRIVILEGES on \*.\* TO 'user'@'192.168.1.%' IDENTIFIED BY 'password';

FLUSH PRIVILEGES;

\q

Let’s make sure that the server has a port that we can access. If it’s not  
already in the my.conf file, open up the file with a text editor.

sudo nano /etc/mysql/my.cnf

Add the following lines to the file which opens up the default port 3306 for  
the mysql server and then bind to current IP address.

[msqld]

port=3306

bind-address=0.0.0.0

Restart the service for the changes to take effect.

sudo service mysql restart

You can log off the machine now. We don’t need to do anything else on the  
raspberry pi for now.

**Access the Data Remotely**

From RStudio on my laptop (while connected to my LAN), we’re going to open  
a connection to the database. You can specify the host as a DNS name that  
can be set up on your router’s administration portal or you can specify the  
IP address. I would recommend making the IP address static if you plan on  
using that method going forward. Since we stored the timestamp in UNIX  
integer form, we can convert it to POSIXct knowing that the origin of UNIX  
time is the start of the year 1970.

library(DBI)

library(ggplot2)

library(trstyles) # optional package for my styling of ggplot2 plots

con <- dbConnect(RMariaDB::MariaDB(), host = 'sensorpi',

user = 'pi', password = 'password',

dbname = 'Monitoring')

query <- 'SELECT ComputerTime, Temperature, Humidity FROM Climate'

readings <- dbGetQuery(con, query)

readings[['ComputerTime']] <- as.POSIXct(readings[['ComputerTime']],

origin = '1970-01-01 00:00:00')

**Plot the Data**

Now that we have the data, let’s plot temperature against time to see what has  
been going on.

ggplot(readings) +

geom\_line(aes(ComputerTime, Temperature)) +

scale\_y\_continuous(name = expression('Temperature ('\*degree\*'C)'),

sec.axis = sec\_axis(~.\*9/5+32,

name =

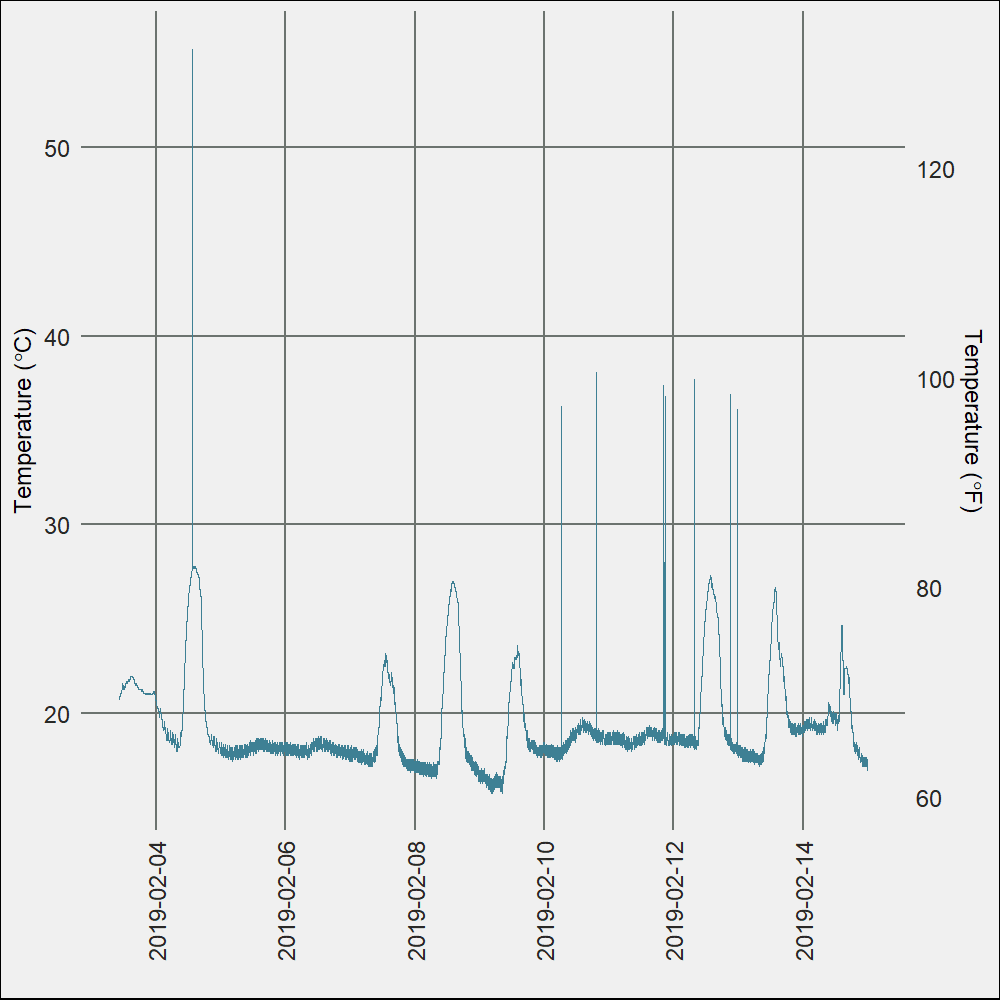
expression('Temperature ('\*degree\*'F)'))) +

scale\_x\_datetime(name = '', date\_breaks = '2 days') +

theme\_tr(base\_size = 18) +

theme(axis.text.x = element\_text(angle = 90, vjust = .5))

It looks good for the most part, but we definitely have some outlier readings.  
I can see probably outliers at 30 degrees Celsius. I’m going to cut those off  
and take a second look.



readings[['Temperature']][readings[['Temperature']] > 30] <- NA

ggplot(readings) +

geom\_line(aes(ComputerTime, Temperature)) +

scale\_y\_continuous(name = expression('Temperature ('\*degree\*'C)'),

sec.axis = sec\_axis(~.\*9/5+32,

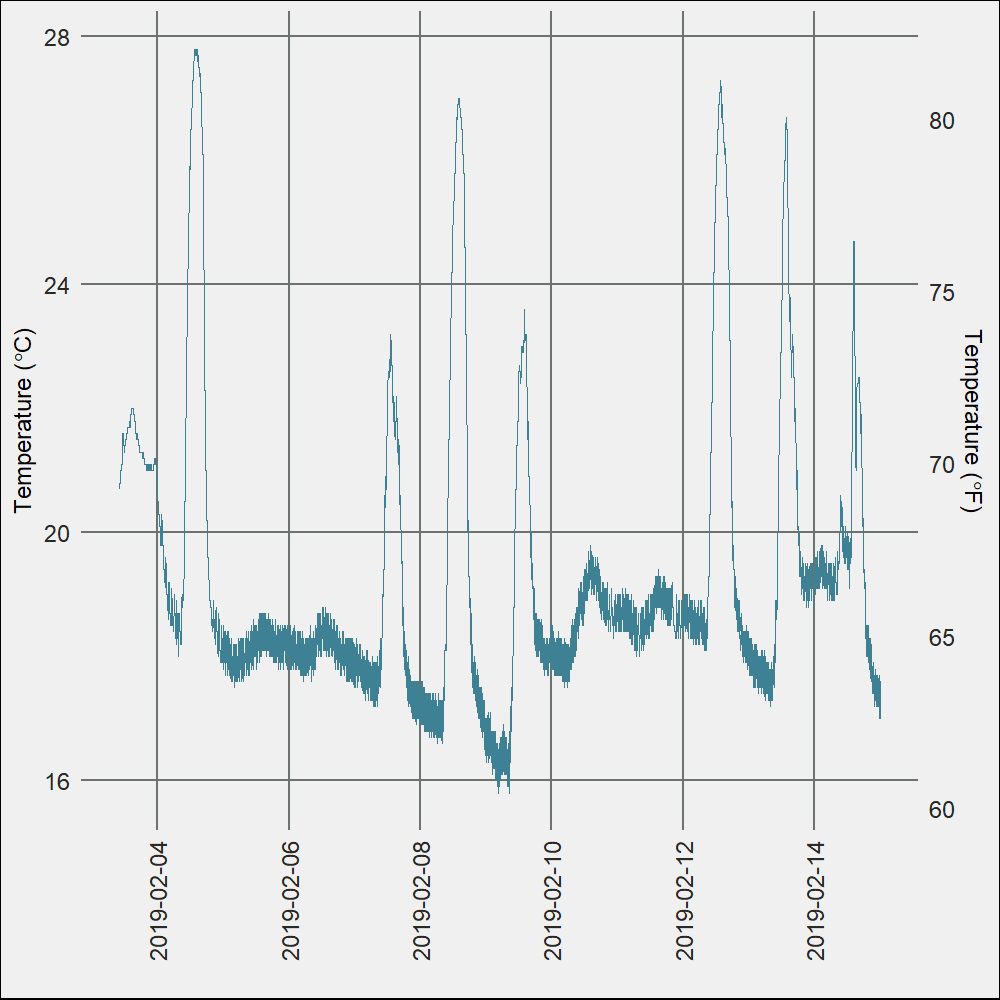
name =

expression('Temperature ('\*degree\*'F)'))) +

scale\_x\_datetime(name = '', date\_breaks = '2 days') +

theme\_tr(base\_size = 18) +

theme(axis.text.x = element\_text(angle = 90, vjust = .5))



You can see some patterns already existing within the data. Given how the  
weather patterns have been, adding in some outside temperature readings would  
provide some more insight into what is going on. I’ll dive into some more  
analysis in another post.

We can do the same for the relative humidity.

ggplot(readings) +

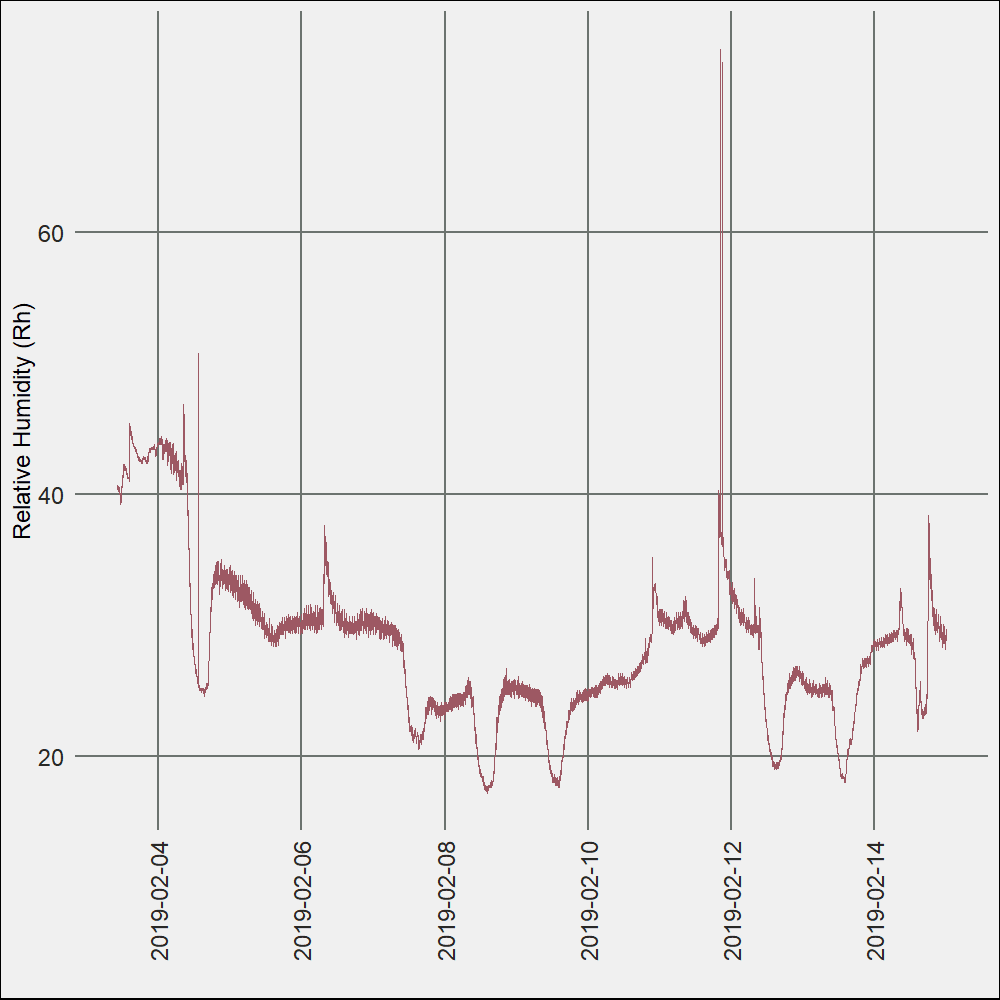
geom\_line(aes(ComputerTime, Humidity), color = '#9D5863') +

scale\_y\_continuous(name = 'Relative Humidity (Rh)') +

scale\_x\_datetime(name = '', date\_breaks = '2 days') +

theme\_tr(base\_size = 18) +

theme(axis.text.x = element\_text(angle = 90, vjust = .5))



readings[['Humidity']][readings[['Humidity']] > 50] <- NA

ggplot(readings) +

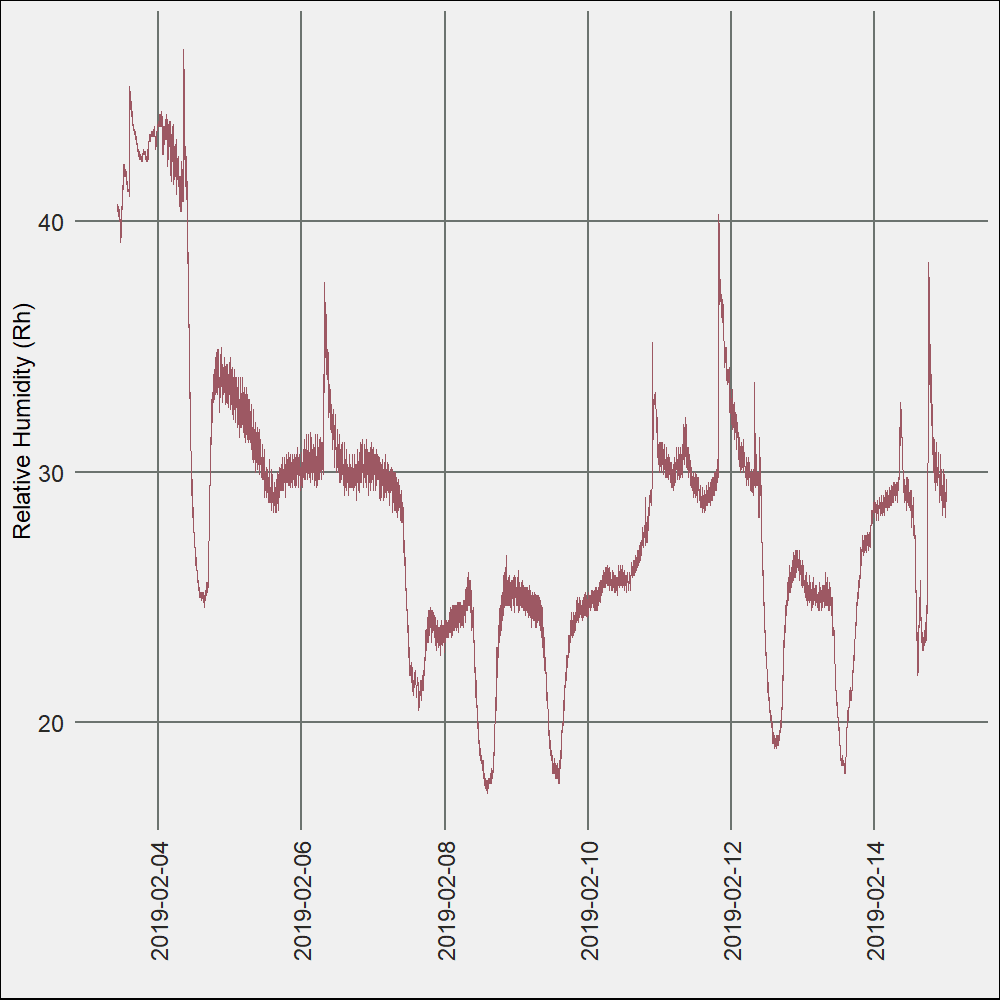
geom\_line(aes(ComputerTime, Humidity), color = '#9D5863') +

scale\_y\_continuous(name = 'Relative Humidity (Rh)') +

scale\_x\_datetime(name = '', date\_breaks = '2 days') +

theme\_tr(base\_size = 18) +

theme(axis.text.x = element\_text(angle = 90, vjust = .5))



And that’s it. You have your very own indoor climate monitoring system and  
time series data to play around with at home.