MEGAHAND

Ryan, John, Noah, Joe

11/8/2018

R packages

Tidyverse is a meta-package (a pack of packages) that is very commonly used in R, and then Tensorflow and Keras are used for Machine Learning. According to Martin, Keras was developed for Python, and then the same dev developed it for R, so it is a framework that can be used in both languages. That being said, we opted for scikit-learn.

```
# install.packages("tidyverse")
# install.packages("tensorflow", dependencies = TRUE)
# install.packages("keras", dependencies = TRUE)

library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1.
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_conflicts
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# library(tensorflow)
# library(keras)
```

R package for interoperability with Python

This package includes functions that allow you to reference Python objects in your R code, or source Python scripts from within R. I will show an example of this shortly.

```
# install.packages("reticulate", dependencies = TRUE)
library(reticulate)
```

Reticulate

With the reticulate package in R, Python code can be integrated into R documents and used alongside R. This is especially convenient in the RMarkdown document format for several reasons:

- R code and Python code can be called in discrete boxes, but within the same document
- Objects built in either environment can be passed back and forth between languages
- RMarkdown offers flexible export formats including pdf, slides, word, and html

This particular aspect of our project interested me due to the scale and diversity of challenges in interoperability, both of which I have yet to fully grasp.

Python library imports, but in an RMarkdown document

Frequently, the autocomplete available with Python functions and syntax will work within a Python chunk in an RMarkdown document, but it is not seamless yet. The words are, however, highlighted and colored as they would be when working within a .py document (despite that not being the case in this slidy presentation)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import rpytools as rpy
```

Exploratory Data Analysis and Visualization

This is a Python script that grabs all of the ".csv" files in a folder, and makes a list of the names. The script is saved as "TrainingDataGrabber.py"

From the documentation for the glob() function:

The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but *, ?, and character ranges expressed with [] will be correctly matched. This is done by using the os.scandir() and fnmatch.fnmatch() functions in concert, and not by actually invoking a subshell. Note that unlike fnmatch.fnmatch(), glob treats filenames beginning with a dot (.) as special cases. (For tilde and shell variable expansion, use os.path.expanduser() and os.path.expandvars().)

```
import os
import glob
path = 'c:\\'
extension = 'csv'
os.chdir(path= "C:/Users/joeje/Desktop/Academics/FAES/Intro_to_Python/MEGAHAND/TrainingData")
Training_Data_Files = [i for i in glob.glob('*.{}'.format(extension))]
print(Training_Data_Files)
```

Using Reticulate to source a Python Script

Here, I used R to source the Python script, create a list object containing all of the file names in the "TrainingData" folder, and then coerced an R DataFrame from that Python list for display.

```
reticulate::source_python("TrainingDataGrabber.py")

Training_Data_Files
```

```
##  [1] "Chuck Grip.csv"     "Fine Pinch.csv"     "H. Open.csv"
##  [4] "Hook Grip.csv"      "Key Grip.csv"       "No Move.csv"
##  [7] "Power Grip.csv"     "Thumb Enclosed.csv" "Tool Grip.csv"
## [10] "W. Abduction.csv"   "W. Adduction.csv"   "W. Extension.csv"
## [13] "W. Flexion.csv"     "W. Pronation.csv"   "W. Supination.csv"
```

```
knitr::kable(as.data.frame(Training_Data_Files))
```

Training_Data_Files
Chuck Grip.csv
Fine Pinch.csv
H. Open.csv
Hook Grip.csv
Key Grip.csv
No Move.csv

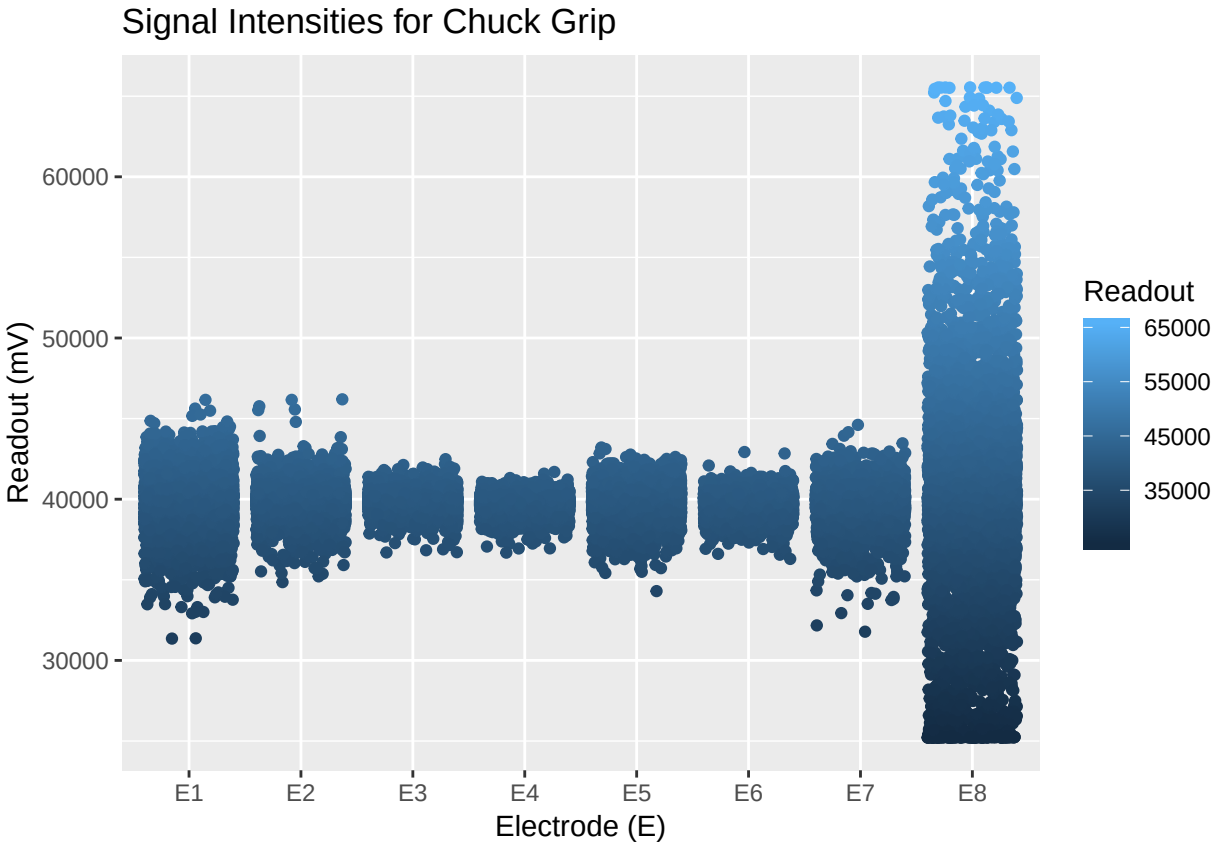
Training_Data_Files
Power Grip.csv
Thumb Enclosed.csv
Tool Grip.csv
W. Abduction.csv
W. Adduction.csv
W. Extension.csv
W. Flexion.csv
W. Pronation.csv
W. Supination.csv

Using R for data tidying and visualization

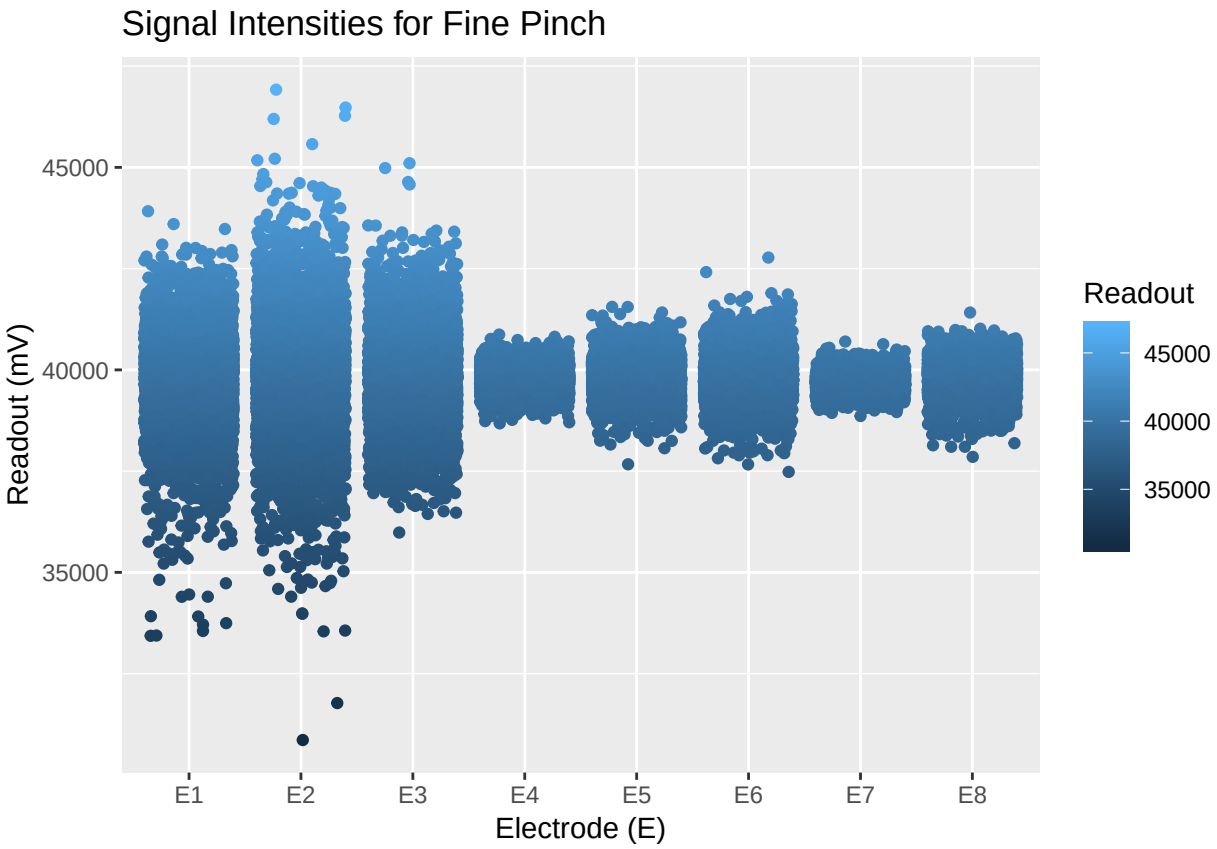
Next, I used the purrr package from R to apply a function I made in R that tidys the data (removing extraneous columns and formatting) and then creates a pre-set visualization for all of the files from the list (that was made in Python.)

```
source("C:/Users/joeje/Desktop/Academics/FAES/Intro_to_Python/MEGAHAND/Megamunge_Jitter.R")
library(purrr)
setwd('TrainingData')
map(Training_Data_Files, Megamunge)
```

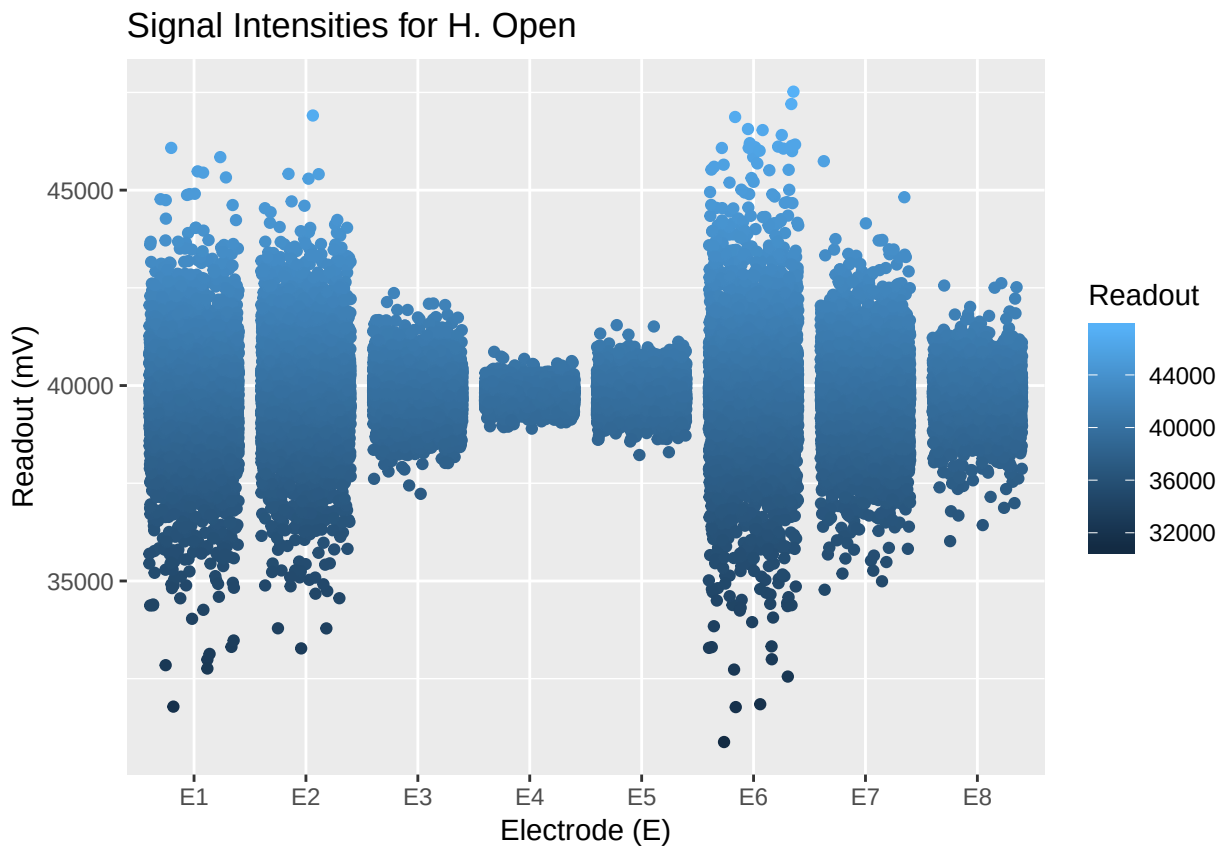
```
## [[1]]
```



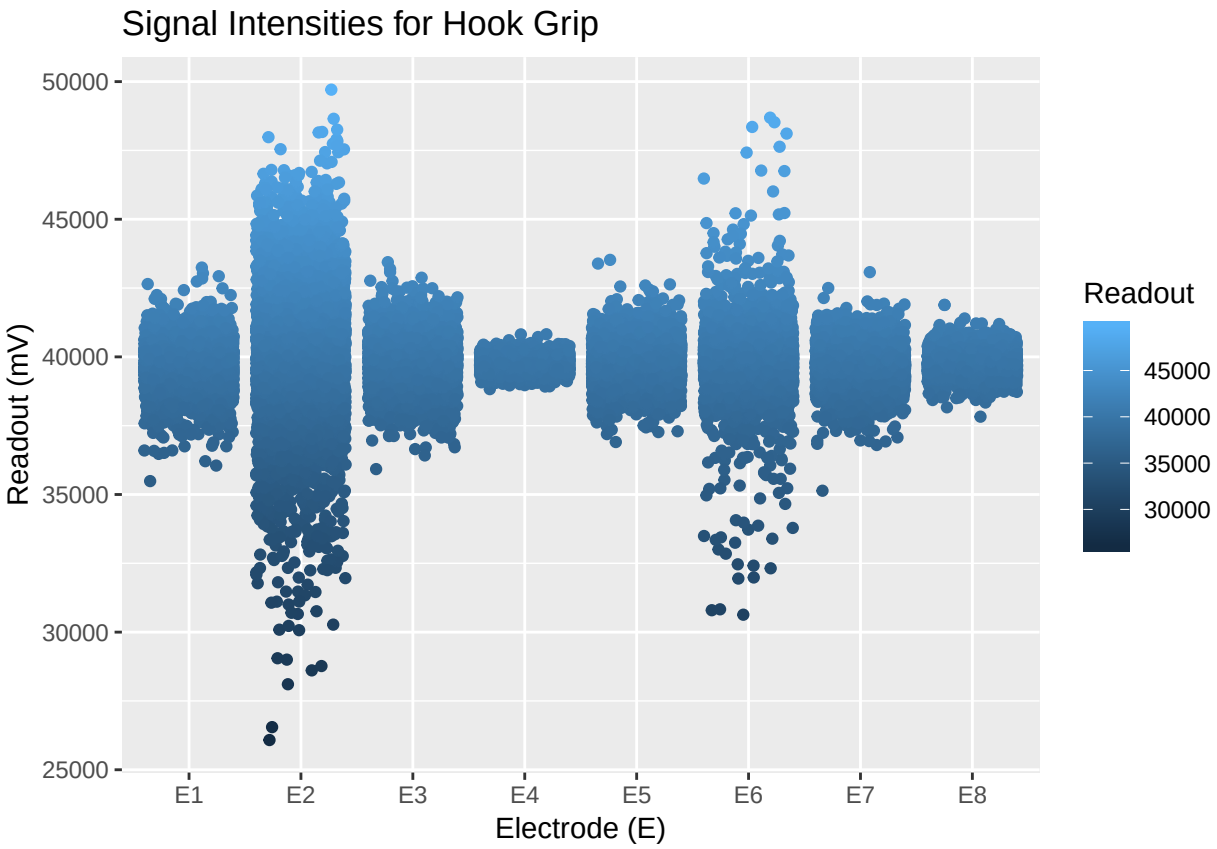

```
##
## [[2]]
```


Signal Intensities for Fine Pinch
Readout (mV)
45000
40000
35000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000

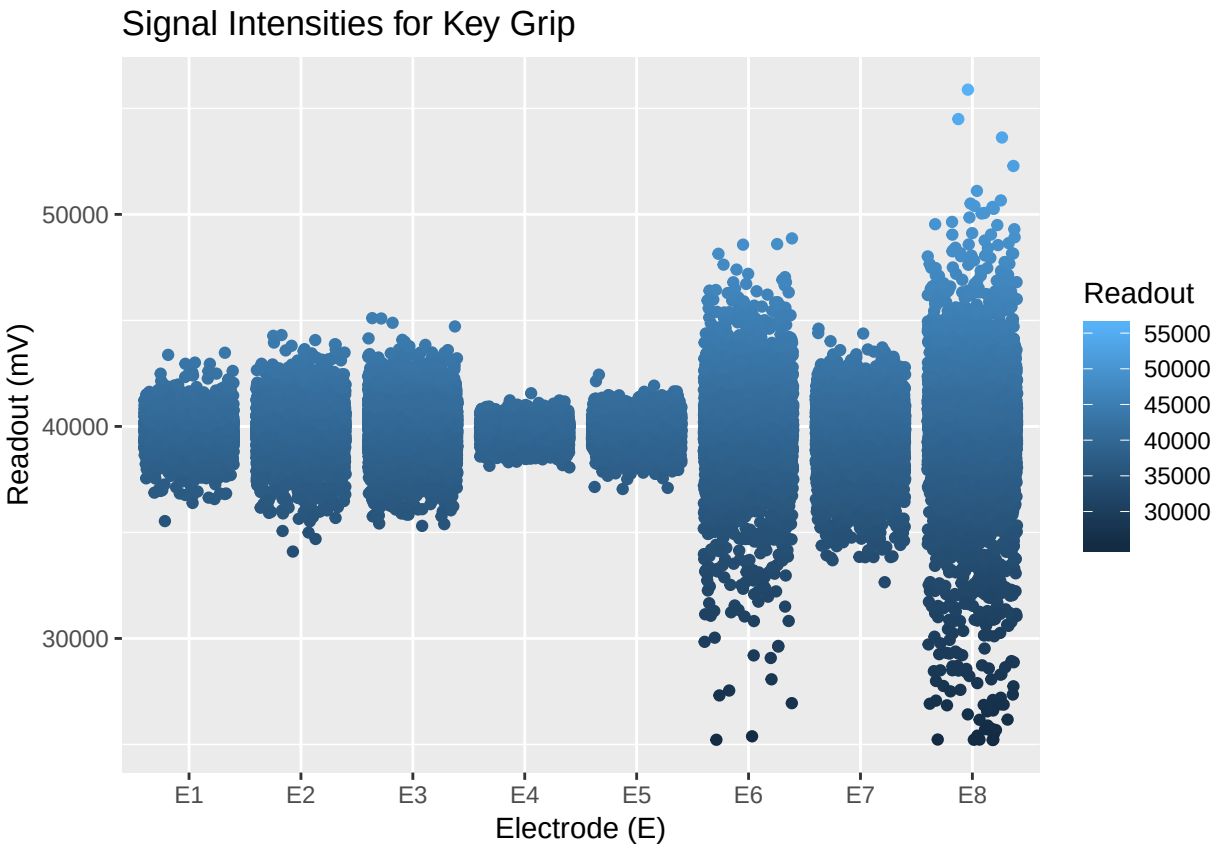
```
##
## [[3]]
```


Signal Intensities for H. Open
Readout (mV)
45000
40000
35000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
44000
40000
36000
32000

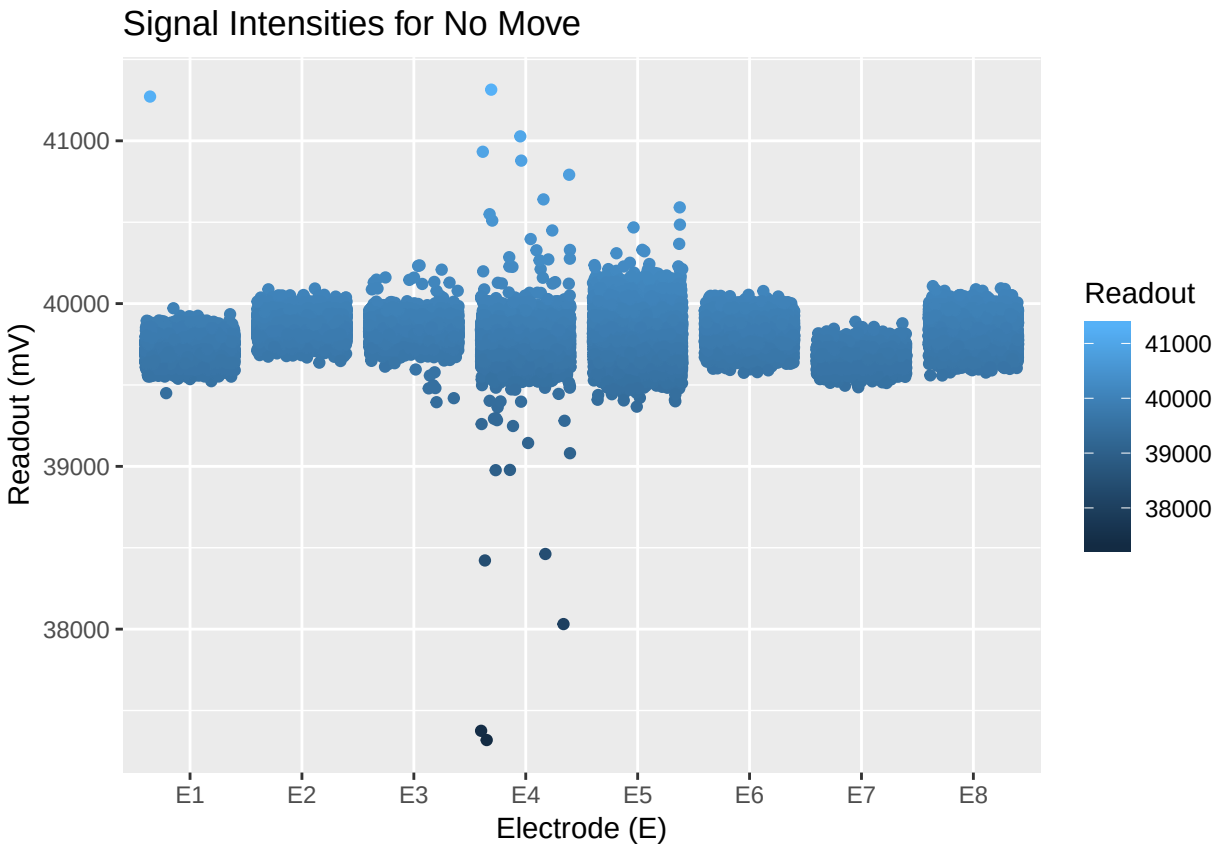
```
##
## [[4]]
```


Signal Intensities for Hook Grip
Readout (mV)
50000
45000
40000
35000
30000
25000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000
30000

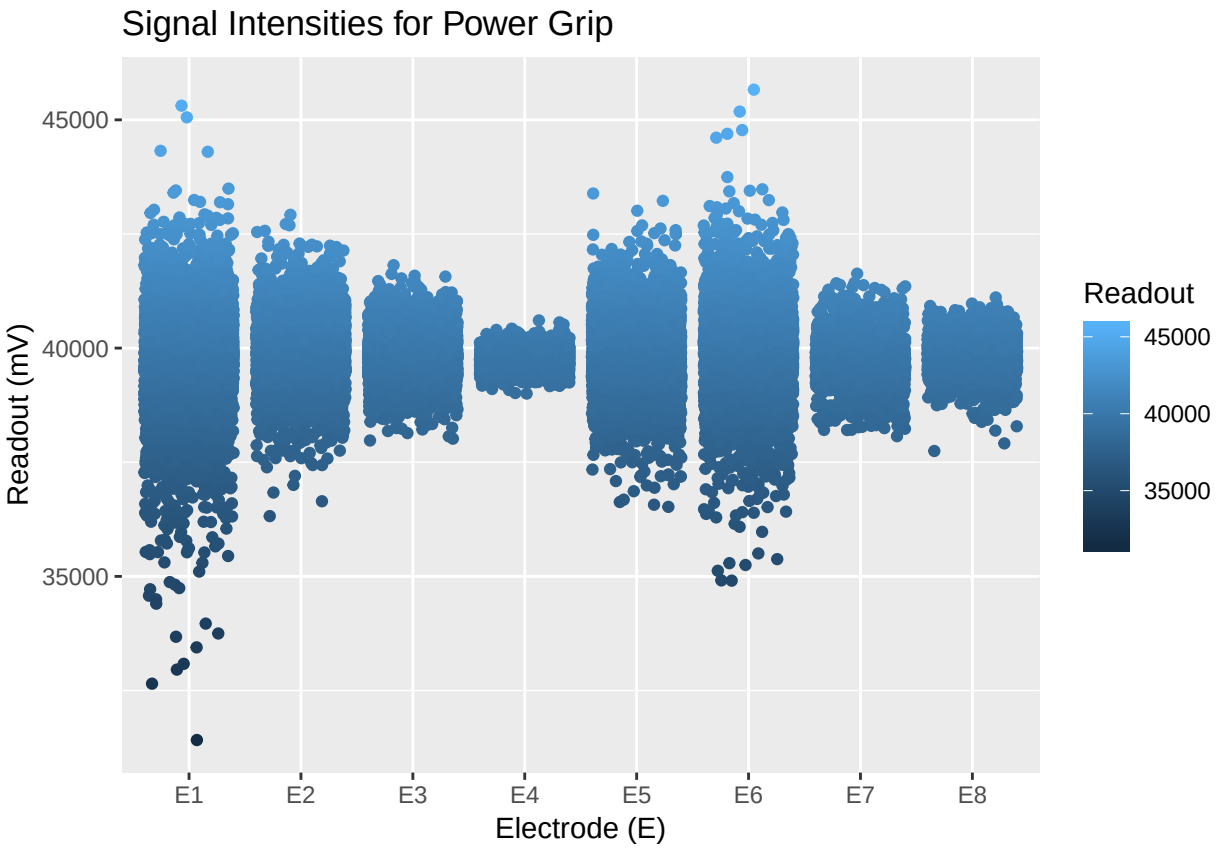
```
##
## [[5]]
```


Signal Intensities for Key Grip
Readout (mV)
50000
40000
30000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
55000
50000
45000
40000
35000
30000

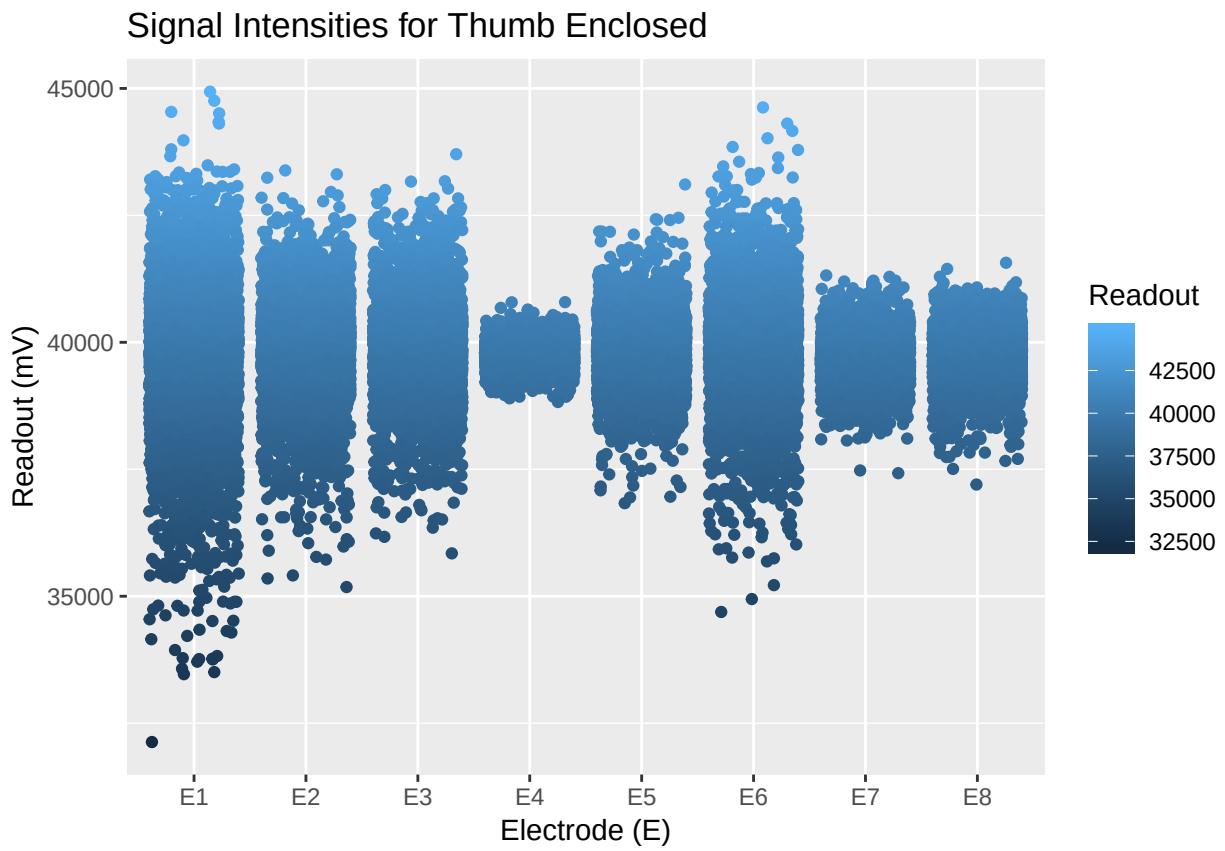
```
##
## [[6]]
```


Signal Intensities for No Move
Readout (mV)
41000
40000
39000
38000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
41000
40000
39000
38000

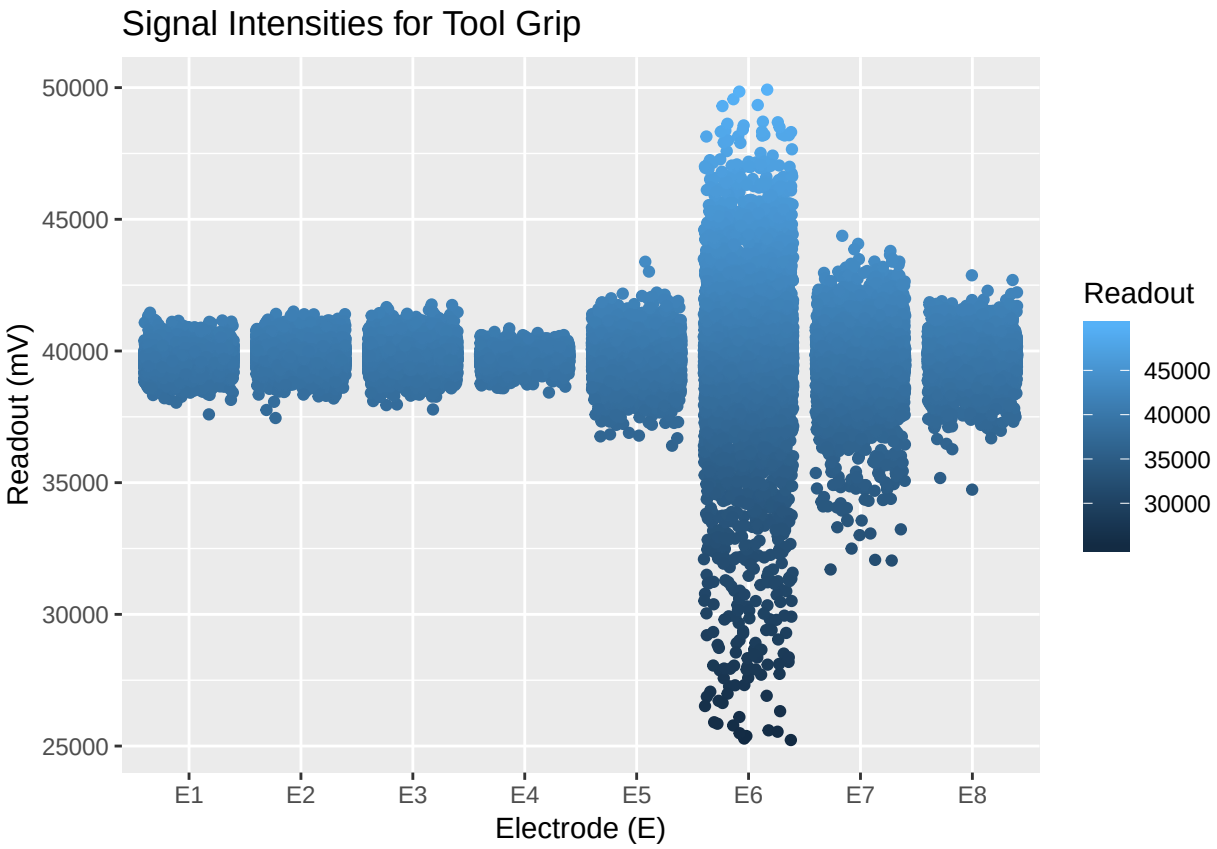
```
##
## [[7]]
```


Signal Intensities for Power Grip
45000
40000
35000
Readout (mV)
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000

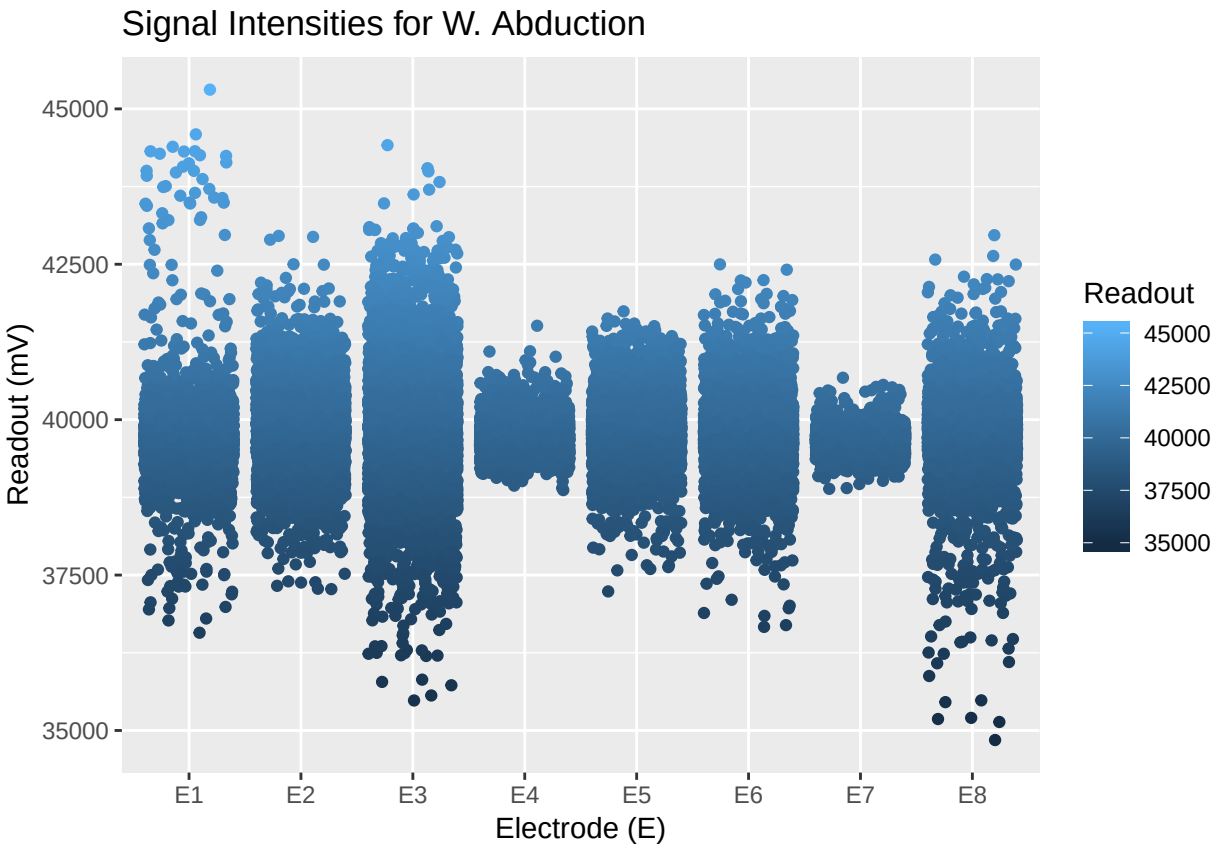
```
##
## [[8]]
```


Signal Intensities for Thumb Enclosed
45000
40000
35000
Readout (mV)
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
42500
40000
37500
35000
32500

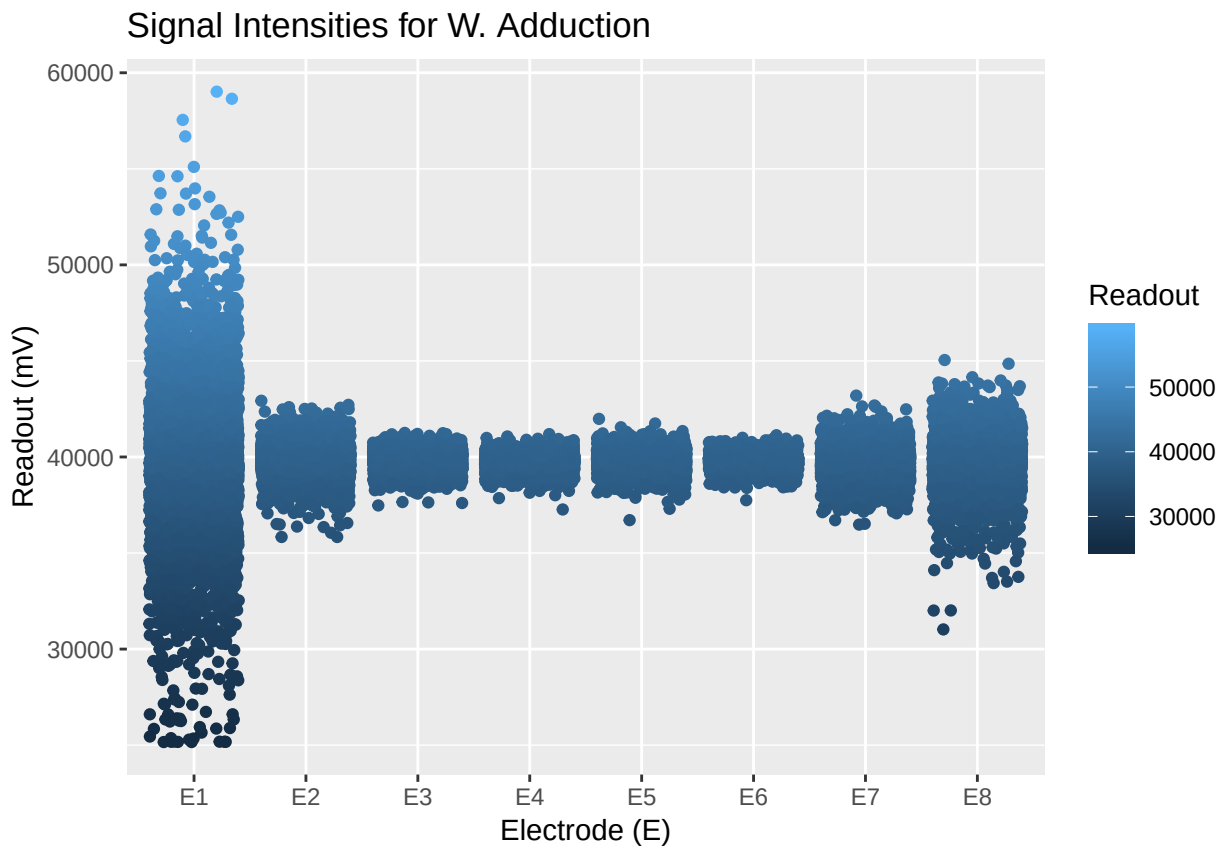
```
##
## [[9]]
```


Signal Intensities for Tool Grip
50000
45000
40000
35000
30000
25000
Readout (mV)
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000
30000

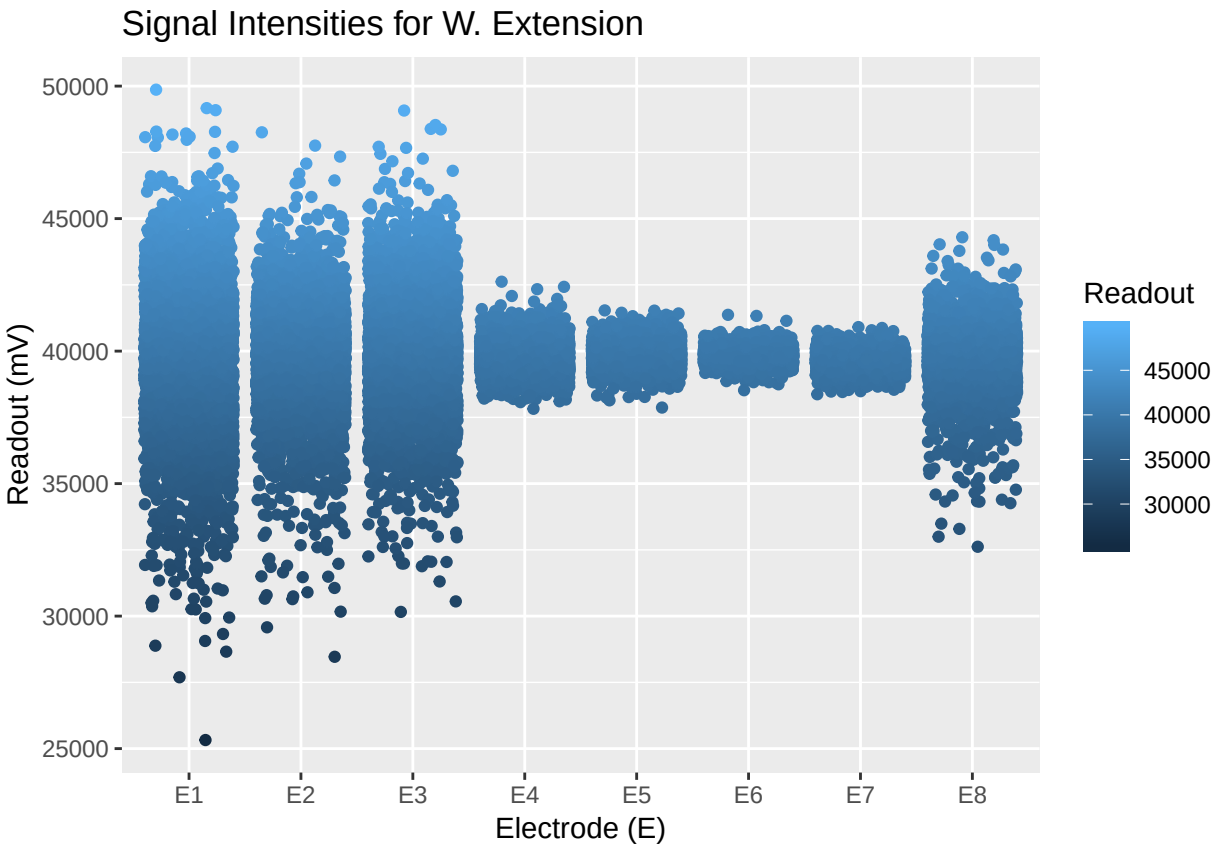
```
##
## [[10]]
```


Signal Intensities for W. Abduction
Readout (mV)
45000
42500
40000
37500
35000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
42500
40000
37500
35000

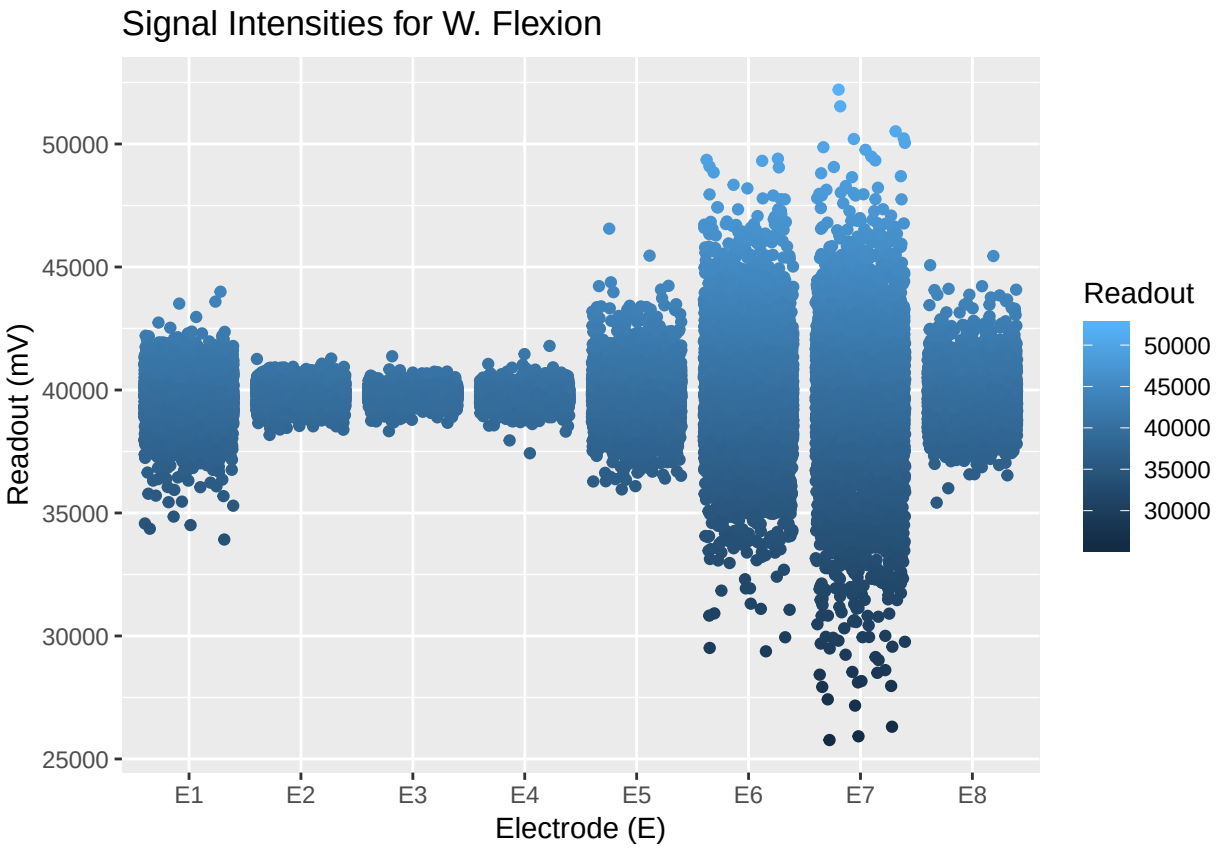
```
##
## [[11]]
```


Signal Intensities for W. Adduction
60000
50000
40000
30000
Readout (mV)
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
50000
40000
30000

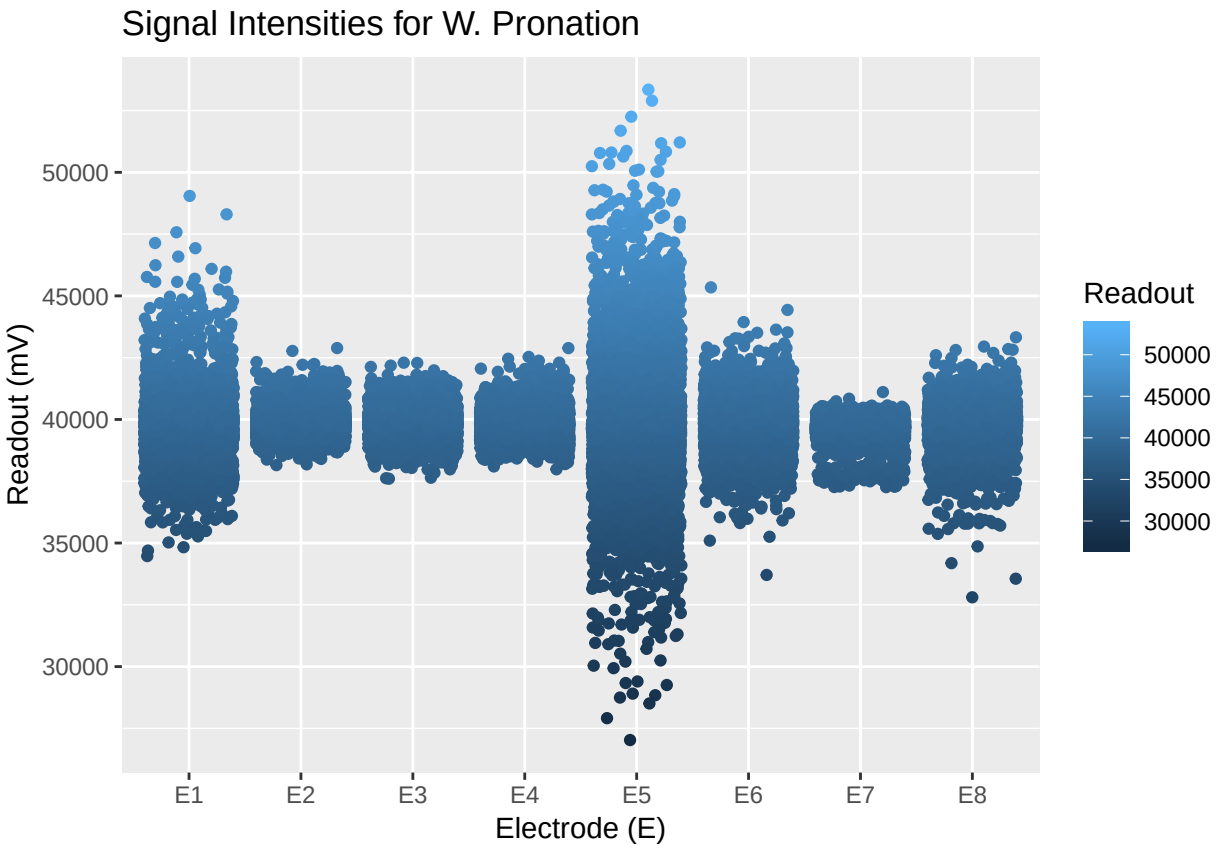
```
##
## [[12]]
```


Signal Intensities for W. Extension
Readout (mV)
50000
45000
40000
35000
30000
25000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000
30000

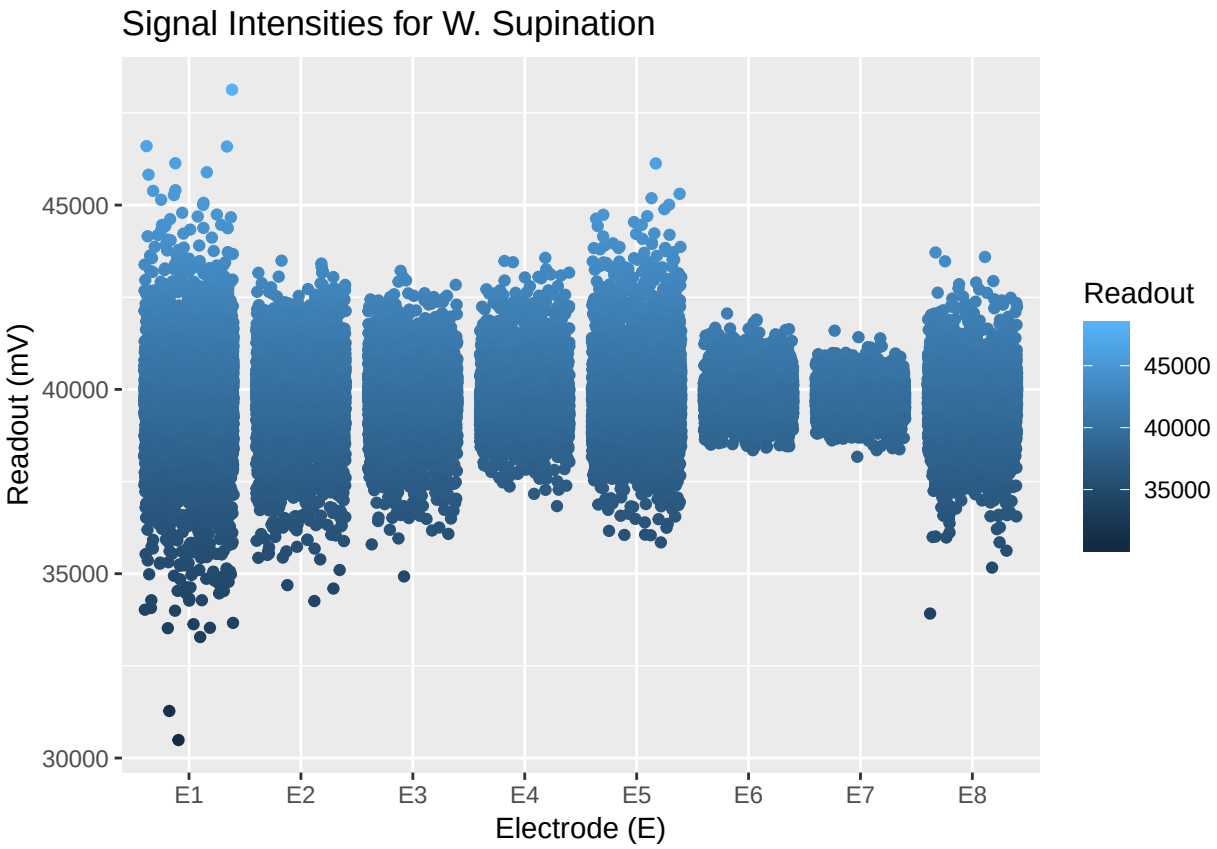
```
##
## [[13]]
```


Signal Intensities for W. Flexion
Readout (mV)
50000
45000
40000
35000
30000
25000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
50000
45000
40000
35000
30000

```
##
## [[14]]
```


Signal Intensities for W. Pronation
Readout (mV)
50000
45000
40000
35000
30000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
50000
45000
40000
35000
30000

```
##
## [[15]]
```


Signal Intensities for W. Supination
Readout (mV)
45000
40000
35000
30000
E1
E2
E3
E4
E5
E6
E7
E8
Electrode (E)
Readout
45000
40000
35000