NLP Tutorial

Last time: Chinese sermon voice recognition plan

Simple validation ⇒ simple model

Extended validation ⇒ improved model

Automated framework ⇒ insightful report

Key questions

- What's the right tool to use?
- How large data can be handled? Is there transformation needed?
- What's the easiest way to formulate the problem? What's the easiest model to use?
- What's the easiest validation approach to use?

Key questions

- Can we define some insight driven validation metrics?
- Can we leverage these validation metrics to improve the model performance?
- Is it needed to leverage CNN or other deep learning approaches?

Key questions

- What is the final outcome needed?
- How to understand the model accuracy vs outcome?
- What are the insights we can provide to the speaker?

Auto translate 10 min Chinese sermon with insights

Simple validation ⇒ simple model

- What's the right tool to use?
- How large data can be handled? Is there transformation needed?
- What's the easiest way to formulate the problem? What's the easiest model to use?
- What's the easiest validation approach to use?

Extended validation ⇒ improved model

- Can we define some insight driven validation metrics?
- Can we leverage these validation metrics to improve the model performance?
- Is it needed to leverage CNN or other deep learning approaches?

Automated framework ⇒ insightful report

- What is the final outcome needed?
- How to understand the model accuracy vs outcome?
- What are the insights we can provide to the speaker?

Step 1: Build a simple model with simple validation in place for a sample audio

- What's the right tool to use?
  - Is R a good language for this? Which package for R?
  - In which environment?
- How large data can be handled? Is there transformation needed?
  - Don't know the answer.
  - How many data points are in 1 min? How many do we need?
  - What does FFT do? Why should I care about it?
- What's the easiest way to formulate the problem? What's the easiest model to use?
  - To formulate it as a clustering problem? a classification problem?
- What's the easiest validation approach to use?
  - Validate all seems to be a huge effort. How to establish a very quick validation approach?

What's the right tool to use

http://samcarcagno.altervista.org/blog/basic-sound-processing-r/?doing_wp_cron=1558214905.3249189853668212890625

Basic Sound Processing with R

10 December 2013 by samcarcagno

This page describes some basic sound processing functions in R. We'll use the tuneR package to read in wav files. No other packages will be needed for this tutorial, however, there is a number of packages that are in general very useful for working with sound in R:

- signal – MATLAB like signal processing functions
- seewave – functions for analysing, manipulating, displaying, editing and synthesizing time waves (particularly sound).

We can install the sound package with

```
install.packages('tuneR', dep=TRUE)
```

Assuming that the tuneR package has already been installed correctly we can load it with the library() function

```
library(tuneR)
```

Now we can read in a wav file, you can download it here 440_sine.wav, it contains a complex tone with a 440 Hz fundamental frequency (F0) plus noise.

```
sndObj <- readWave('440_sine.wav')
```

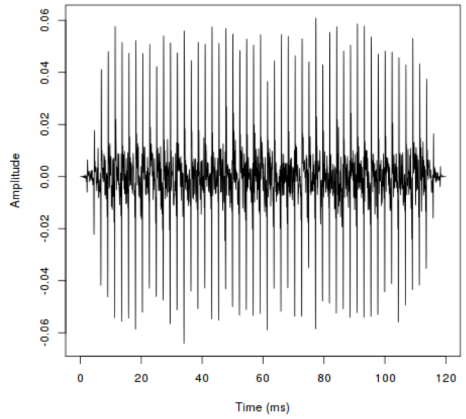
Plotting the Tone

A time representation of the sound can be obtained by plotting the pressure values against the time axis. However we need to create an array containing the time points first:

```
timeArray <- (0:(5292-1)) / sndObj@samp.rate
timeArray <- timeArray * 1000 #scale to milliseconds
```

now we can plot the tone:

```
plot(timeArray, s1, type='l', col='black', xlab='Time (ms)', ylab='Amplitude')
```



Plotting the Frequency Content

Another useful graphical representation is that of the frequency content, or spectrum of the tone. We can obtain the frequency spectrum of the sound using the fft function, that implements a Fast Fourier Transform algorithm. We'll follow closely the technical document available here to obtain the power spectrum of our sound.

```
n <- length(s1)
p <- fft(s1)
```

the fft is computed on the number of points of the signal n. Since we're not using a power of two the computation will be a bit slower, but for signals of this duration this is negligible.

```
nUniquePts <- ceiling((n+1)/2)
p <- p[1:nUniquePts] #select just the first half since the second half
                     # is a mirror image of the first
p <- abs(p)  #take the absolute value, or the magnitude
```

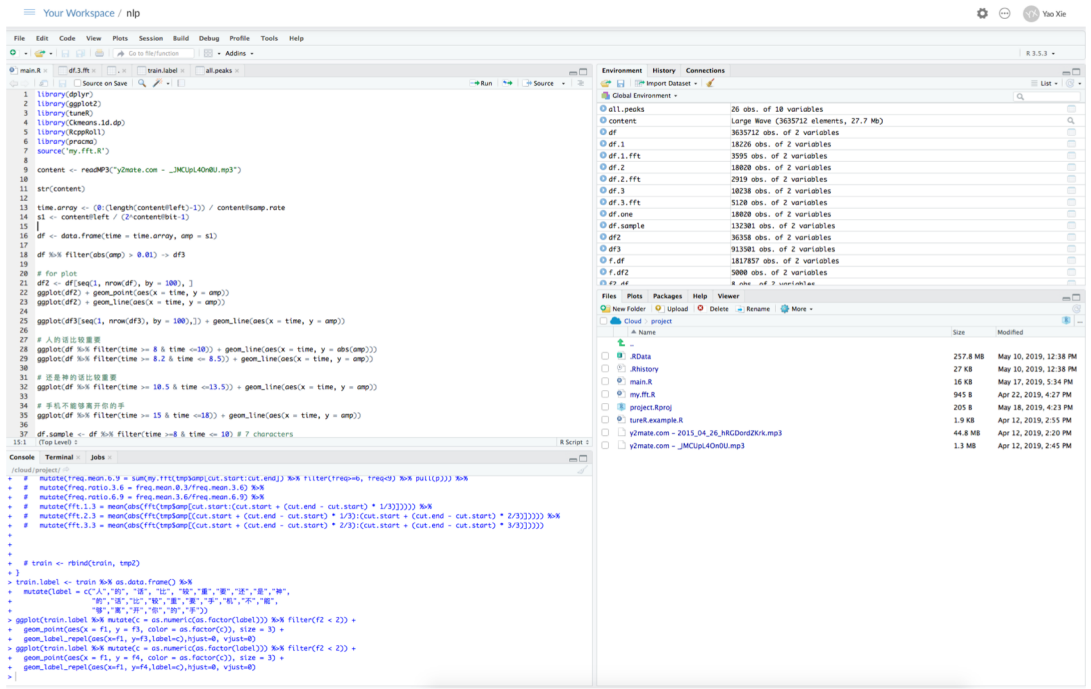
the fourier transform of the tone returned by the fft function contains both magnitude and phase information and is given in a complex representation (i.e. returns complex numbers). By taking the absolute value of the fourier transform we get the information about the magnitude of the frequency components.

```
p <- p / n #scale by the number of points so that
           # the magnitude does not depend on the length
           # of the signal or on its sampling frequency
p <- p^2  # square it to get the power

# multiply by two (see technical document for details)
# odd nfft excludes Nyquist point
if (n %% 2 > 0){
  p[2:length(p)] <- p[2:length(p)]*2 # we've got odd number of points fft
} else {
  p[2: (length(p) -1)] <- p[2: (length(p) -1)]*2 # we've got even number of points fft
}

freqArray <- (0:(nUniquePts-1)) * (sndObj@samp.rate / n) #  create the frequency array

plot(freqArray/1000, 10*log10(p), type='l', col='black', xlab='Frequency (kHz)',
ylab='Power (dB)')
```

The resulting plot can bee seen below, notice that we're plotting the power in decibels by taking 10*log10(p), we're also scaling the frequency array to kilohertz by dividing it by 1000

What's the right environment to use?

- Need to have the flexibility to work on multiple machines
- Need to be able to install tuneR, dplyr, ggplot2 packages



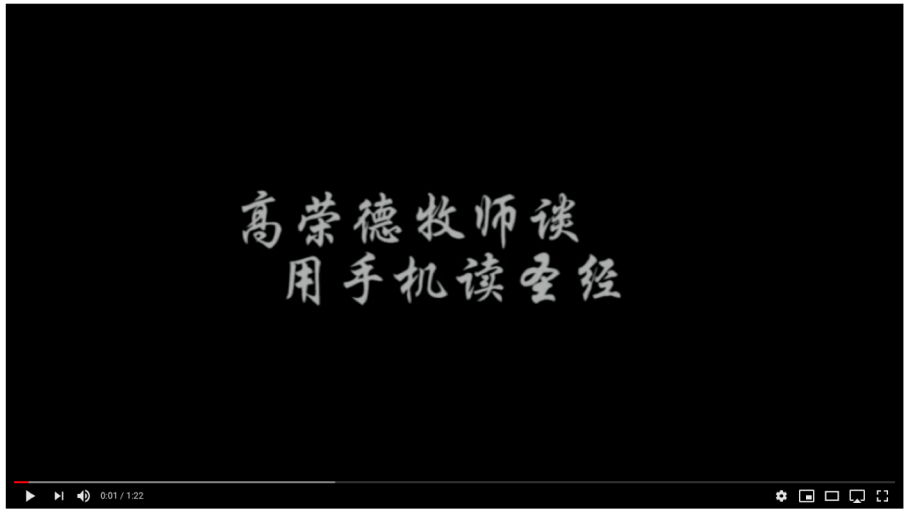
How large data can be handled?

事奉之道 2015-04-26

87 views

40 mins,
44.8 MB

RStudio
Freezed!!!



高荣德牧师谈用手机读圣经

1min
22seconds,
1.3MB

What decision do we need to make after loading the data?

Simplify the problem by considering only ‘left’

content	S4 (tuneR::Wave)	S4 object of class Wave
left	integer [3635712]	0 0 0 0 0 0 ...
right	integer [3635712]	0 0 0 0 0 0 ...
stereo	logical [1]	TRUE
samp.rate	double [1]	44100
bit	double [1]	16
pcm	logical [1]	TRUE

“CD Quality” audio is sampled at 44.1khz (44,100 readings per second). But for speech recognition, a sampling rate of 16khz (16,000 samples per second) is enough to cover the frequency range of human speech.
But thanks to the Nyquist theorem, we know that we can use math to perfectly reconstruct the original sound wave from the spaced-out samples — as long as we sample at least twice as fast as the highest frequency we want to record.

3 million data points for a 1 min 22 seconds audio ➔ probably sampling is needed, but how? Sampling from 44,100 to 16,000 will still have like 1 million data points, which seem to be still too many...

Step 1 status check

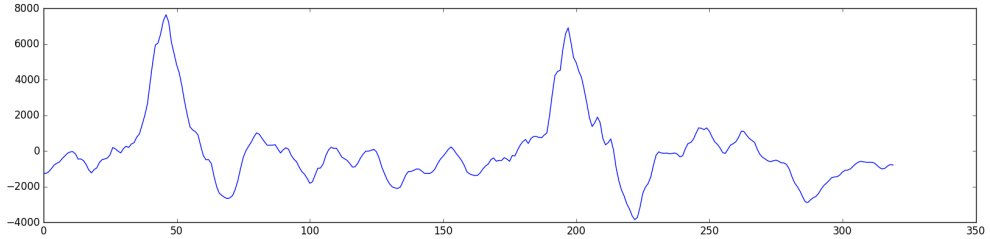
- What's the right tool to use?
  - Is R a good language for this? Which package for R?
  - In which environment?
- How large data can be handled? Is there transformation needed?
  - Don't know the answer.
  - How many data points are in 1 min? How many do we need?
  - What does FFT do? Why should I care about it?
- What's the easiest way to formulate the problem? What's the easiest model to use?
  - To formulate it as a clustering problem? a classification problem?
- What's the easiest validation approach to use?
  - Validate all seems to be a huge effort. How to establish a very quick validation approach?

How to report at this stage?

- Identified the right tool to use
- Identified the right size of data to use
- Successfully loaded data and made necessary simplification assumptions
- Next step
  - Understand what FFT is and how to use it
  - Identify the approach to decompose the words
  - Identify the approach to cluster the words and define validation approach

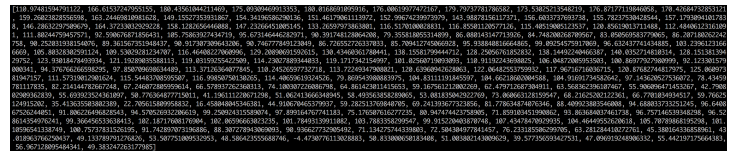
The most popular voice recognition algorithm

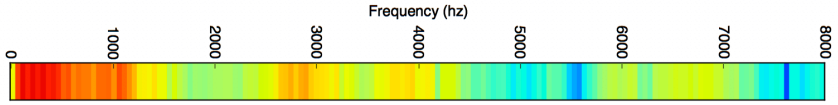
- https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a

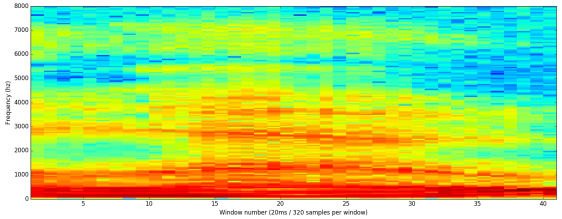


Fourier Transform

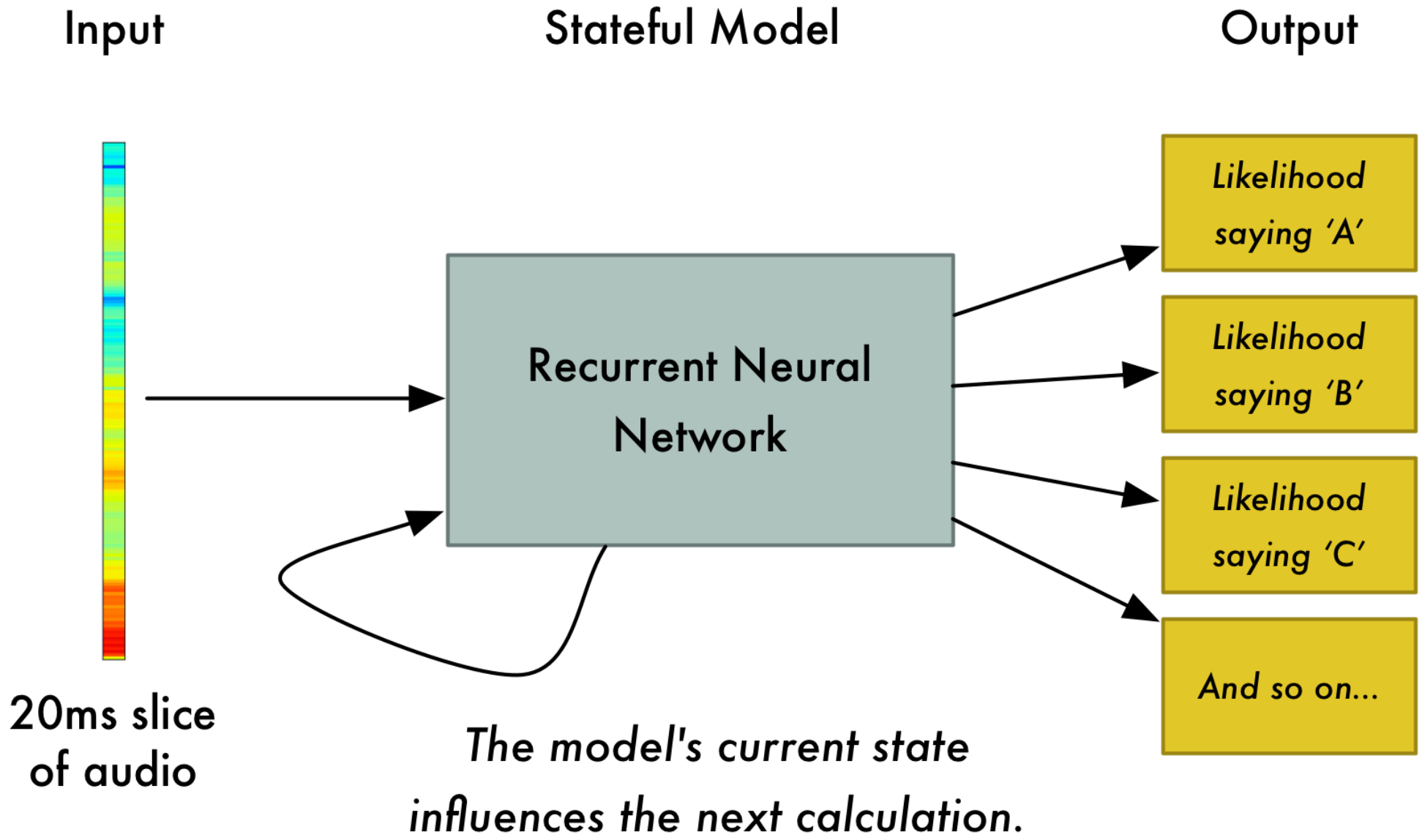
It breaks apart the complex sound wave into the simple sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one.







The most popular voice recognition algorithm



Which algorithm to use for this project?

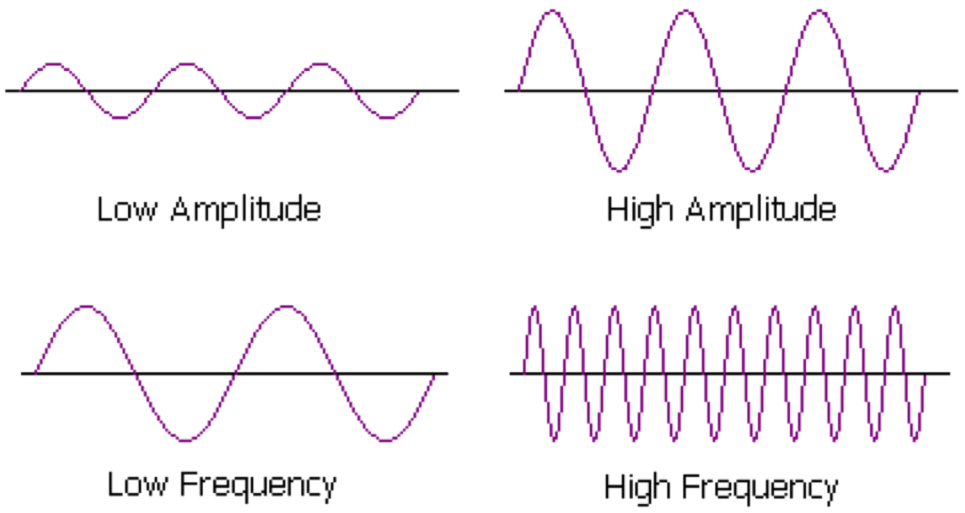
	Deep Learning	Split words ➔ cluster ➔ machine learning /deep learning
Data size	Need a lot of data	Can start from 1 min data
Running time	Need very long time to run	Can be very fast
Results as project progresses	Very hard to get	Yes
How long does it take to get some results?	Data collection, data labeling, model tuning (>3 months)	Results are generated as the project progresses
Recommendation		👍👍👍

What is FFT? Why should I care?

Wikipedia

A **fast Fourier transform** (**FFT**) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

Amplitude vs. high amplitude



- **Amplitude is unrelated to words**
- **Only frequency is related to actual words**
- **Only frequency related features can be useful in prediction**

Propose a modeling workflow given learnings so far

Audio data

Algorithm to decompose words (based on amplitude)

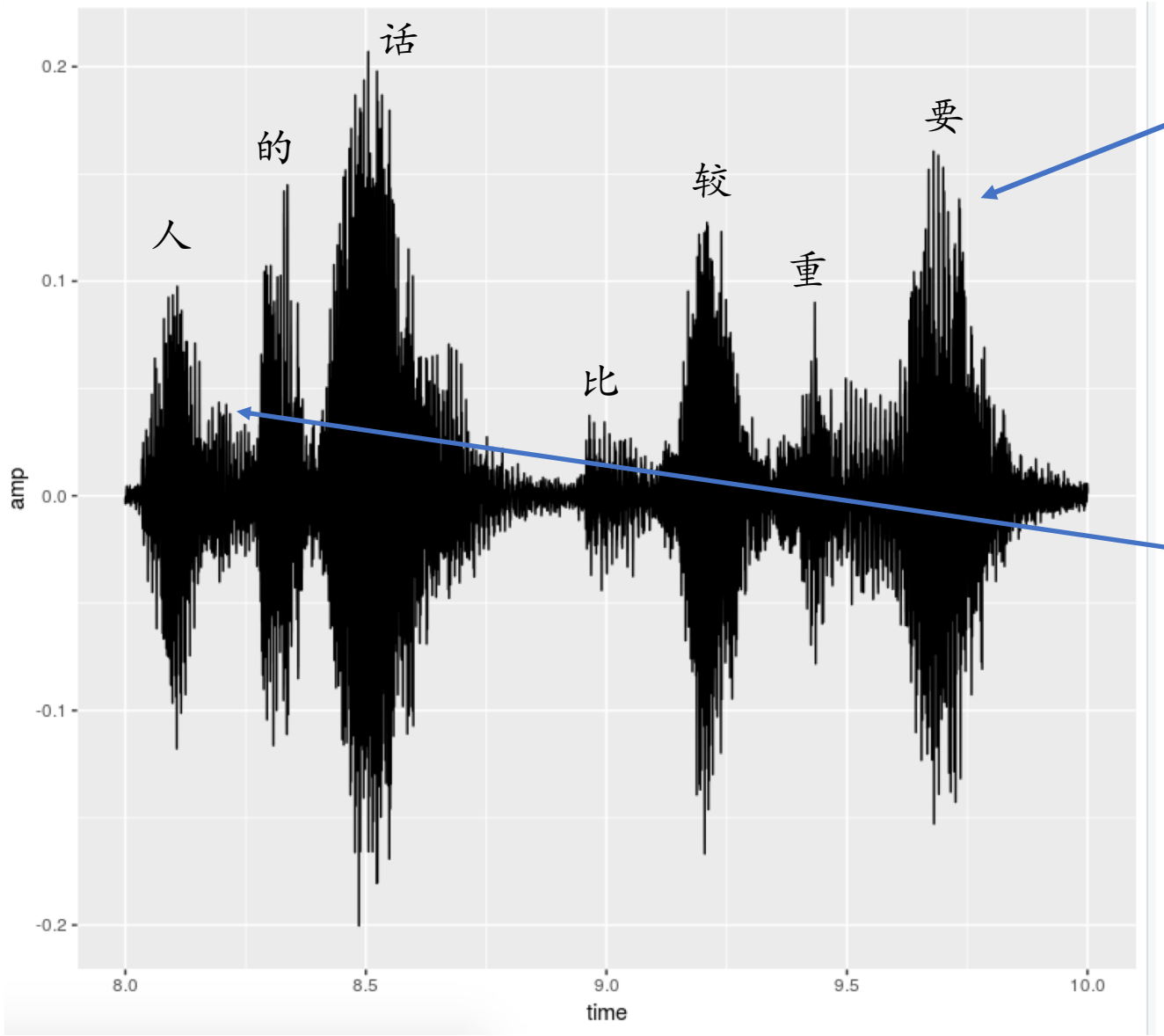
FFT on each word

Derive features from FFT

Clustering algorithm (based on frequency)

Supervised learning (TBD)

Plot of the first sentence form the audio suggest key levers for the decomposition algorithm



Each character is a peak in terms of amplitude

Need to ignore some minor peaks

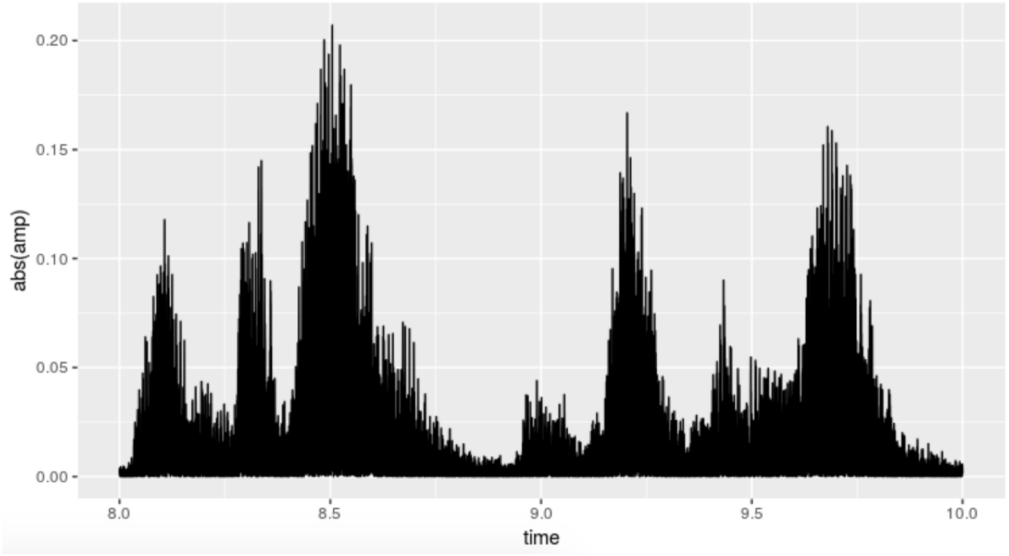
Define a ‘pace’ parameter to characterize talking speed

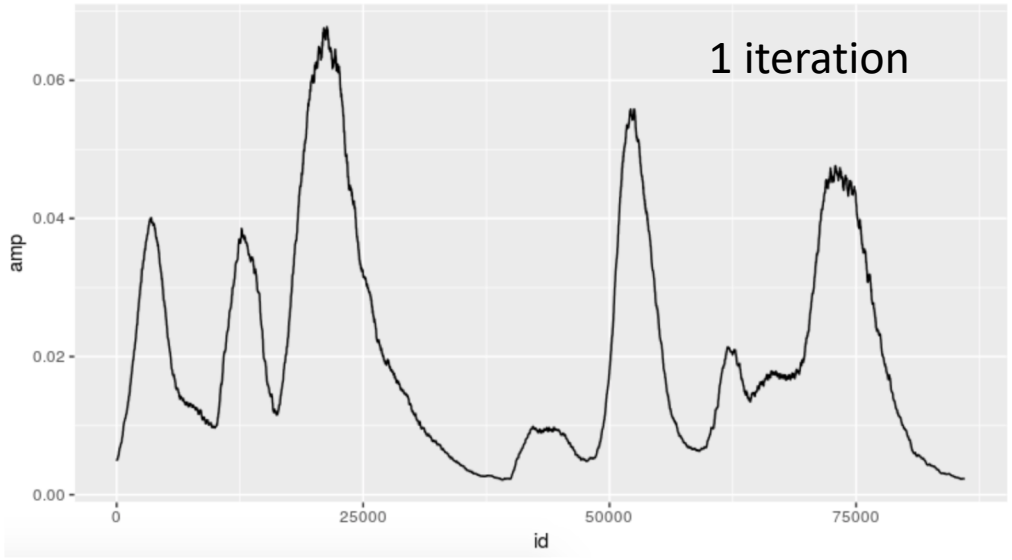
```
tmp <- data.frame(amp=roll_mean(abs(df.sample$amp), 44100*pace)) %>% mutate(id = 1:n())
```

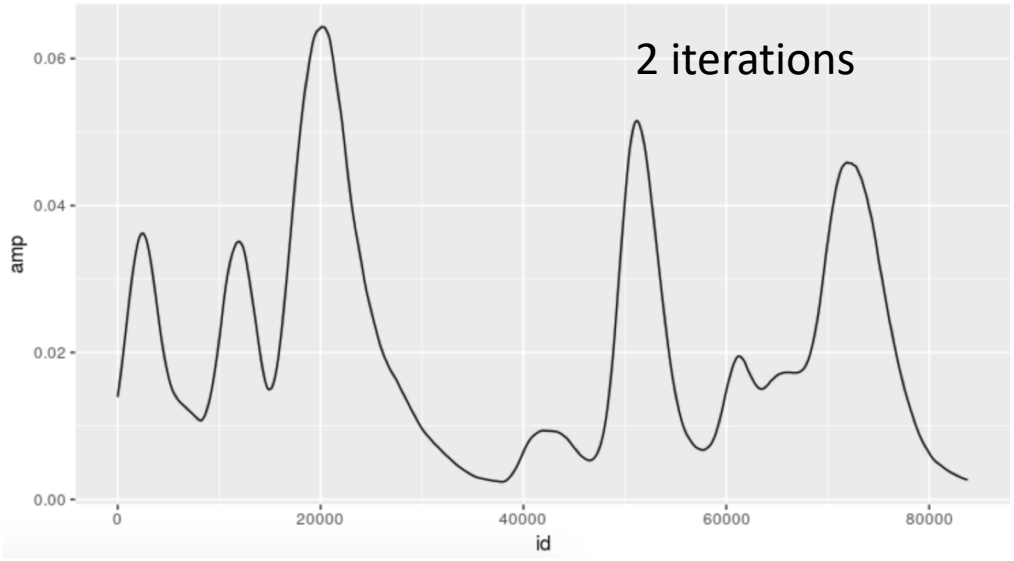
Rolling window size

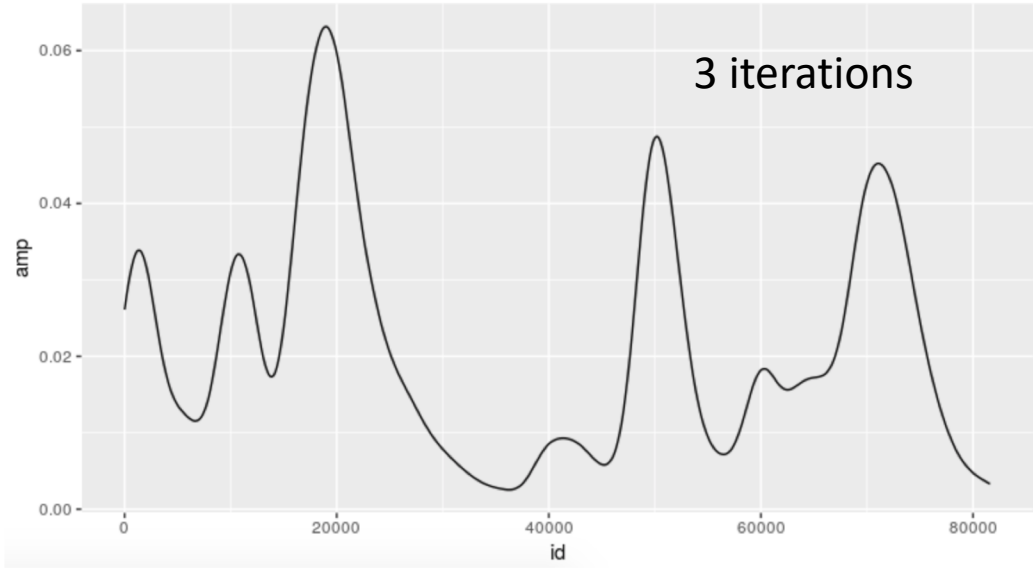
- Rolling window size is dependent on the speed of the voice
- Decisions
  - Step 1: define a parameter pace to characterize the speed and hence the smoothing
  - Step 2: learn the pace parameter from each audio data (future)

Leverage roll_mean to smooth the signal

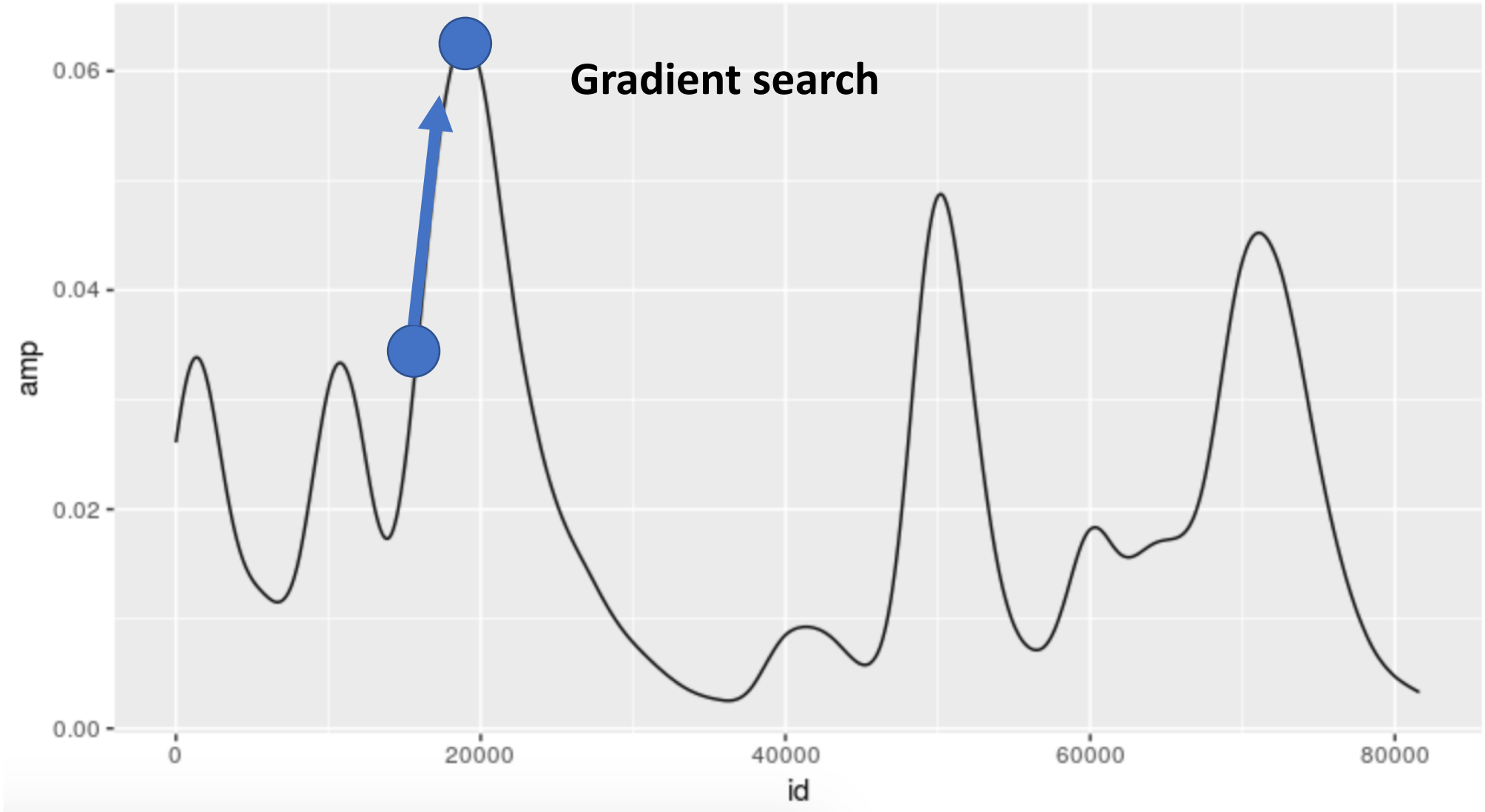








How to identify the peaks?



Gradient search is a tiny step in the big picture ➔ probably I don't want to spend time implementing my own algorithm

Audio data

Algorithm to decompose words (based on amplitude)

- Gradient search

FFT on each word

Derive features from FFT

Clustering algorithm (based on frequency)

Supervised learning (TBD)

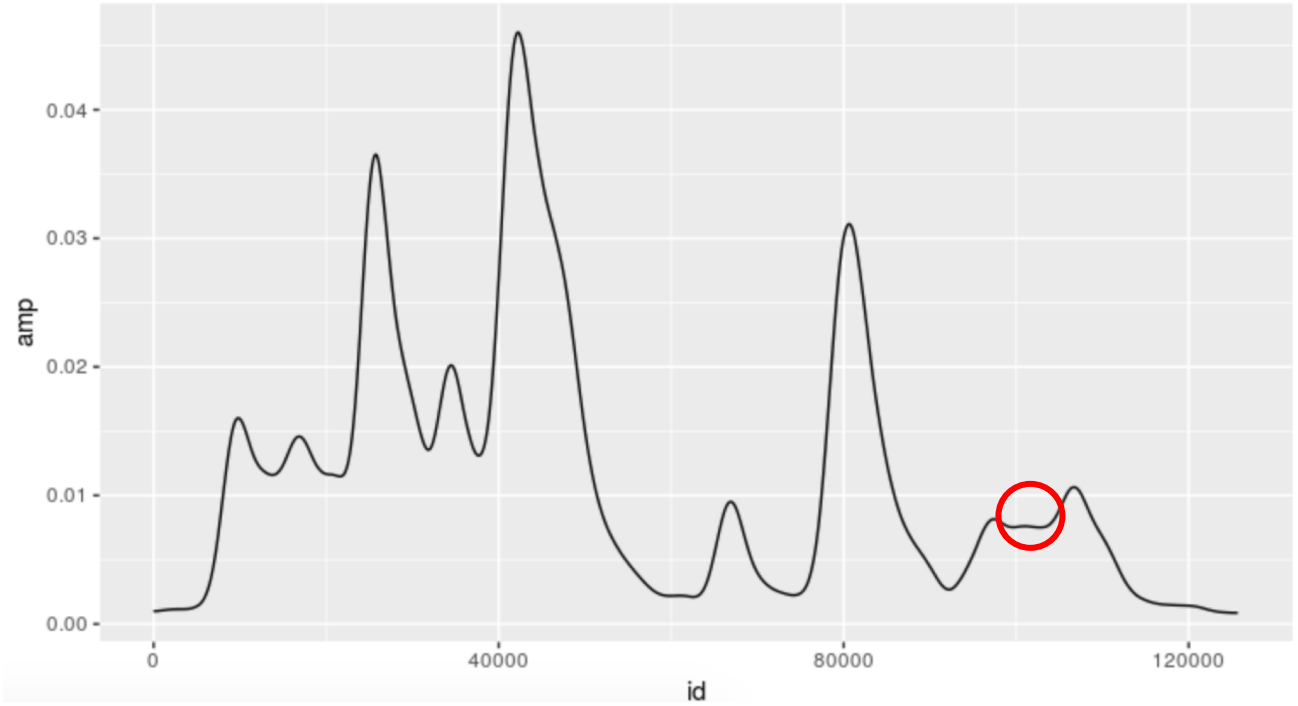
Leverage ‘findpeaks’ function with another parameter

```
peaks <- findpeaks(tmp3$amp, floor(44100*pace*peak.pace.ratio), minpeakheight = min.peak.amp)
```

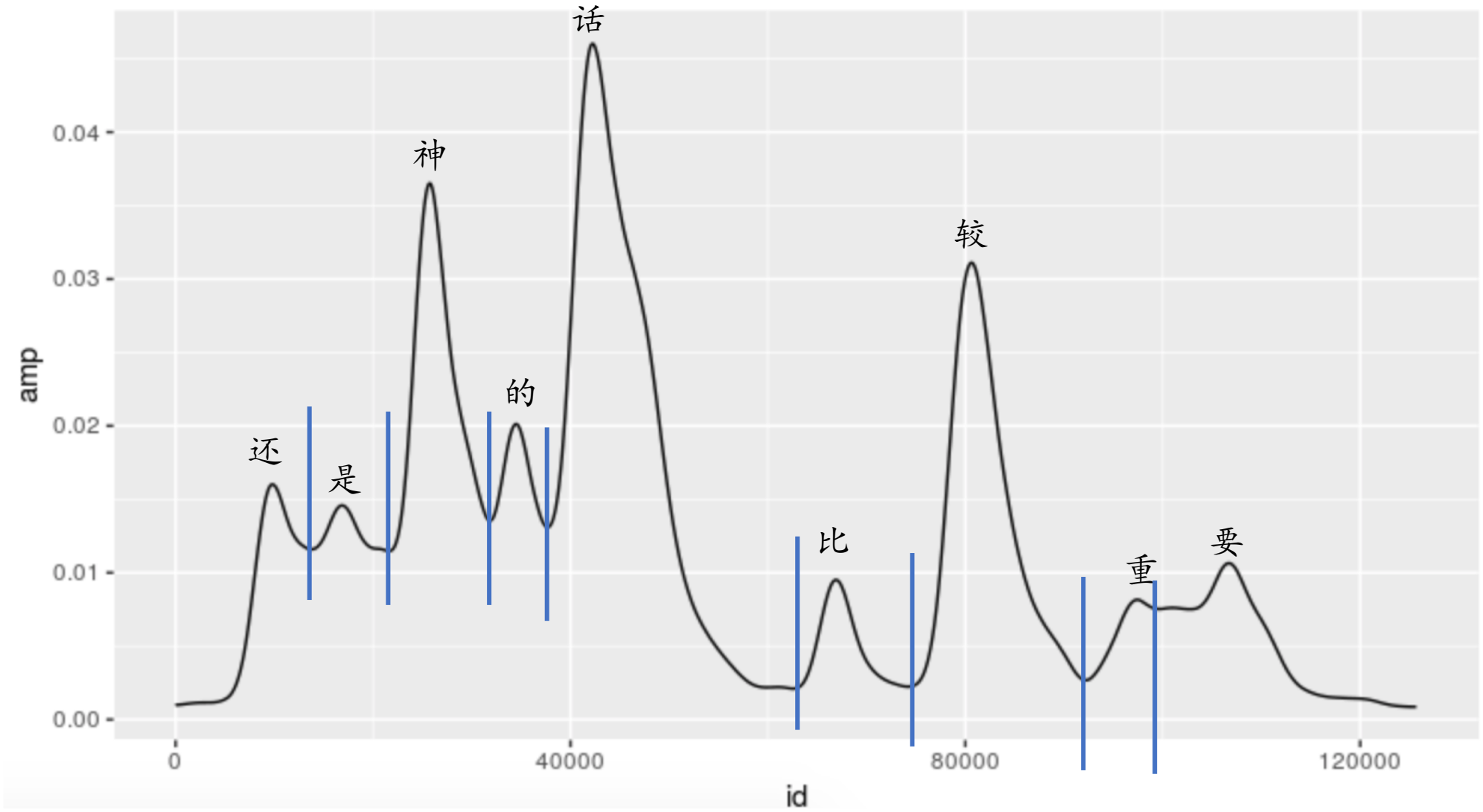
- Why?
  - Purpose is to make sure the algorithm does not identify very small bumps
- Define a new parameter to require certain number of increases/decreases for a peak

Define another parameter to remove small bumps

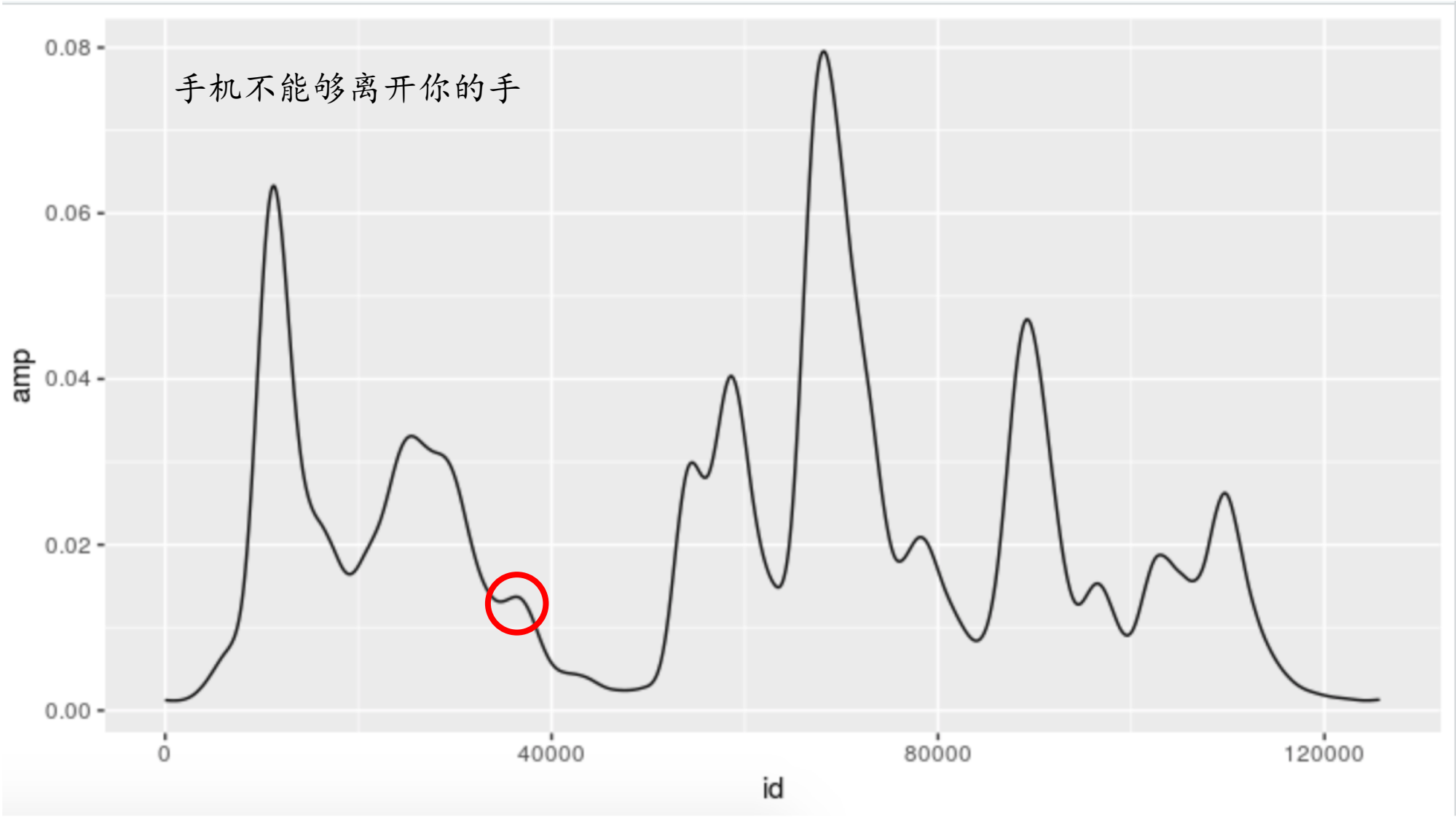
```
findpeaks(tmp3$amp, floor(44100*pace*peak.pace.ratio), minpeakheight = min.peak.amp) %>%
as.data.frame %>%
   rename(cut = V1, cut.id = V2, cut.start = V3, cut.end = V4) %>%
   left_join(tmp3, by = c('cut.id' = 'id')) %>% rename(cut.amp = amp) %>%
   left_join(tmp3, by = c('cut.start' = 'id')) %>% rename(cut.start.amp = amp) %>%
   left_join(tmp3, by = c('cut.end' = 'id')) %>% rename(cut.end.amp = amp) %>%
   mutate(start.ratio = cut.start.amp / cut.amp, end.ratio = cut.end.amp/cut.amp) %>%
   mutate(remove = (start.ratio > max.peak.ratio | end.ratio > max.peak.ratio))
```



Validate the algorithm other sentences



Validate by the 3rd sentence



How to report at this step?

Audio data

Algorithm to decompose words (based on amplitude)

FFT on each word

- Developed an algorithm to break words by smoothing waves followed by a gradient search algorithm
- Validated the algorithm by 3 sentences (to be validated with more, metric to be determined)

Derive features from FFT

Clustering algorithm (based on frequency)

Supervised learning (TBD)

Failed to jump over a few steps...

Audio data

Algorithm to decompose words (based on amplitude)

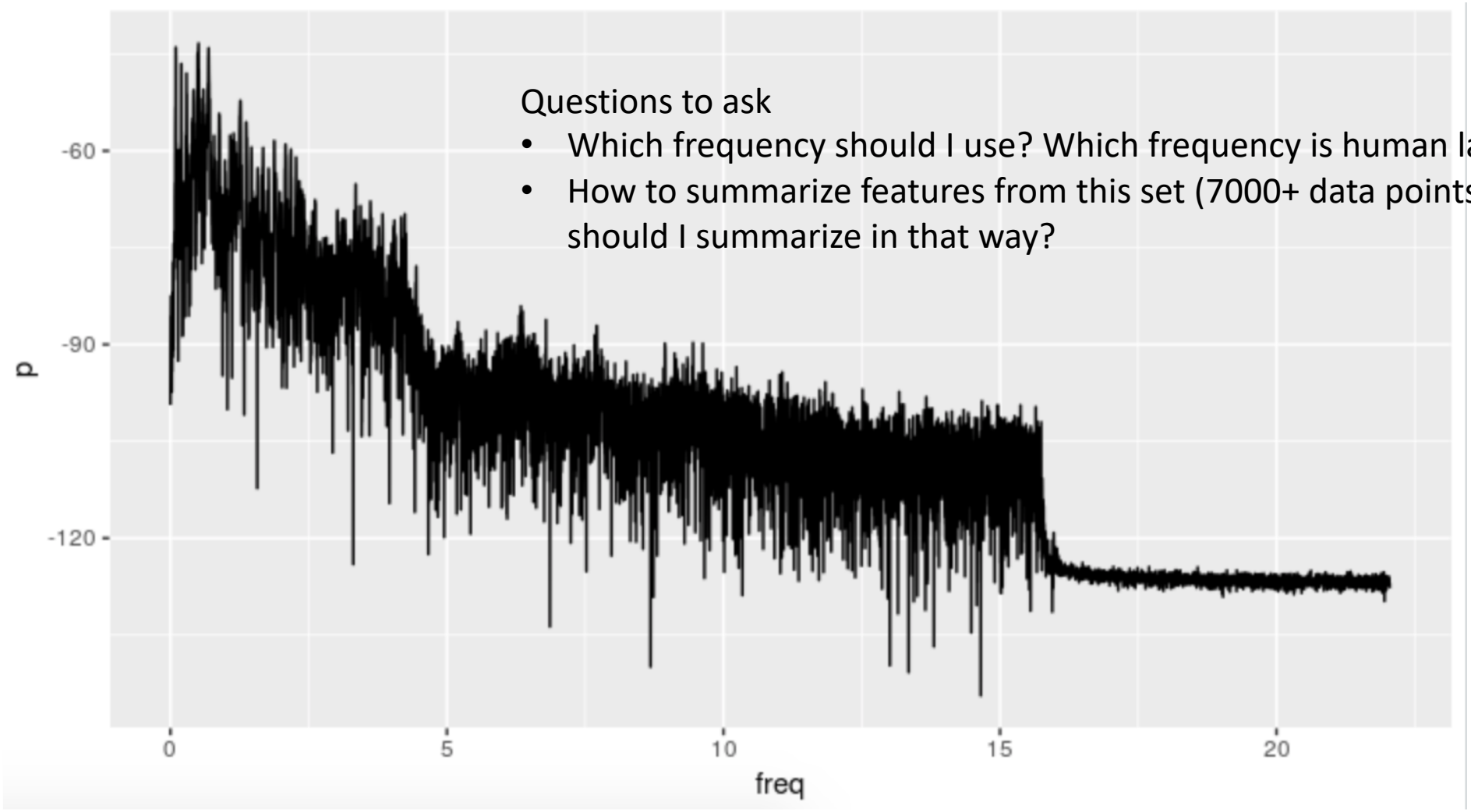
FFT on each word

Derive features from FFT

Clustering algorithm (based on frequency)

Supervised learning (TBD)

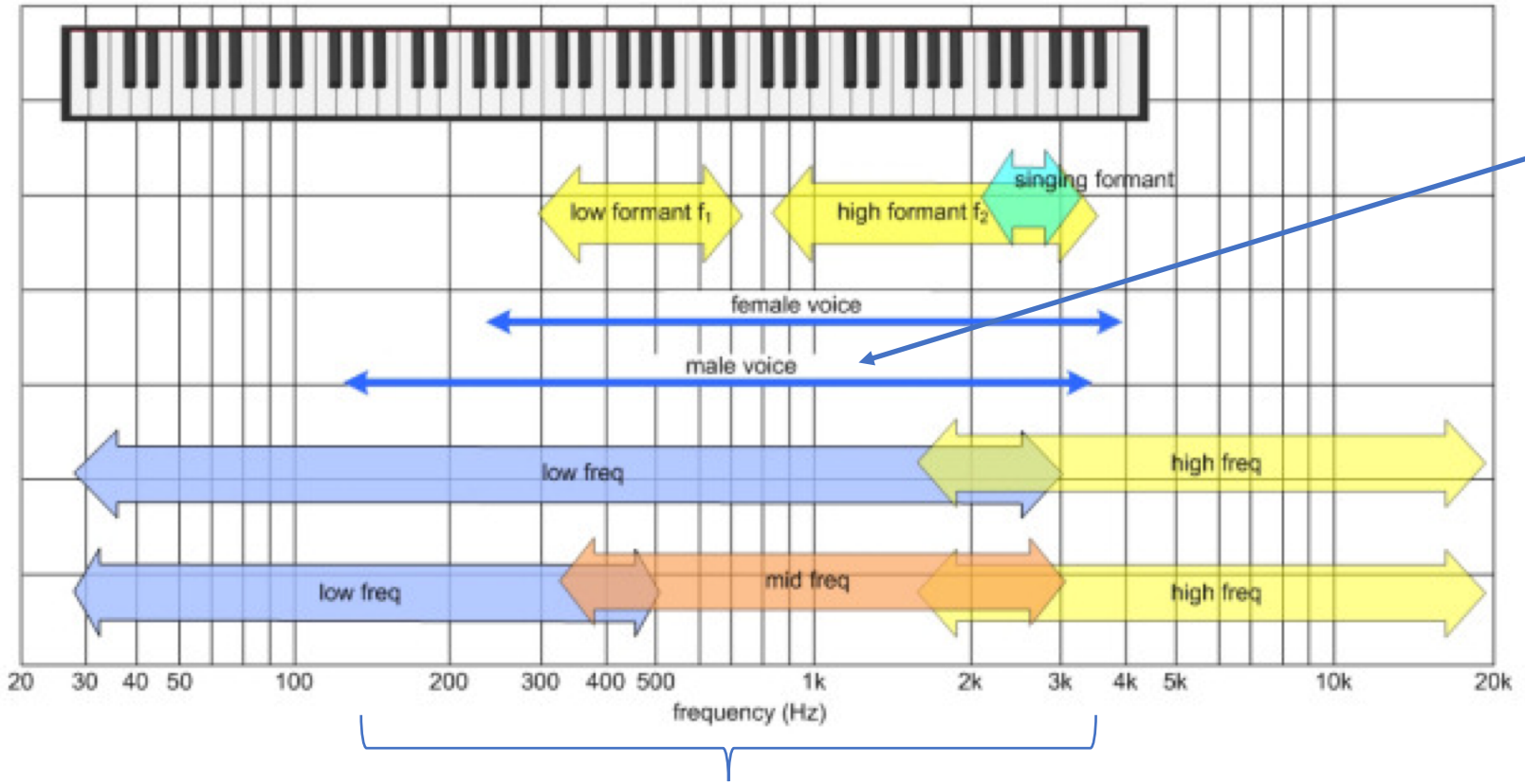
Ask question by checking FFT of one single word



Questions to ask

- Which frequency should I use? Which frequency is human language?
- How to summarize features from this set (7000+ data points) and why should I summarize in that way?

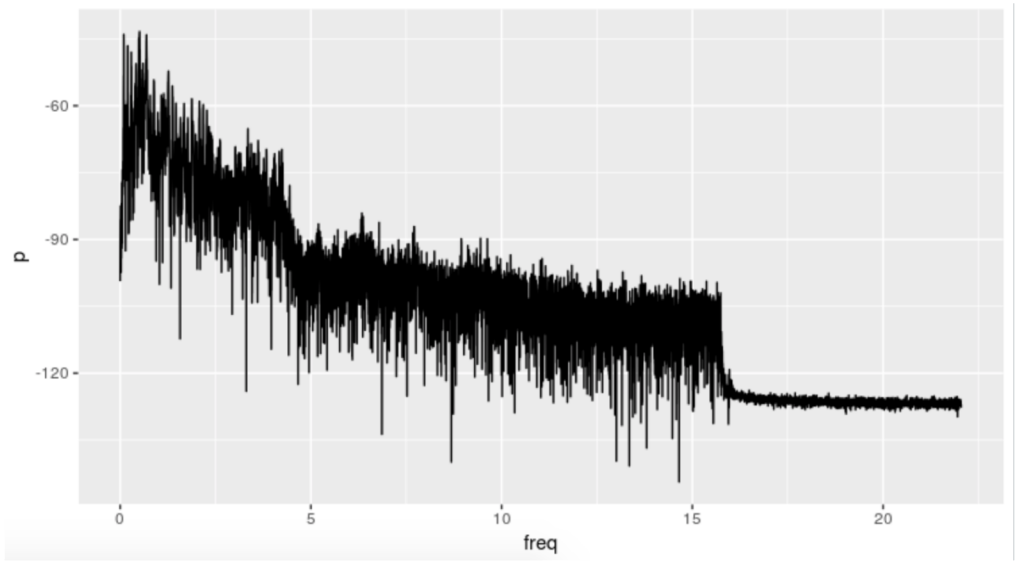
Search and understand the range of frequency for male voice...



Frequency range of male voice

Leverage the frequency cuts to define features

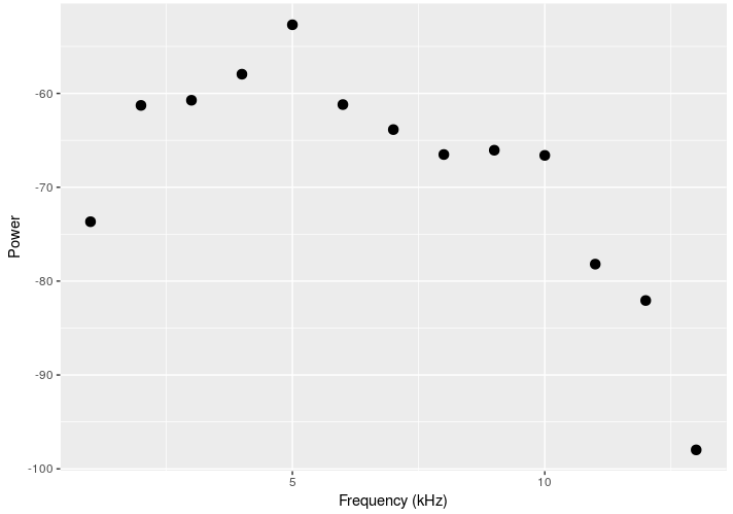
Propose a new workflow for defining features



Define frequency ranges

```
band = case_when(freq >= 0 & freq < 0.1 ~ 0.05,
                 freq >= 0.1 & freq < 0.2 ~ 0.15,
                 freq >= 0.2 & freq < 0.3 ~ 0.25,
                 freq >= 0.3 & freq < 0.4 ~ 0.35,
                 freq >= 0.4 & freq < 0.5 ~ 0.45,
                 freq >= 0.5 & freq < 0.6 ~ 0.55,
                 freq >= 0.6 & freq < 0.7 ~ 0.65,
                 freq >= 0.7 & freq < 0.8 ~ 0.75,
                 freq >= 0.8 & freq < 0.9 ~ 0.85,
                 freq >= 0.9 & freq < 1 ~ 0.95,
                 freq >= 1 & freq < 2 ~ 1.5,
                 freq >= 2 & freq < 3 ~ 2.5,
                 T ~ 4)) %>%
```

Mean power by frequency range



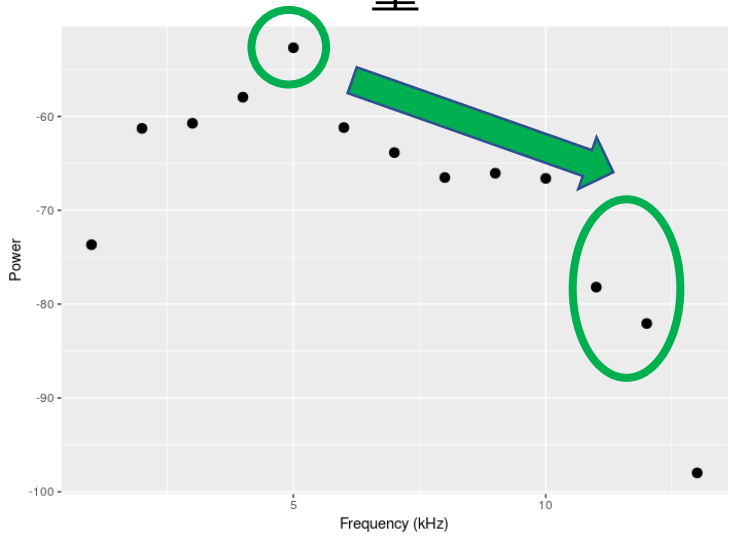
Define features, but how?

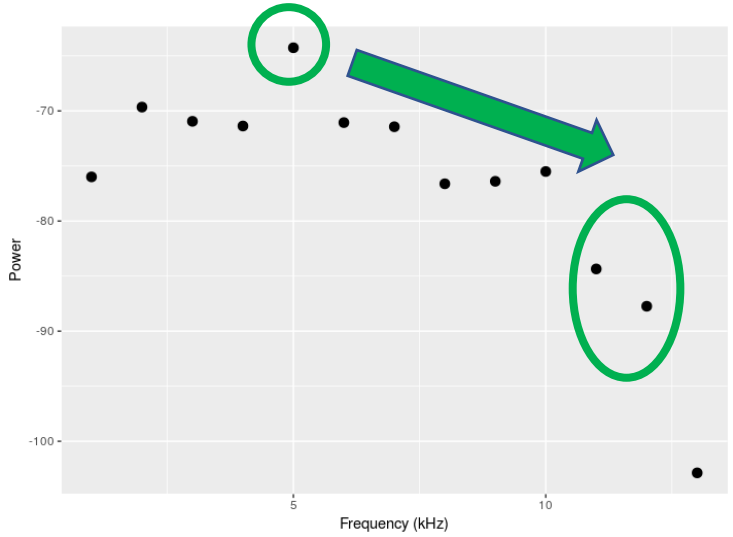
Ask a few questions

- How can I validate the aggregation?
- Is the aggregation enough to differentiate words?
- What analysis can I define to validate that the aggregation works?

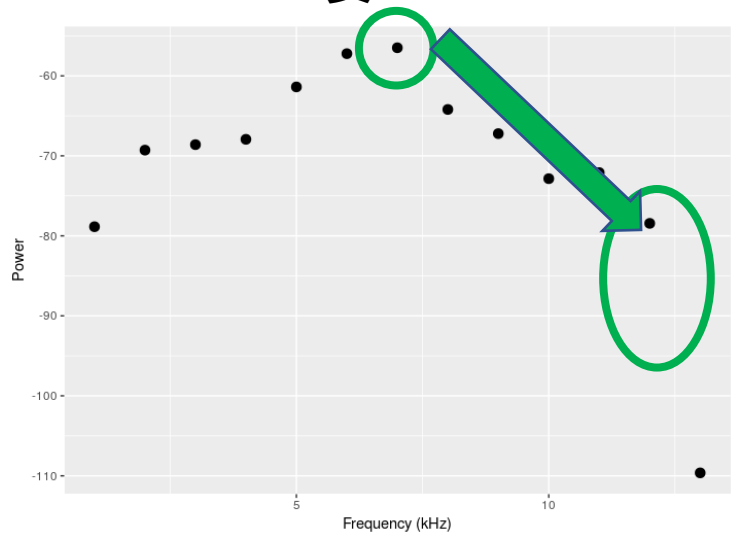
Checking four plots give us confidence and ideas to derive features!

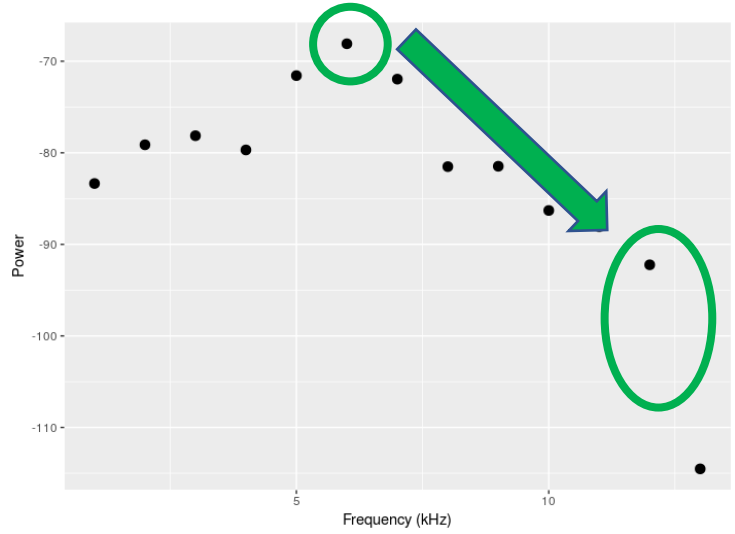
重





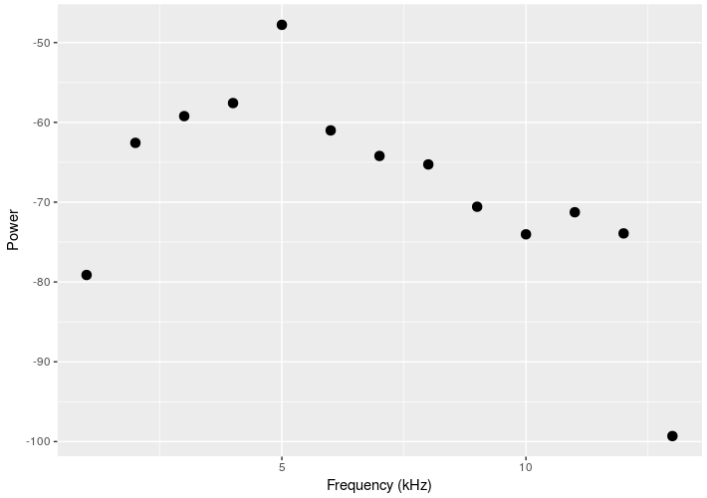
要

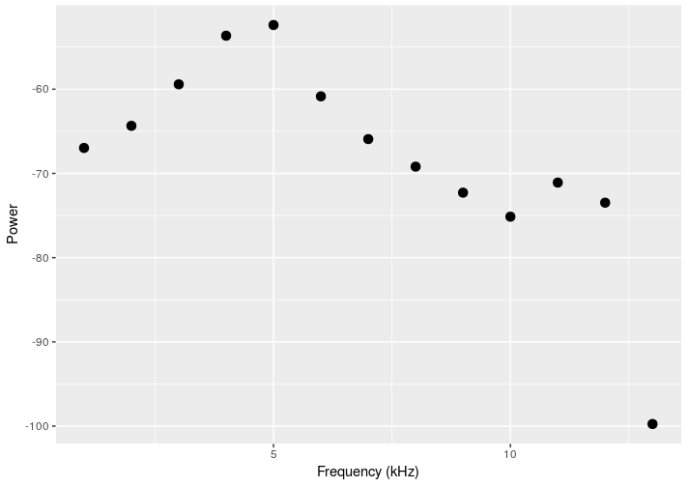




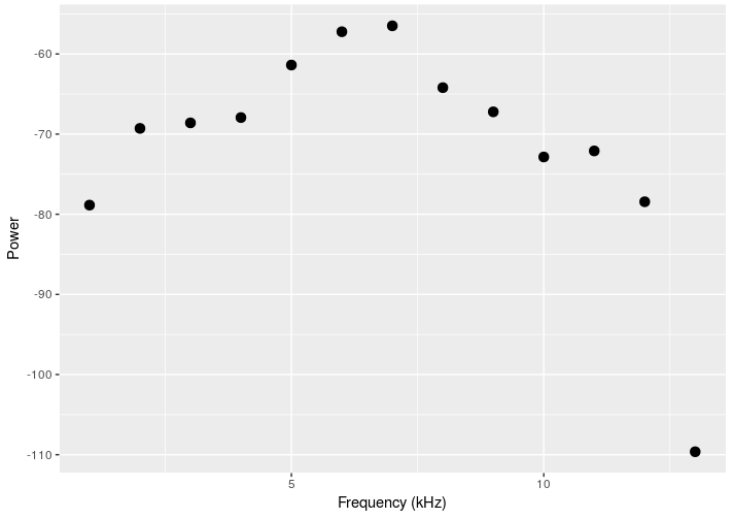
Another comparison!

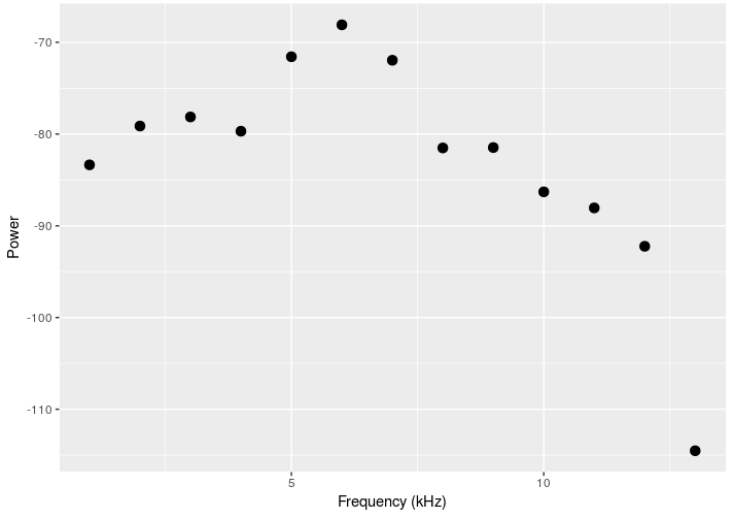
的



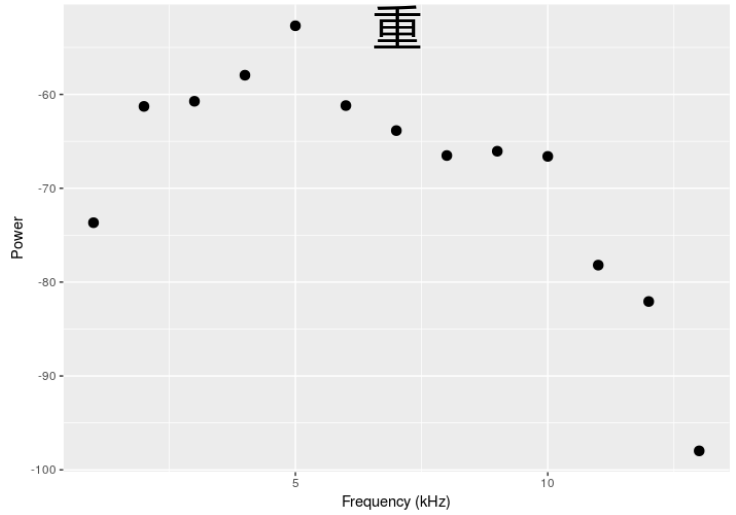


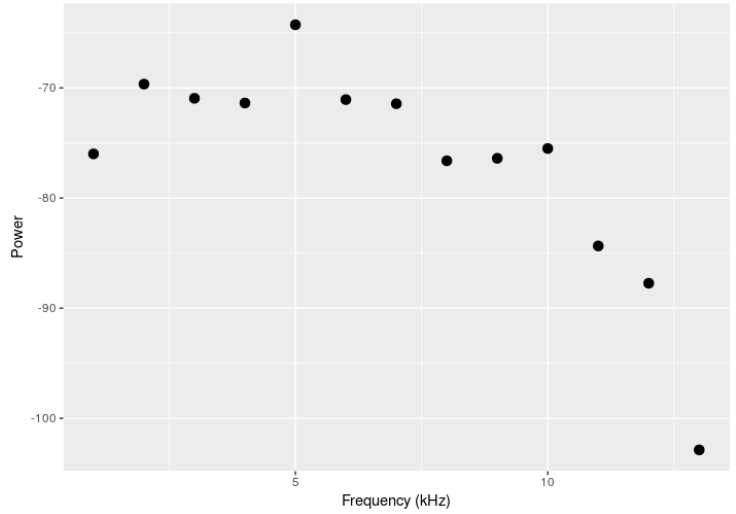
要

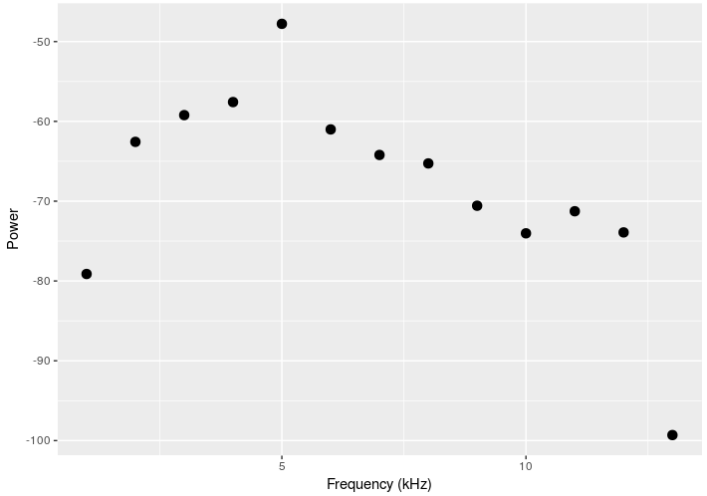




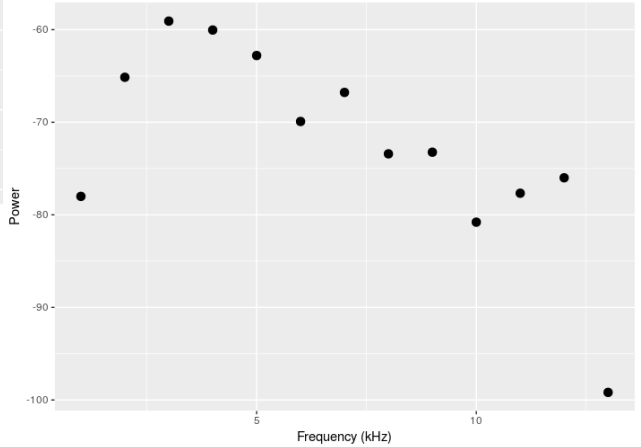
Another comparison

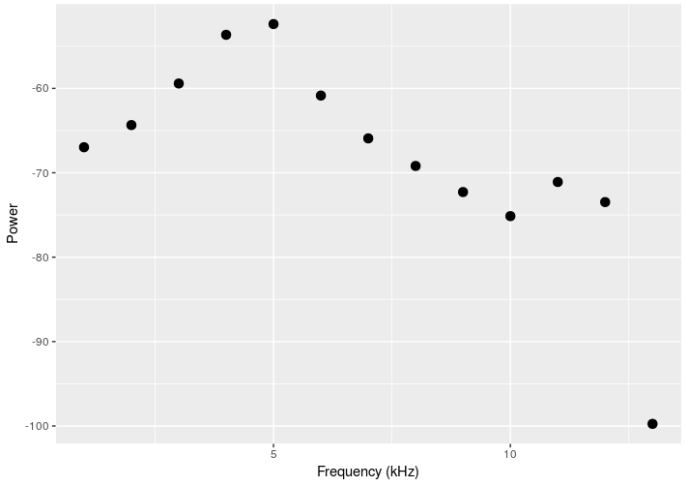




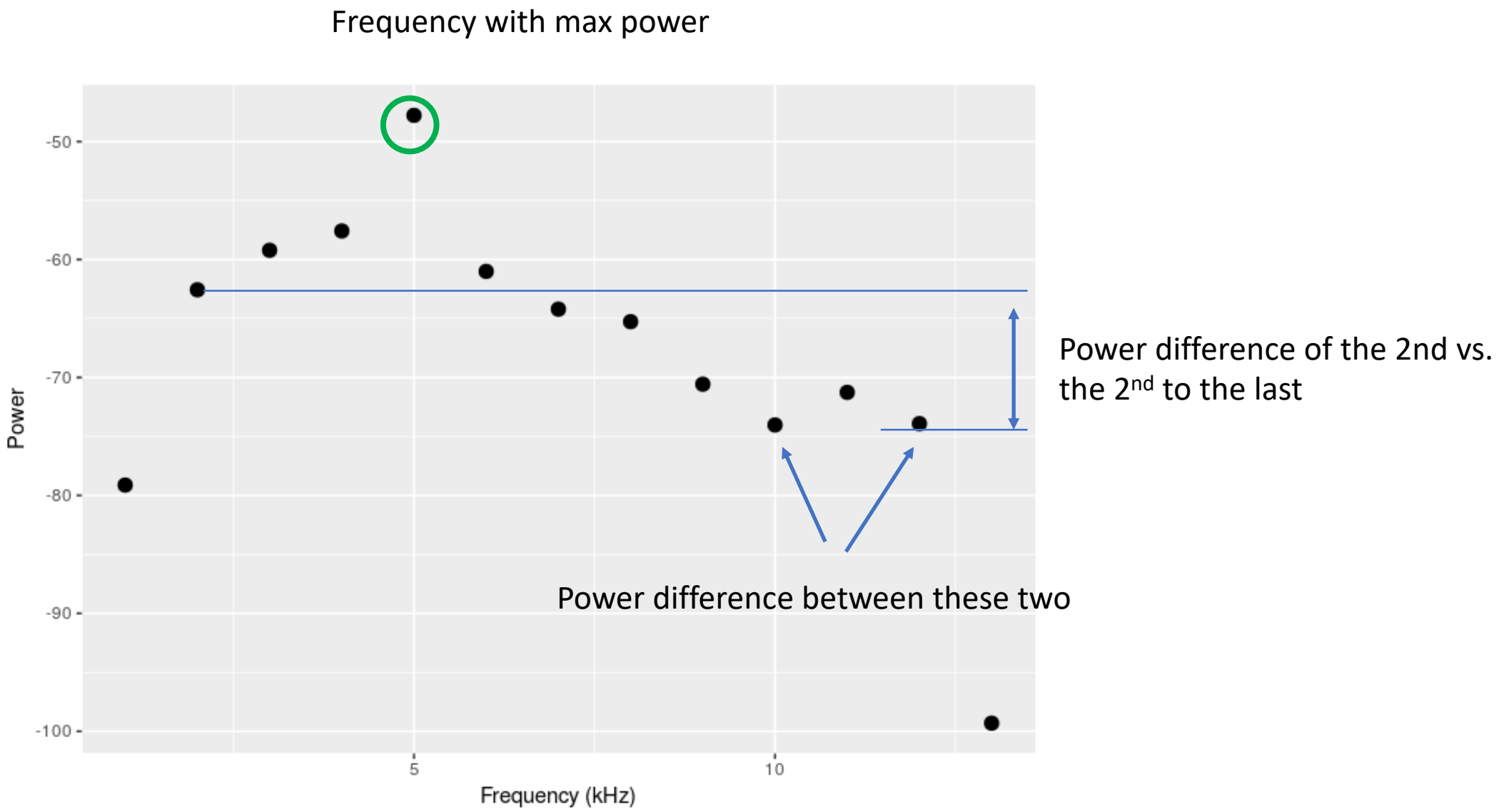


的

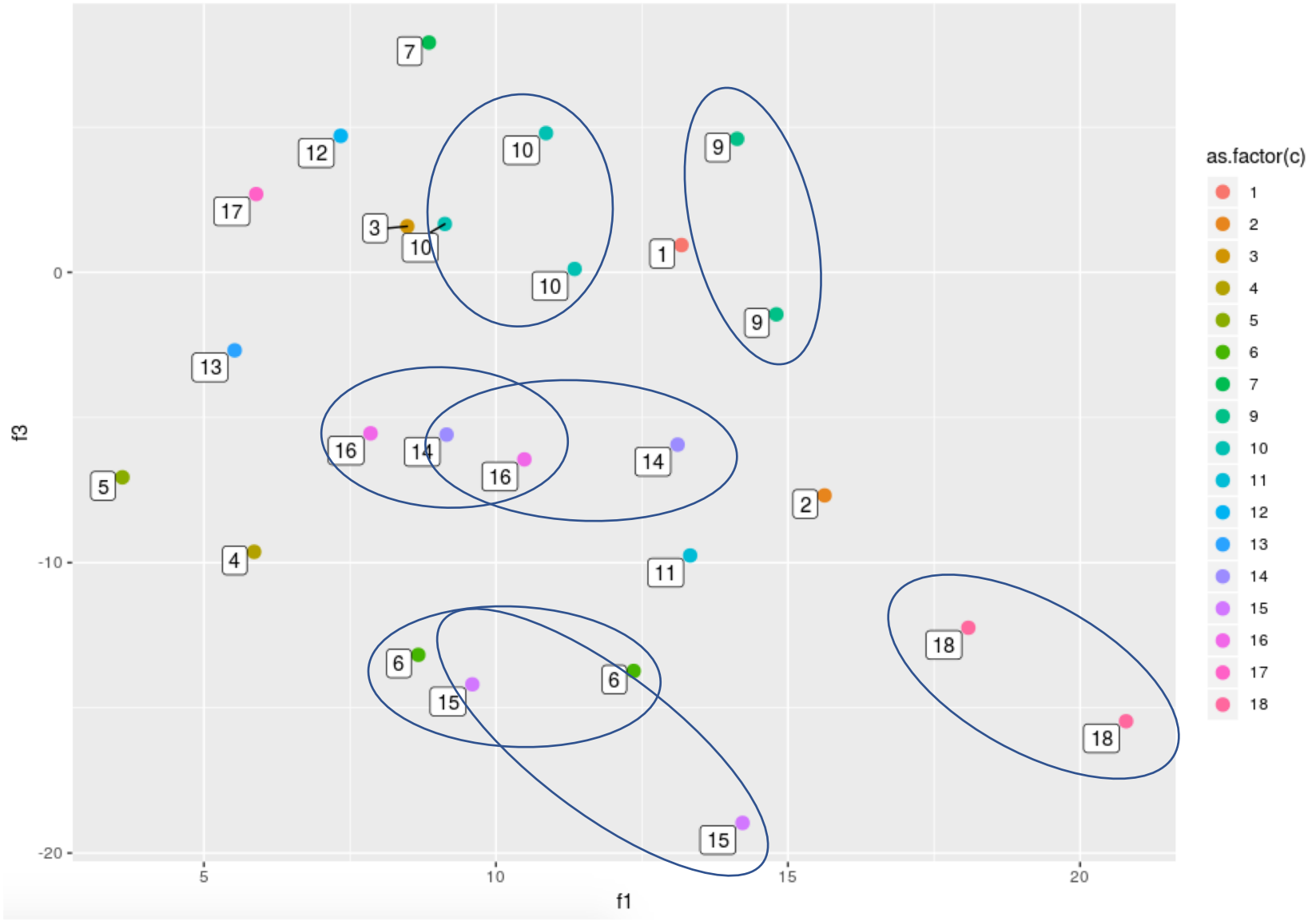




Now we can finally define some features!



Two features almost successfully cluster all 18 words



How to report now?

Audio data

Algorithm to decompose words (based on amplitude)

FFT on each word

Derive features from FFT

Clustering algorithm (based on frequency)

Supervised learning (TBD)

How to report now?

- Developed an algorithm to break words by smoothing waves followed by a gradient search algorithm
- Validated the algorithm by 3 sentences (to be validated with more, metric to be determined)
- Feature generation framework has been established and several features have been created with very good performance
- Next steps
  - Expand the features
  - Start to train a predictive model

Jump out to take a look at the big picture; solid progress has been made with tangible results and a lot of learnings

Simple validation ⇒ simple model

- What's the right tool to use?
- How large data can be handled? Is there transformation needed?
- What's the easiest way to formulate the problem? What's the easiest model to use?
- What's the easiest validation approach to use?

Extended validation ⇒ improved model

- Can we define some insight driven validation metrics?
- Can we leverage these validation metrics to improve the model performance?
- Is it needed to leverage CNN or other deep learning approaches?

Automated framework ⇒ insightful report

- What is the final outcome needed?
- How to understand the model accuracy vs outcome?
- What are the insights we can provide to the speaker?

不一样的人生——从圣经看工作

树立正确的工作观

- 正确的事奉观念
- 正确的工作观念
- 正确的家庭观念

- 什么是事奉？工作是不是事奉神？
- 圣经上教导什么样的工作观？
- 家庭和工作的关系应该如何看待？如何平衡优先次序？

正确的角度看工作

- 从圣经看工作
- 从永恒看工作
- 从安息看工作

- 工作的目的是什么？
- 工作在永恒中有没有价值？
- 忙碌的工作中如何归回安息？

翻转工作中的挑战

- 从逆境看得胜
- 从无常看改变
- 从野心看梦想

- 工作中压力很大，如何得胜和做见证？如何更好地使用恩赐？
- 我只想有一份简单的工作，可是公司中变幻无常，如何看待？
- 圣经允许有工作的野心吗？圣经中怎么样教导工作上的梦想？

实战问题解答

- 什么时候加班什么时候不加班？每天八小时工作不多不少可以吗？
- 刚到一个新环境，如何很快地与人融洽相处？
- 如何看待北美华人在职场glass ceiling的问题？
- 工作上遭遇不公平时如何处理？
- 一个家里只有一个人工作还是两个人工作？
- ……