Getting vocal in R

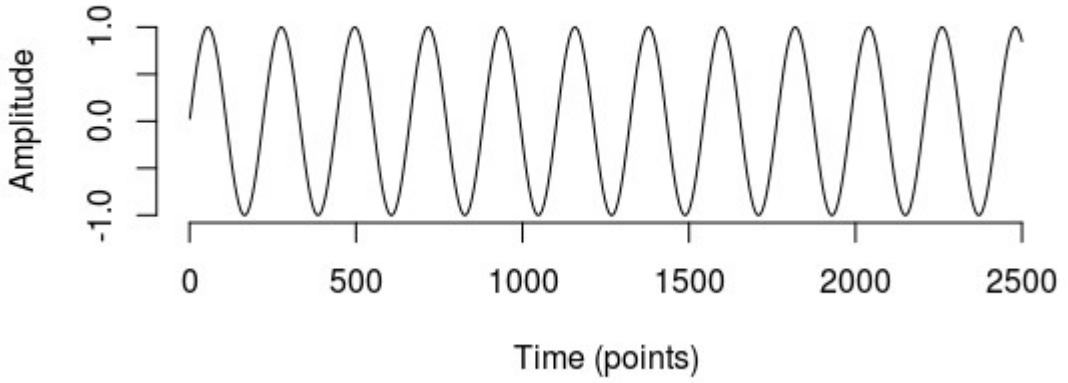
Andrey Anikin
andrey.anikin@lucs.lu.se

Project web page:
http://cogsci.se/practice/sound_synthesis/scream.html

Contents

Part 1: sound generation

- Digital audio is a vector of floats/integers with /samplingRate/ points per second
- library(tuneR)
  load / play / save .wav audio files
- Human voice can be modeled as a filtered sum of sinusoids (fundamental frequency + harmonics) and stochastic component (noise)

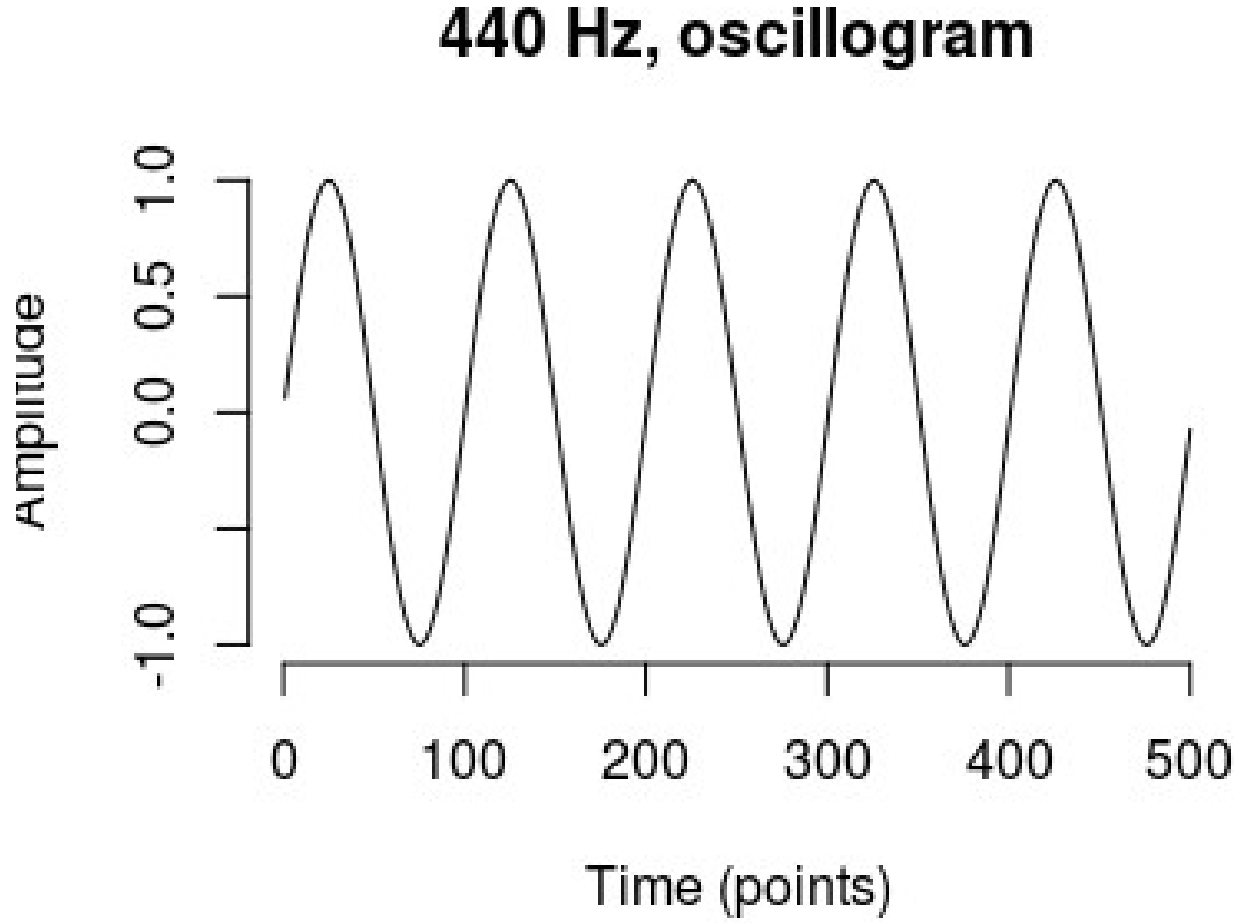


Minimal R code: generate concert A (440 Hz)

```
library(tuneR)
pitch_Hz = 440 # constant pitch 400 Hz
duration_s = .5 # half a sec in duration
samplingRate = 44100 # 44100 points/s

# generate a vector of amplitudes
pitchContour = rep(pitch_Hz, duration_s * samplingRate) # a vector
integr = cumsum(pitchContour) / samplingRate # integral of pitch contour
ampl = sin(2*pi*integr) # our target

# normalize and save as pcm WAV
soundWave = Wave(left=ampl, samp.rate=samplingRate, bit=16, pcm=TRUE)
soundWave = normalize (soundWave, unit='16') # 16-bit precision (audio quality)
play (soundWave, 'play') # play in R
writeWave (soundWave, filename='example.wav') # save as a .wav file
```

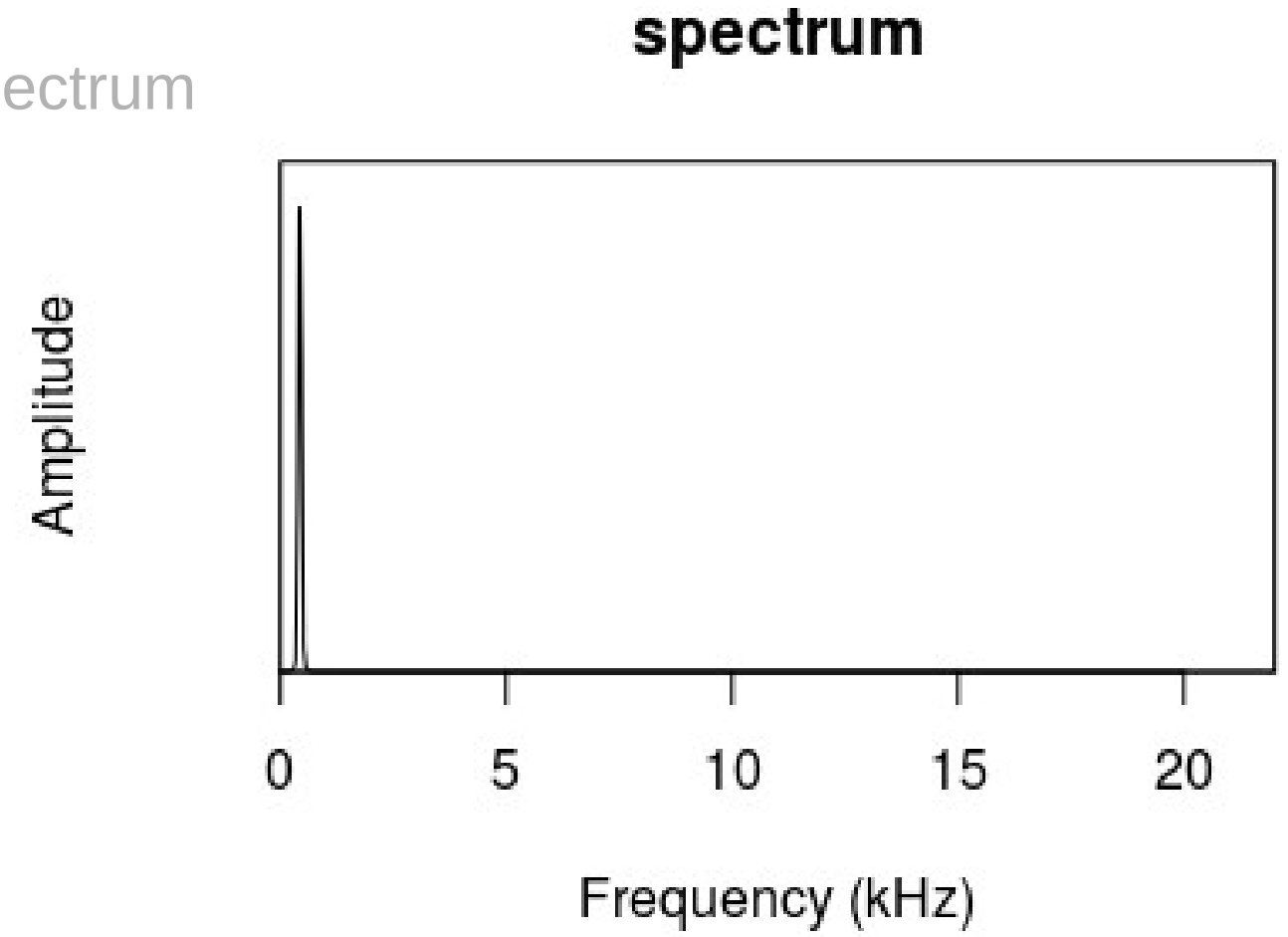
Three basic plots: oscillogram

```
plot (ampl[1:500], type='l', xlab='Time (points)',
ylab='Amplitude', bty='n', main='440 Hz, oscillogram')
# plot the first ~10 ms of the resulting sound vector
```



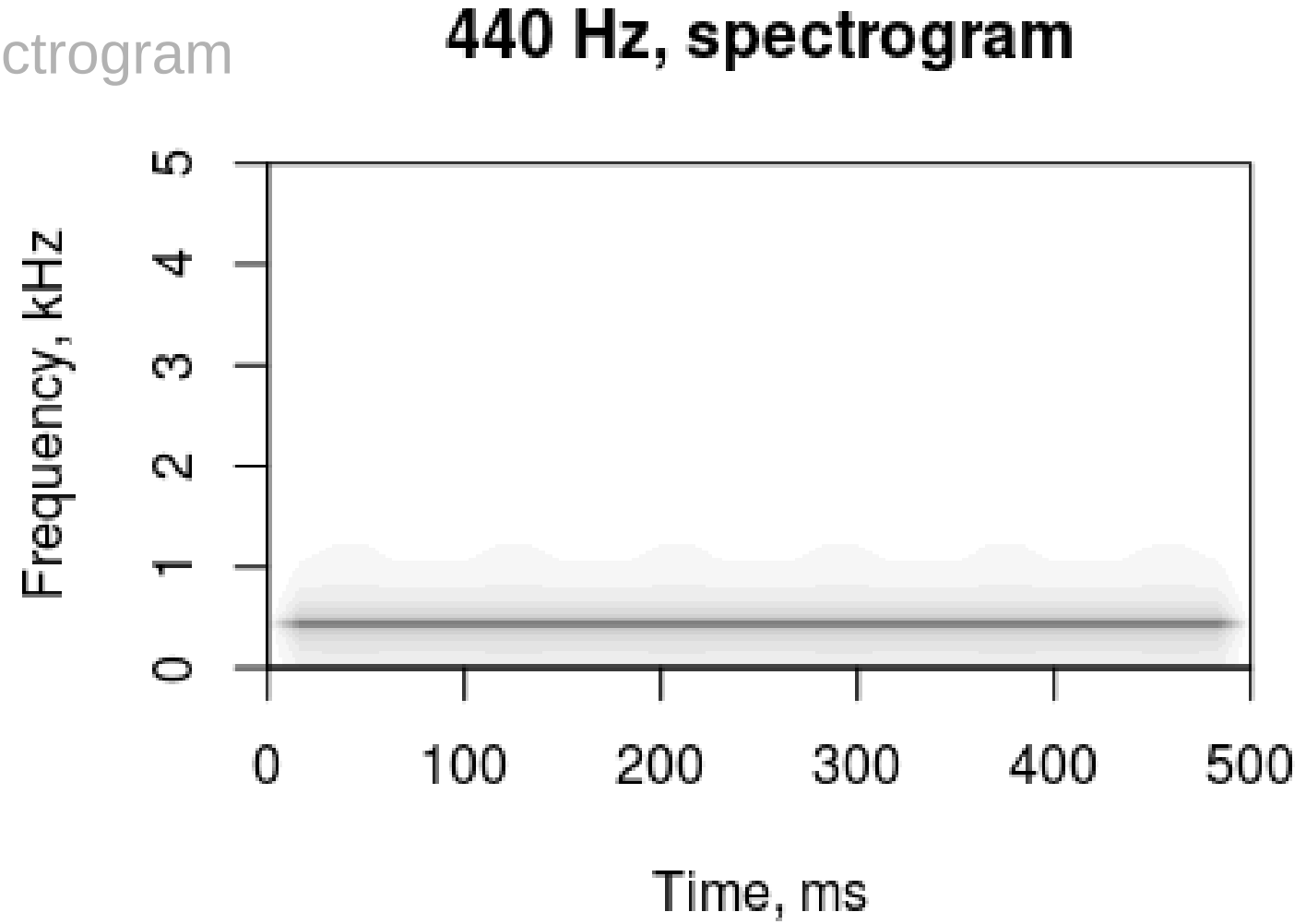
Three basic plots: spectrum

```
library (seewave)
meanspec () # spectrum
```



Three basic plots: spectrogram

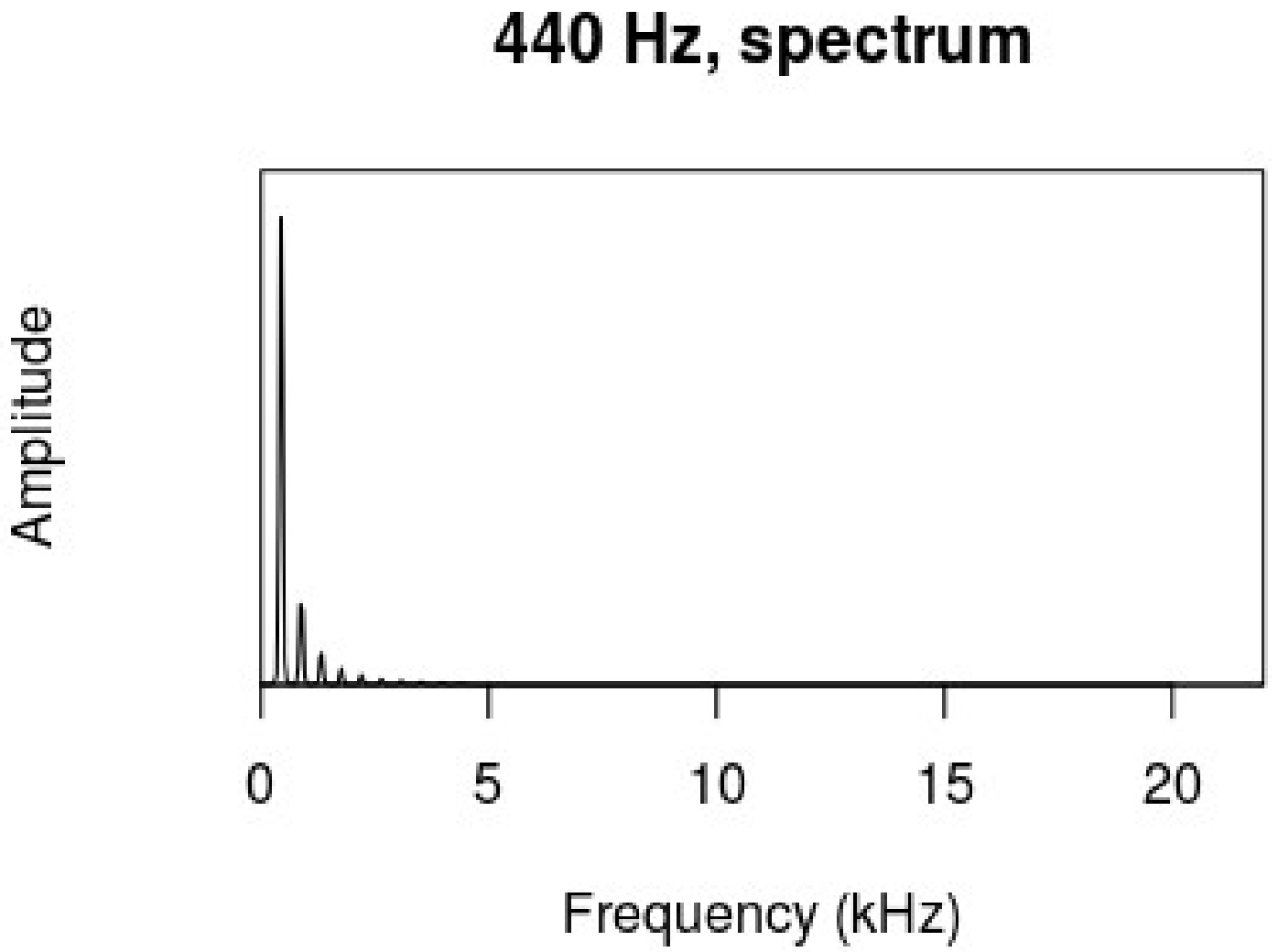
```
library (seewave)
spectro () # spectrogram
```



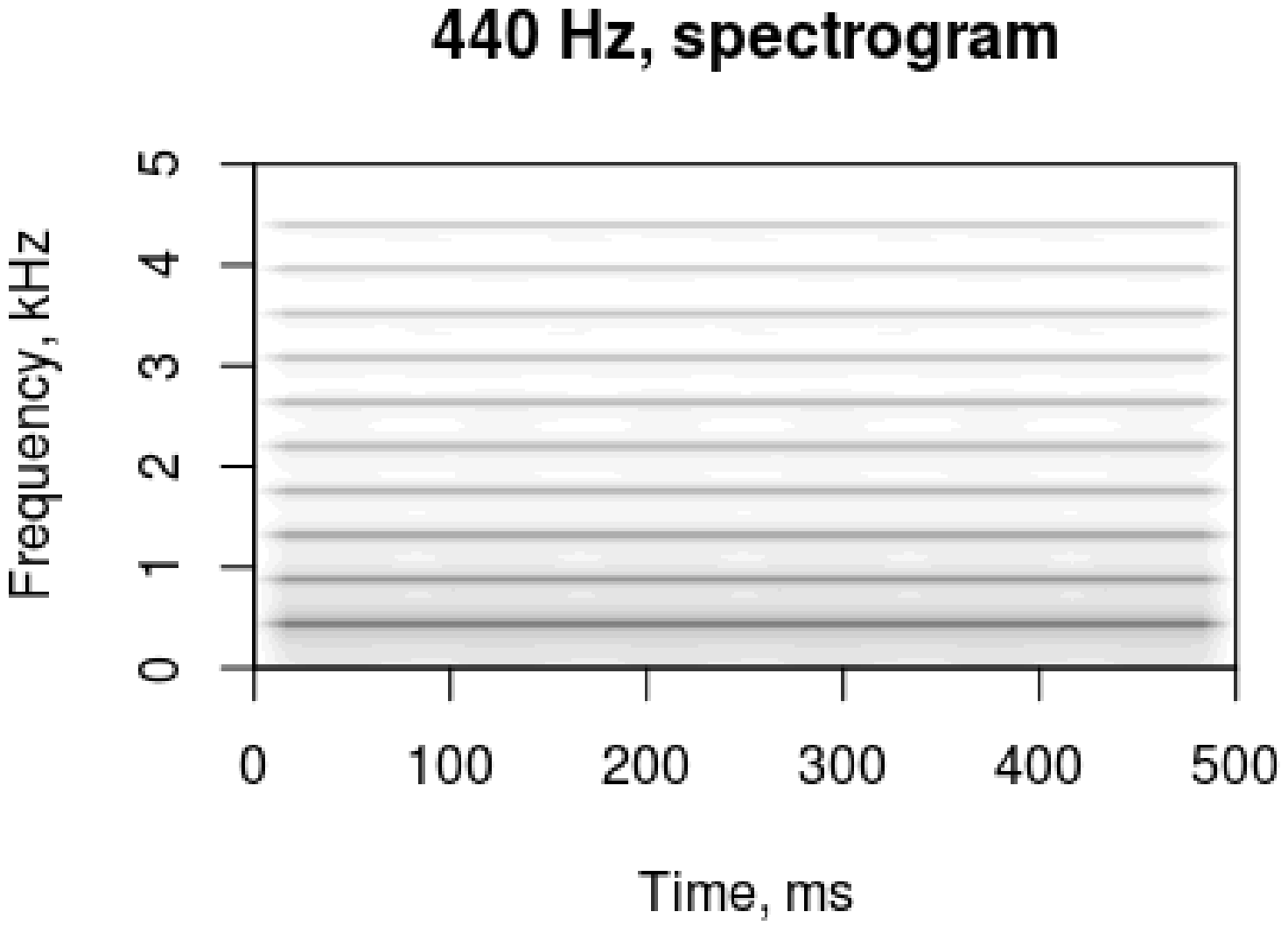
Add harmonics

```
nHarmonics = 10 # how many harmonics?
rolloff = 24 # loss of energy in harmonics, dB/octave


ampl = rep(0, length(integr))
for (h in 1:nHarmonics){
  ampl = ampl + sin(2*pi*h*integr) * h^(-rolloff/10)
}
```

Add harmonics

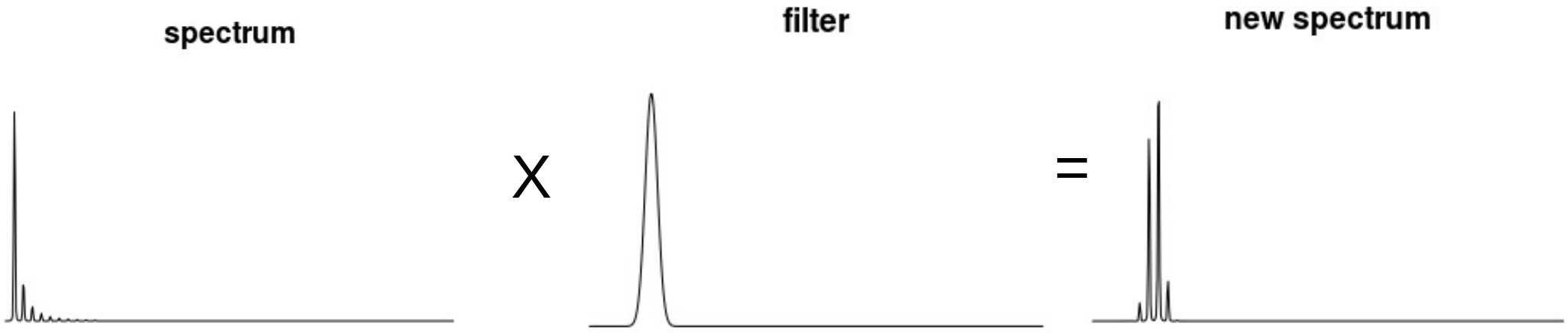


Add harmonics



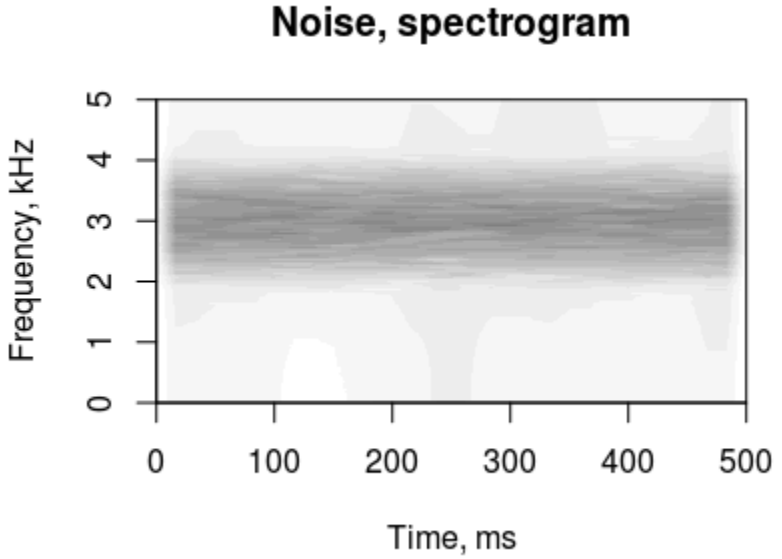
Filter the spectrum (vowels)

- Do FFT of sound with harmonics (from time series to spectrum)
- Multiply spectrum by filter
- Do inverse FFT (from spectrum to time series)



Breathing noise

- Generate white noise:
  runif (n=length_points, min=-1, max=1)
- Do FFT
- Multiply by filter (same as for harmonics)
- Do inverse FFT



Other effects

- Jitter: random variation in pitch per glottal cycle (noise)
- Vibrato (like an opera singer)
- Subharmonics (“vocal fry”)
- Etc

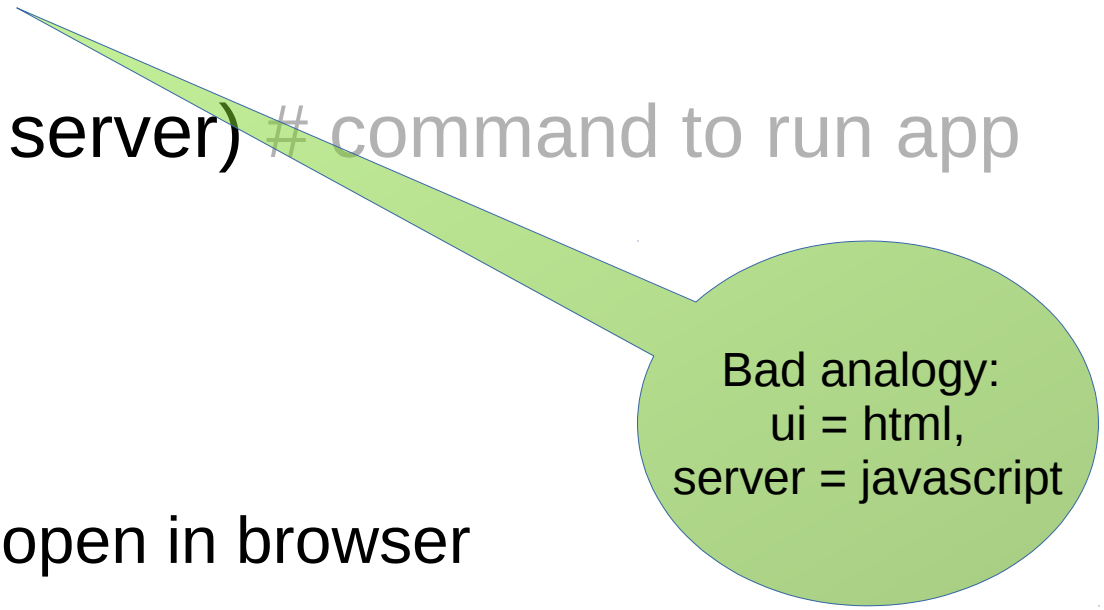
Contents

Shiny R

- http://shiny.rstudio.com/
- Interactive web applications written in R
- Online hosting: own server or cloud

Shiny app: basic structure

```
ui = fluidPage(…) # UI: layout, inputs

server = function(input, output, session) {…} # R scripts
for processing inputs

shinyApp (ui = ui, server = server) # command to run app
```



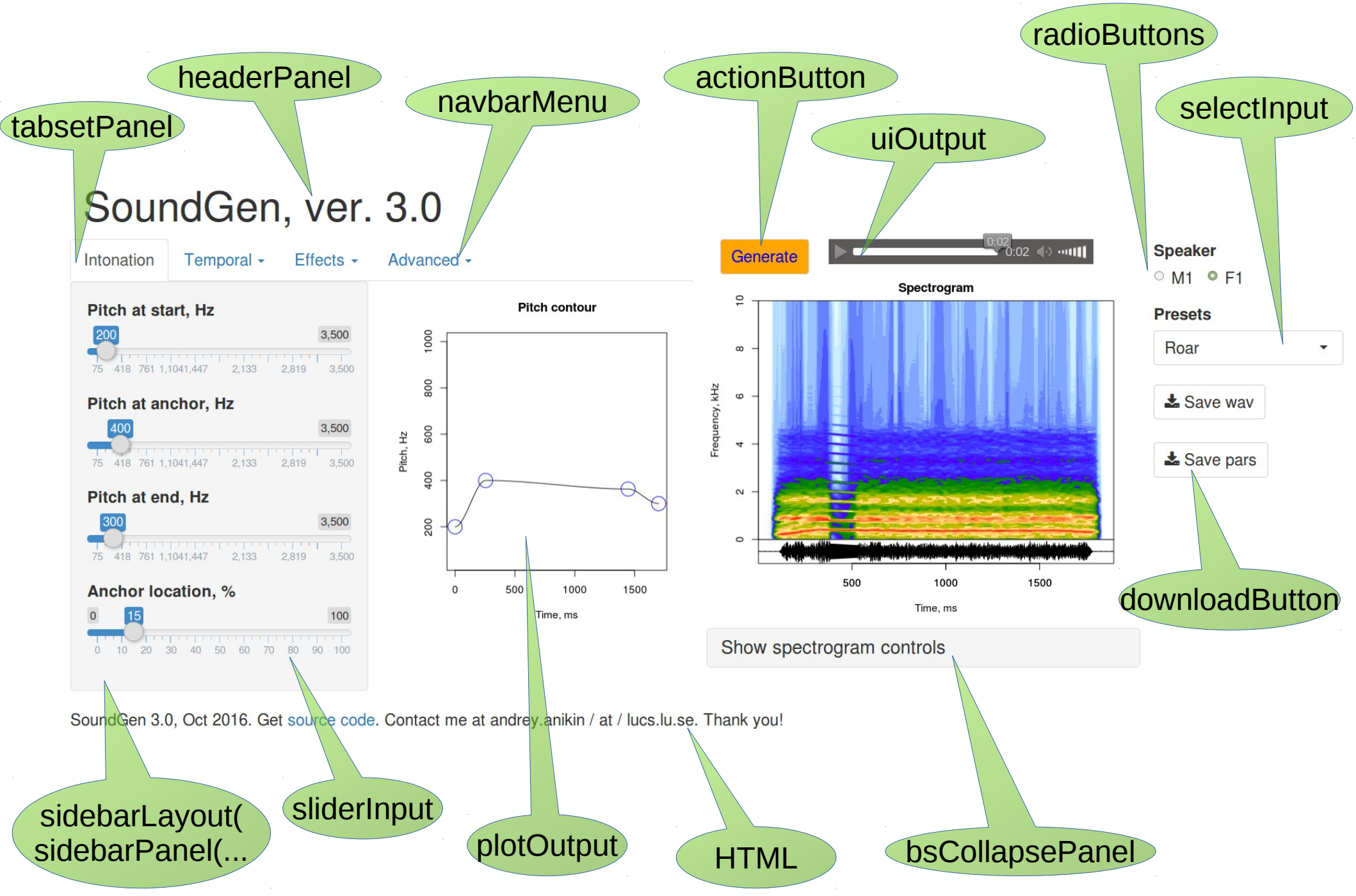
- To run locally:
  in RStudio, “Run app” to open in browser
- To run online:
  publish to server in 2 clicks

Shiny app: minimal example

```
ui <- fluidPage(

  sliderInput ('pitch_start', 'Pitch
at start, Hz', value=120, min=50,
max=300, step=25),

  uiOutput ("myPar")

)
```


```
server <- function(input, output, session) {
  output$myPar = renderUI (paste ('You have set pitch to',
input$pitch_start, 'Hz'))
}

shinyApp(ui = ui, server = server)
```

Shiny app: heavy lifting

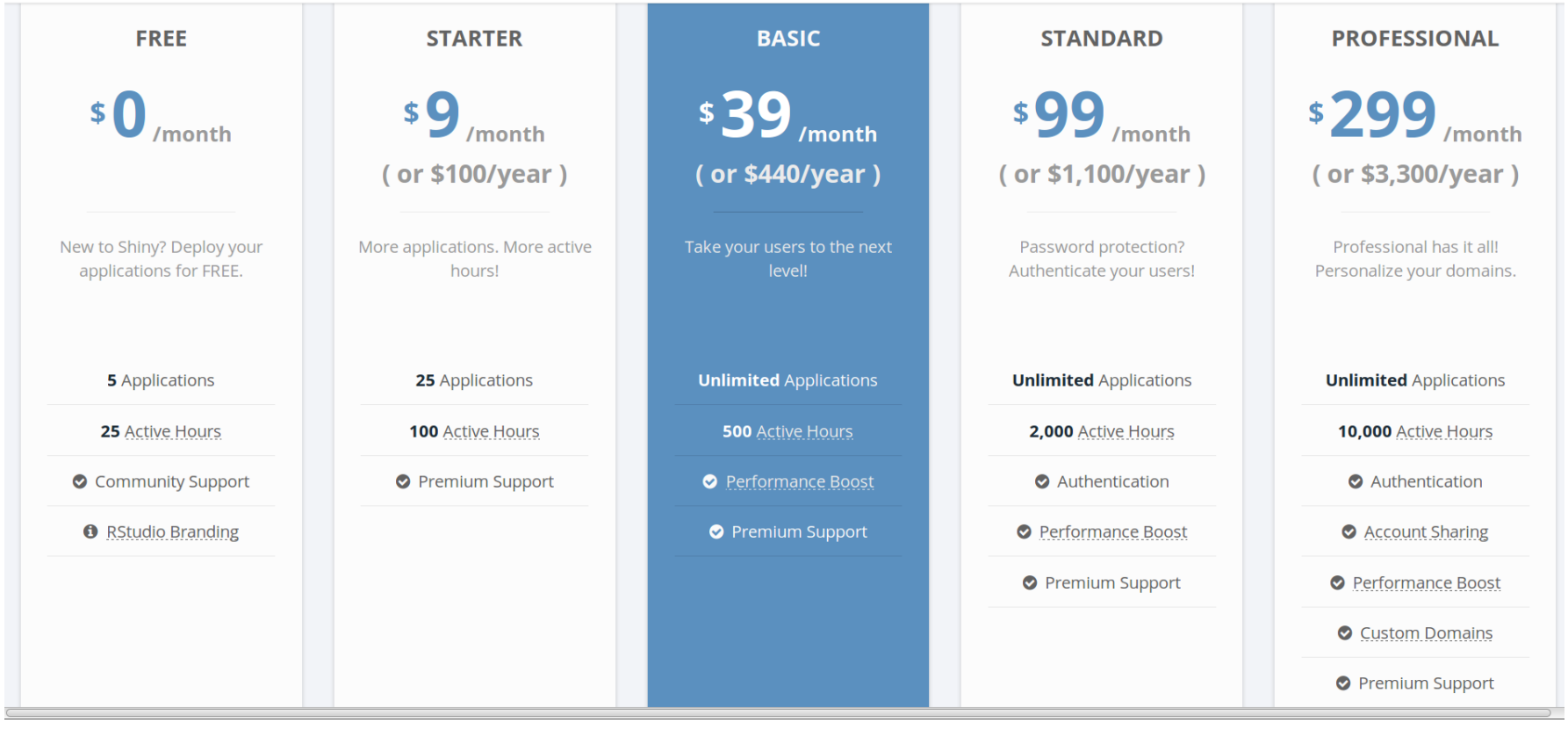


Under the hood: server(){}

- updateSliderInput() # force new slider value
- reactiveValues() # store reactive values
- reactive() # take user-provided input
- renderUI() # generate HTML
- observeEvent() # event listener
- renderPlot() # create a plot
- downloadHandler() # set up file download

Learn more about Shiny

- Tutorial: http://shiny.rstudio.com/tutorial/
- Examples: http://shiny.rstudio.com/gallery/
- Cloud hosting: http://www.shinyapps.io/



Thank you!

Project web page:
http://cogsci.se/practice/sound_synthesis/scream.html