

Getting vocal in R

Andrey Anikin
andrey.anikin@lucs.lu.se

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

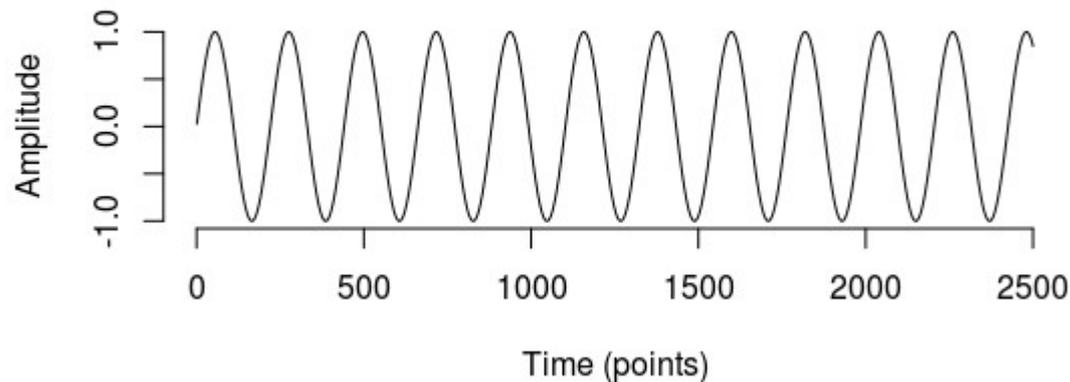
Contents

Part 1: sound generation

Part 2: interactive UI

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

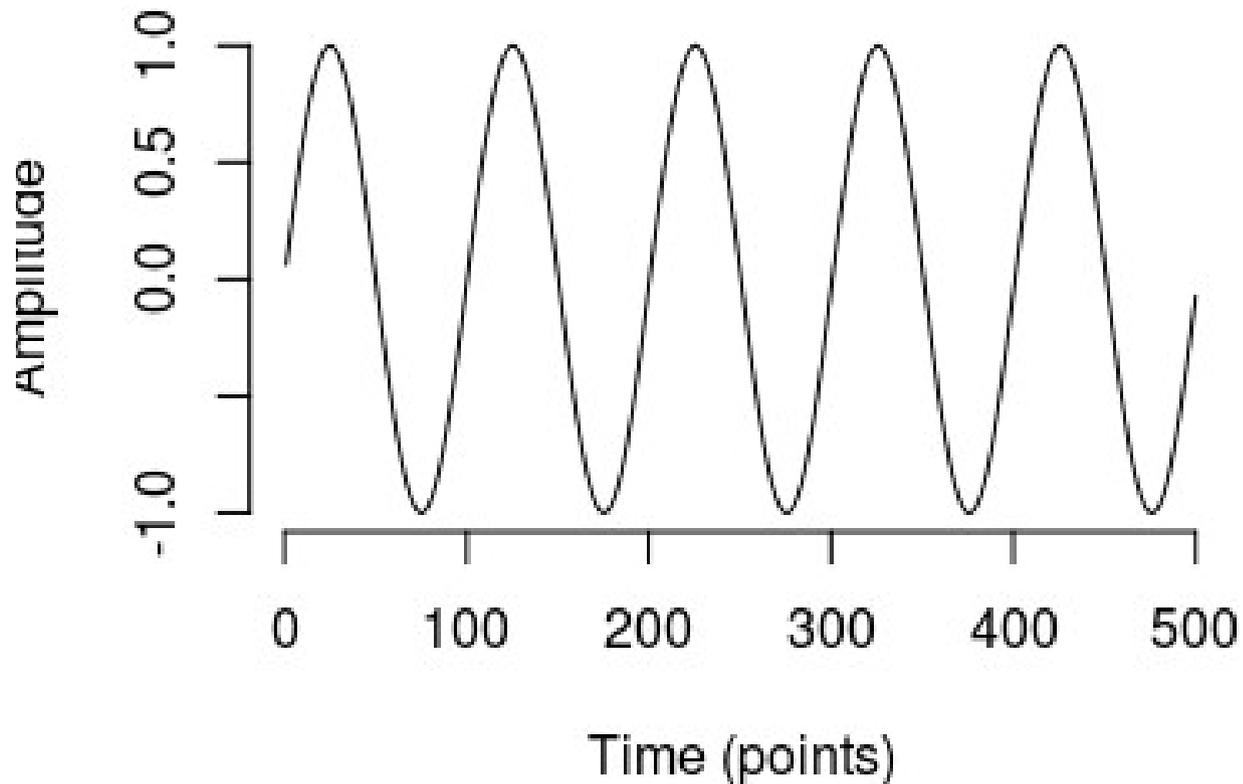
# generate a vector of amplitudes
pitchContour = rep(pitch_Hz, duration_s * samplingRate) # a vector
integr = cumsum(pitchContour) / samplingRate # integral of pitch contour
ampl = sin(2*pi*integr) # our target

# 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
```

440 Hz, oscillogram

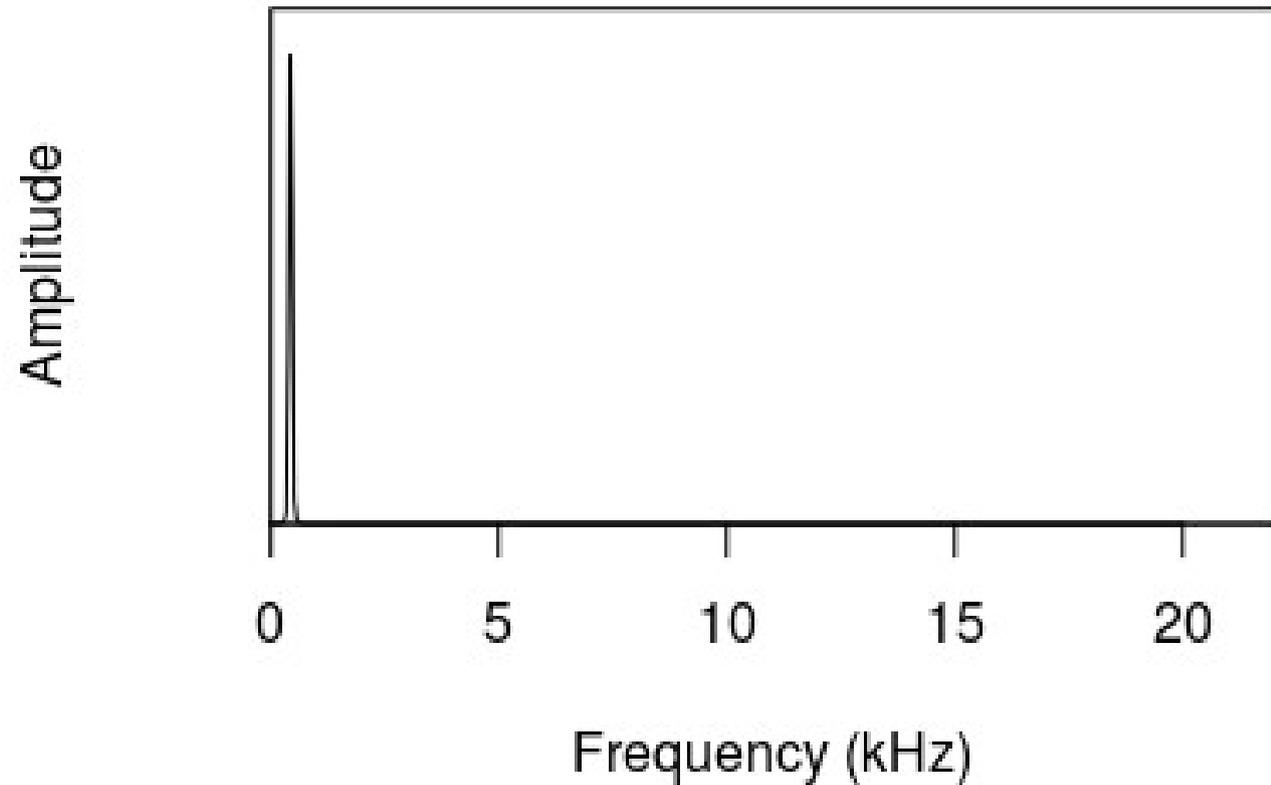


Three basic plots: spectrum

library (seewave)

`meanspec ()` # spectrum

spectrum

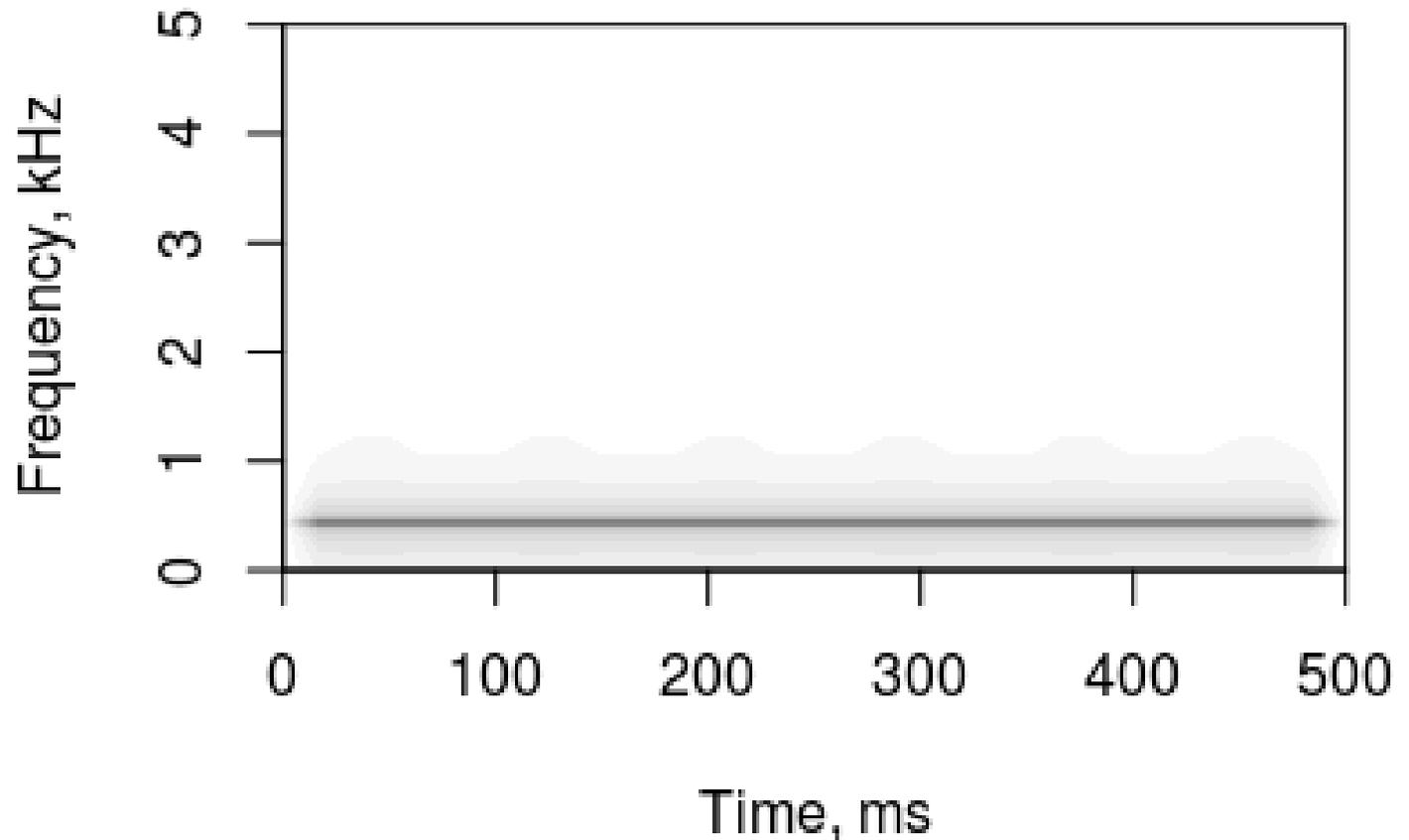


Three basic plots: spectrogram

library (seewave)

`spectro ()` # spectrogram

440 Hz, spectrogram



Add harmonics

nHarmonics = 10 # how many harmonics?

rolloff = 24 # loss of energy in harmonics,
dB/octave

```
ampl = rep(0, length(integr))
```

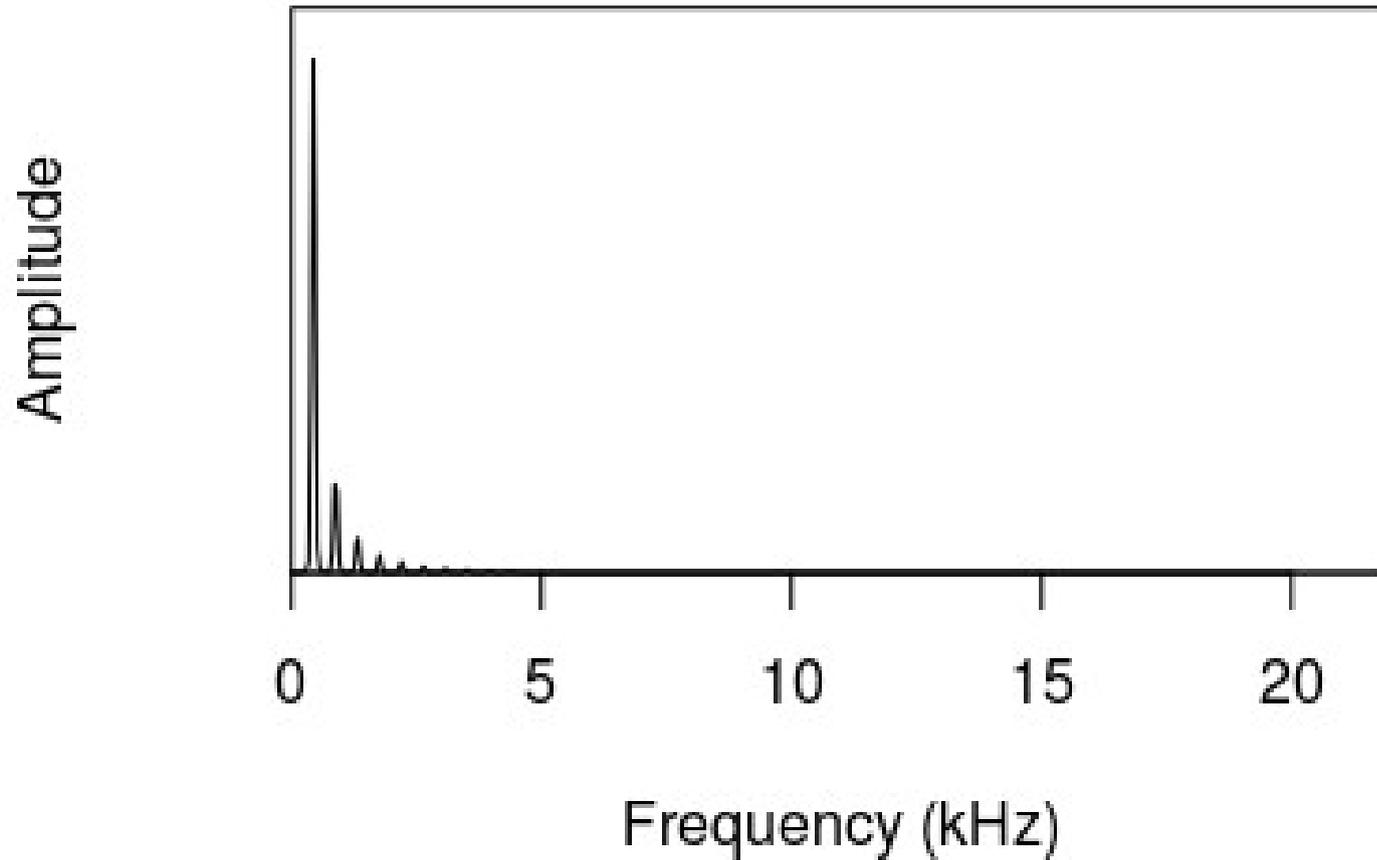
```
for (h in 1:nHarmonics){
```

```
    ampl = ampl + sin(2*pi*h*integr) * h^(-rolloff/10)
```

```
}
```

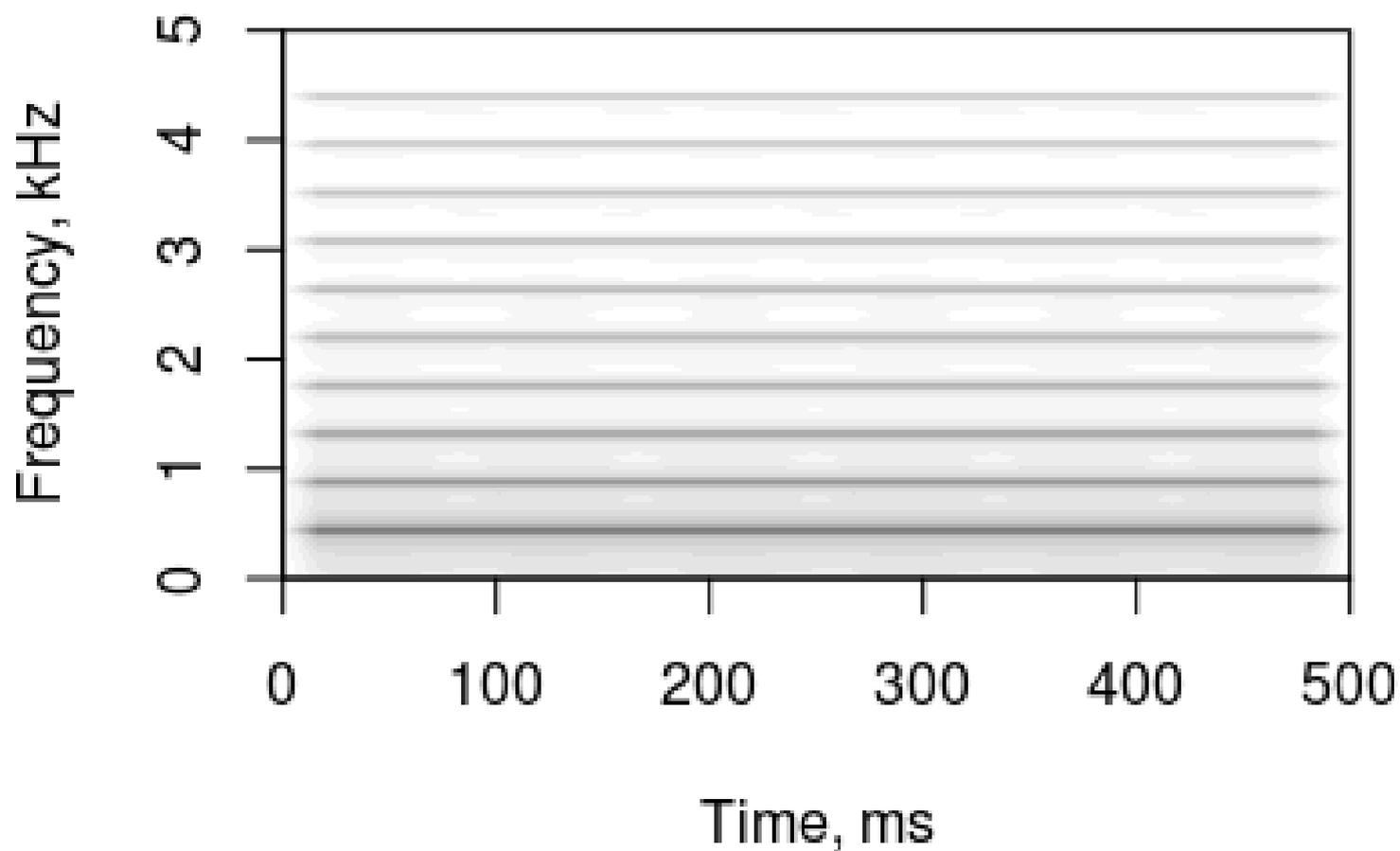
Add harmonics

440 Hz, spectrum



Add harmonics

440 Hz, spectrogram



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)

spectrum



X

filter



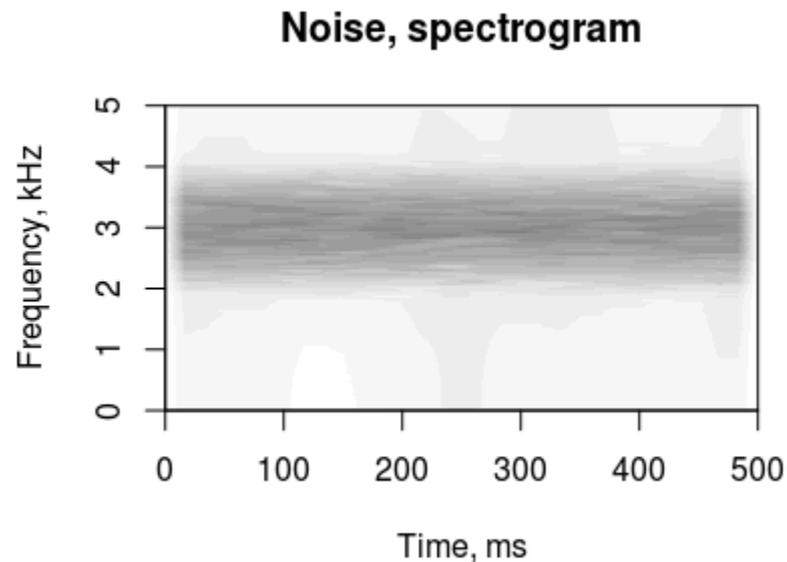
=

new spectrum



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

Part 1: sound generation

Part 2: interactive UI

Shiny R

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



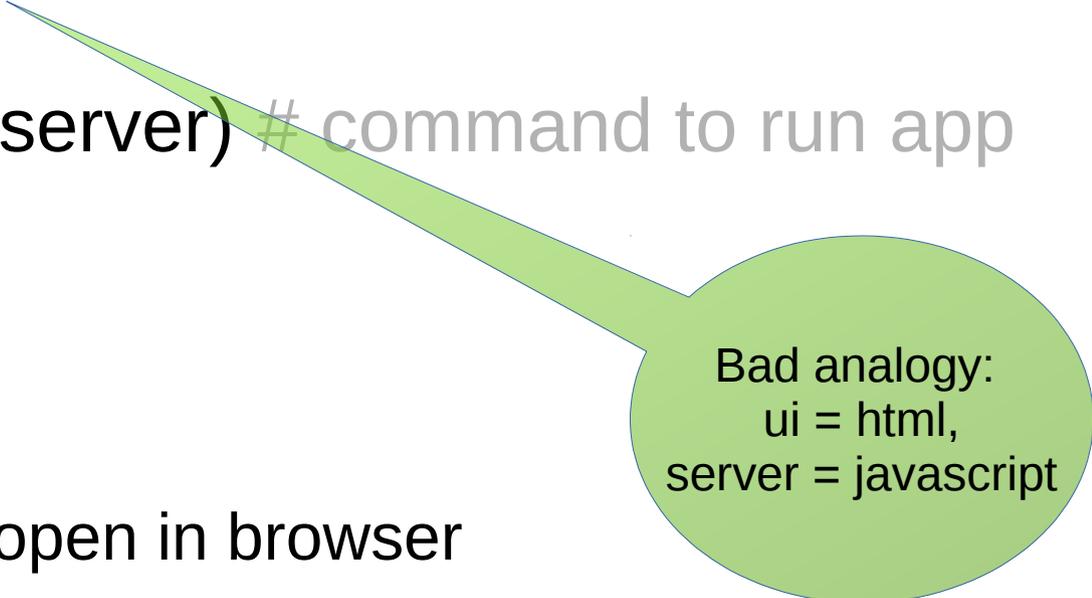
Shiny app: basic structure

```
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

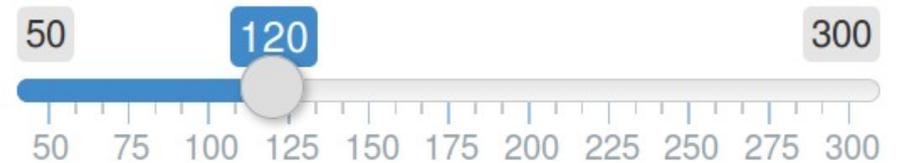


Bad analogy:
ui = html,
server = javascript

Shiny app: minimal example

```
ui <- fluidPage(  
  sliderInput('pitch_start', 'Pitch  
at start, Hz', value=120, min=50,  
max=300, step=25),  
  uiOutput("myPar")  
)
```

Pitch at start, Hz



You have set pitch to 120 Hz

```
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

The screenshot shows the SoundGen 3.0 Shiny app interface. The top navigation bar includes tabs for Intonation, Temporal, Effects, and Advanced. The main content area is divided into a sidebar on the left and a main panel on the right. The sidebar contains four sliders: Pitch at start, Hz (set to 200), Pitch at anchor, Hz (set to 400), Pitch at end, Hz (set to 300), and Anchor location, % (set to 15). The main panel features a Generate button, a progress bar, a Speaker selection (M1, F1), a Presets dropdown (set to Roar), Save wav, and Save pars buttons. A Spectrogram plot shows Frequency (kHz) on the y-axis (0 to 10) and Time (ms) on the x-axis (0 to 1500). A Pitch contour plot shows Pitch (Hz) on the y-axis (0 to 1000) and Time (ms) on the x-axis (0 to 1500). A Show spectrogram controls button is located below the spectrogram.

Callouts identify the following UI components:

- tabsetPanel
- headerPanel
- navbarMenu
- actionButton
- uiOutput
- radioButtons
- selectInput
- downloadButton
- sliderInput
- plotOutput
- HTML
- bsCollapsePanel

SoundGen 3.0, Oct 2016. Get [source code](#). Contact me at [andrey.anikin / at / lucs.lu.se](mailto:andrey.anikin@lucs.lu.se). Thank you!

sidebarLayout(
sidebarPanel(...

sliderInput

plotOutput

HTML

bsCollapsePanel

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/>

FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
\$0 /month	\$9 /month (or \$100/year)	\$39 /month (or \$440/year)	\$99 /month (or \$1,100/year)	\$299 /month (or \$3,300/year)
New to Shiny? Deploy your applications for FREE.	More applications. More active hours!	Take your users to the next level!	Password protection? Authenticate your users!	Professional has it all! Personalize your domains.
5 Applications	25 Applications	Unlimited Applications	Unlimited Applications	Unlimited Applications
25 Active Hours	100 Active Hours	500 Active Hours	2,000 Active Hours	10,000 Active Hours
✓ Community Support	✓ Premium Support	✓ Performance Boost	✓ Authentication	✓ Authentication
📄 RStudio Branding		✓ Premium Support	✓ Performance Boost	✓ Account Sharing
			✓ Premium Support	✓ Performance Boost
				✓ Custom Domains
				✓ Premium Support

Thank you!

Project web page:

http://cogsci.se/practice/sound_synthesis/scream.html