


# Using Deep Learning in R to Generate Offensive License Plates

*An introduction to neural networks!*



Jacqueline Nolis  
Principal Data Scientist, Nolis, LLC  
 @skyetetra



For years I had been JEALOUS 🙄

## Neural network-generated Coachella band names don't get any better than 'Bustles Muckson'

*So excited to see my favorite band, 'Billions of Mario'*

By [Dami Lee](#) | [@dami\\_lee](#) | Jan 23, 2018, 3:46pm EST

BOTNIK STUDIOS PRESENTS IN INDIO

# COACHELLA

COACHELLA VALLEY MUSIC AND ARTS FESTIVAL

INDIO CALIFORNIA  EMPIRE POLO CLUB

**Fanch** FRIDAY APRIL 13 & 20

**Hoop of Gomb · Lab Raid · Jacked Like A Man · Giraffics · Ben Sex**  
**Hissing in the Suite · Fistopia · Slaw Bomb · Bing the Bung · Ow · House of the Gavins · Big Big Sting**  
**Glopoline · Murf of Style · Empt-Ti · Goody Fyne · Bark · Sagging Lack · Psychic Gloria · Bouse Glous · Boys Harden**  
**Giant N · I/Me/Blood · Belt · Furious Band · Don't Kiss · John Party x4 · Mother Acid · Jonathan Is High · Hoftie Baghdood**  
**Math Moon · Barry Sky People · Cumpo · Coma On Hay · Drunk June · In the Noise of Electric City · Matt Rock Danger · Jonathan Jazz**  
**Bassbench · Dave Dump McDan · Labratross · Say Kids · Jonathan Mushboy · Lard House · Matt Drummer · Starque · Belly Sisters**  
**Girls of Parks · Lisanasia · Savid Bowry · Postwolves · Nello Buthrott · Man Mist**



Neural net by Botnik Studios

Deep learning seemed insurmountable



?

?

?

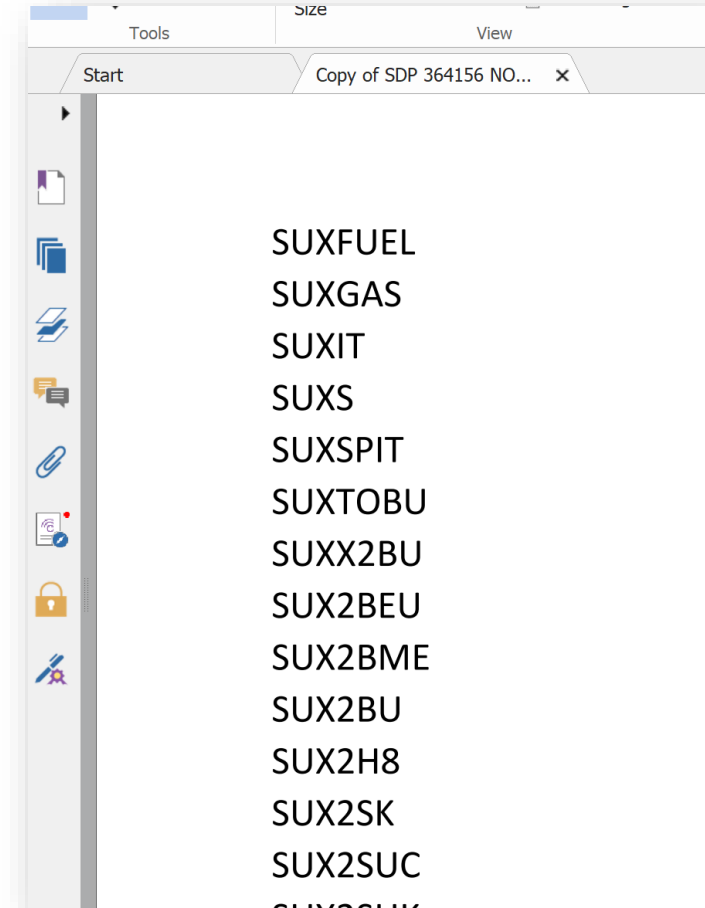
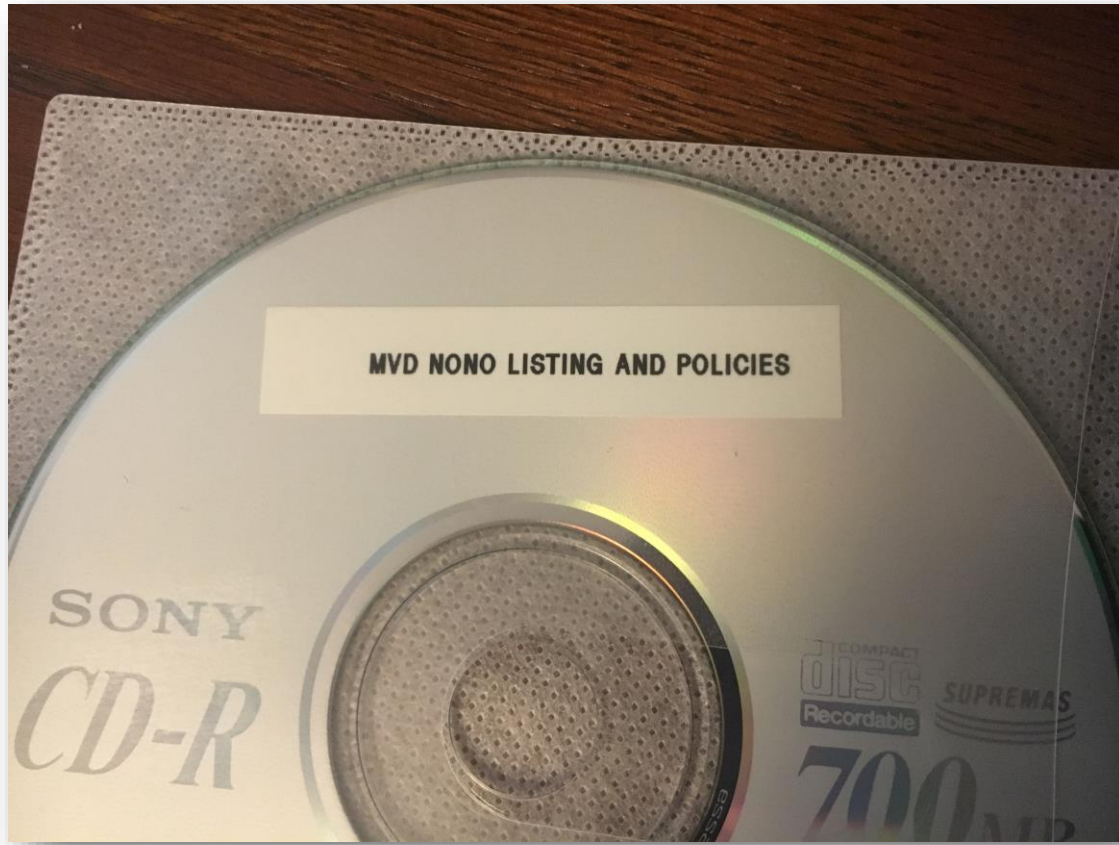
?

?



**Arizona's List of Rejected Custom License Plates Will Make You Lose Faith in Humanity**

# I DEMANDED the data (FOIA request)



The goal:

*an R function that makes new offensive plates*

```
new_funny_plate()
```

```
> "UPUNK"
```

# What is a linear regression?

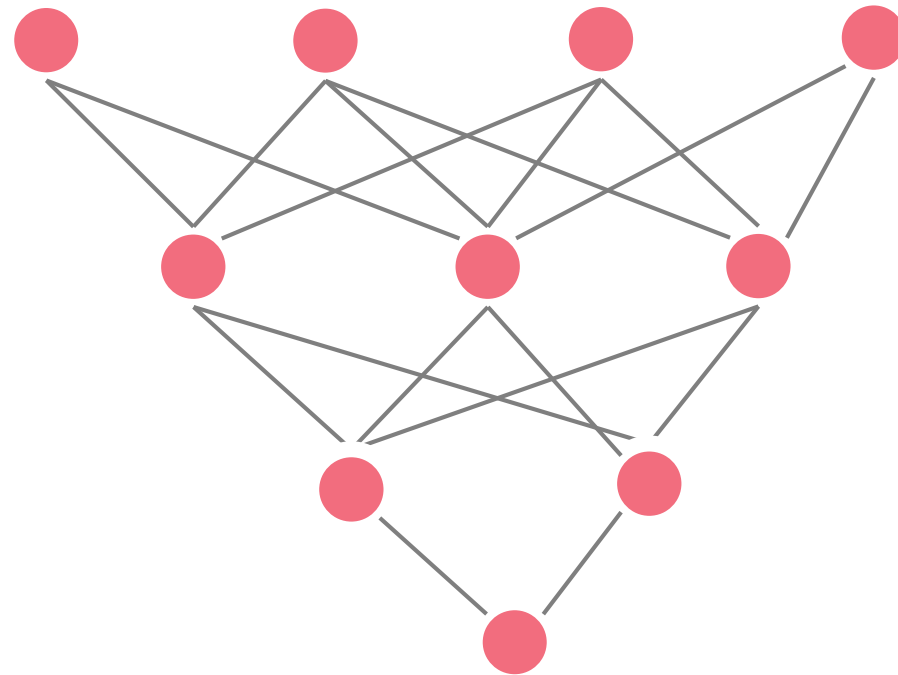
$$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$



Linear regression  $y \sim X$

# What is deep learning?

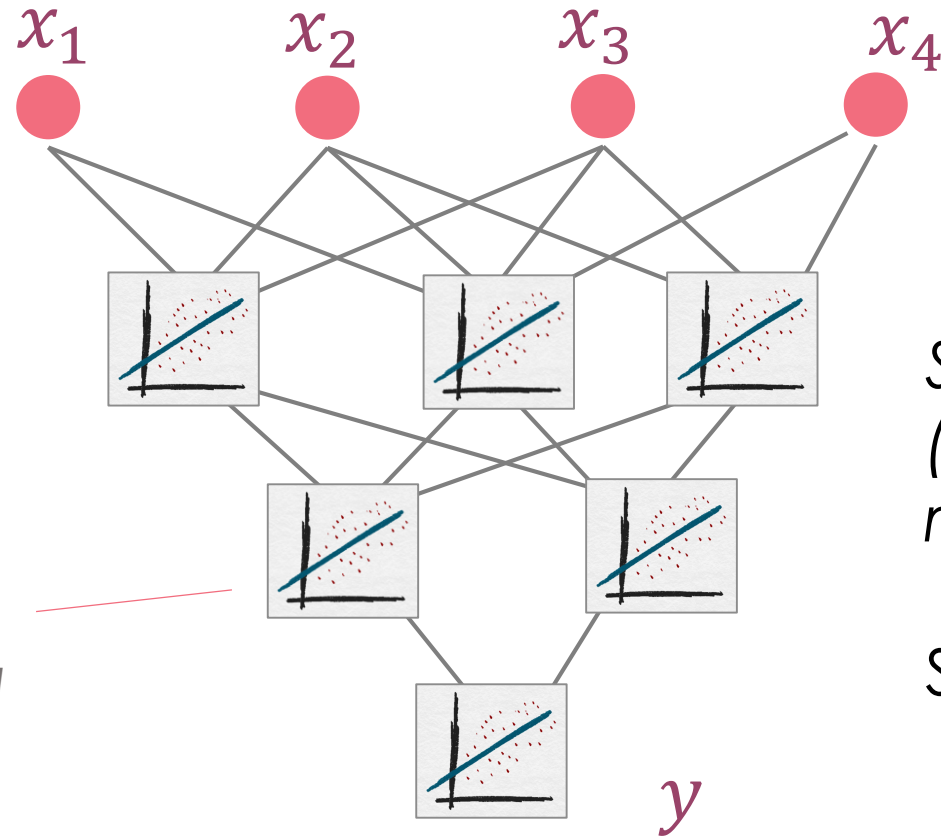
“Huh?”





# What is deep learning?

Deep learning is *powerful*



The rows of linear regressions called "layers"

String a bunch of (basically) linear regressions together!

Still  $y \sim X$

# Now deep learning is in R!

Keras library by RStudio

- Modern “good stuff”
- Simple tidy syntax
- “Secretly” runs Python and TensorFlow on the backend
- Doesn’t require an expensive computer (!!)



# Format the data

- Want to predict the next letter based on previous letters
- Training data for name “FUN1”:

X_1	X_2	X_3	X_4	X_5	Y
(blank)	(blank)	(blank)	(blank)	(blank)	F
(blank)	(blank)	(blank)	(blank)	F	U
(blank)	(blank)	(blank)	F	U	N
(blank)	(blank)	F	U	N	1
(blank)	F	U	N	1	(stop)

# Convert characters to numbers

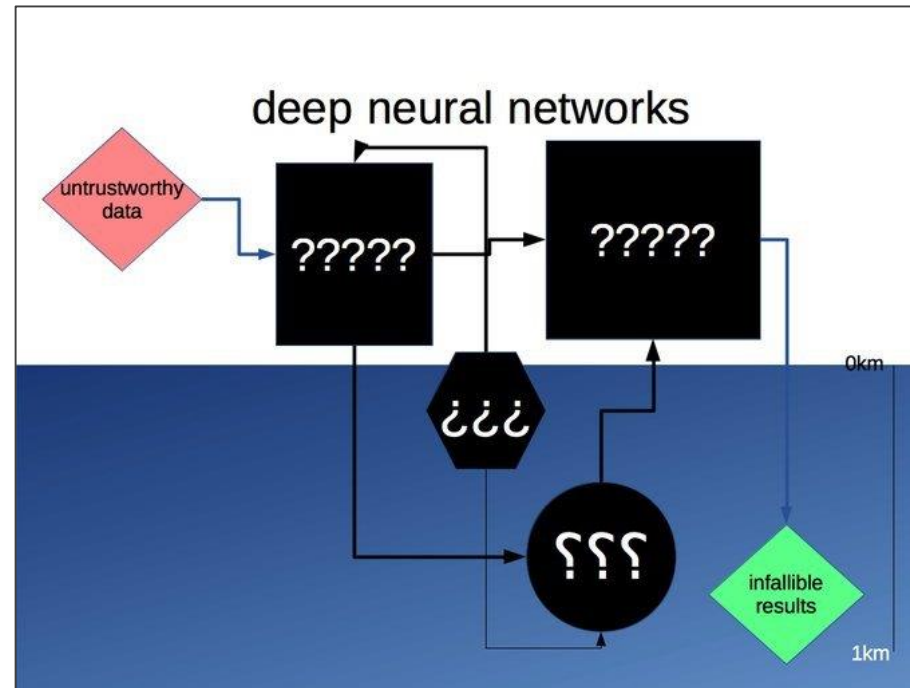
- Create a dictionary (A=1,B=2,...)
- One hot encode each number (3 = [0,0,1,0,...,0])

\_\_FUN  $\rightarrow$  [0,0,19,16,15]  $\rightarrow$  [...,[0,0,...,1,...,0], ...]

Input X is a 3-dimensional binary matrix! Size: (num\_rows, max\_length, num\_chars)

Target y is a 2-dimensional binary matrix! Size: (num\_rows, num\_chars)

Once we've formatted the data,  
make the neural network!

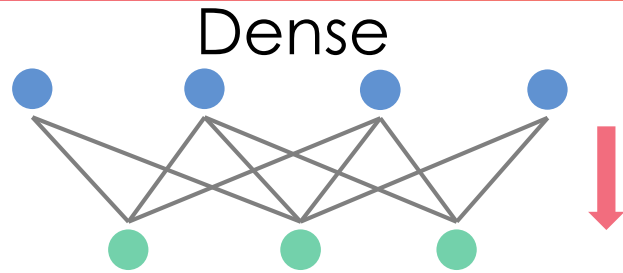


But how do we design the network?

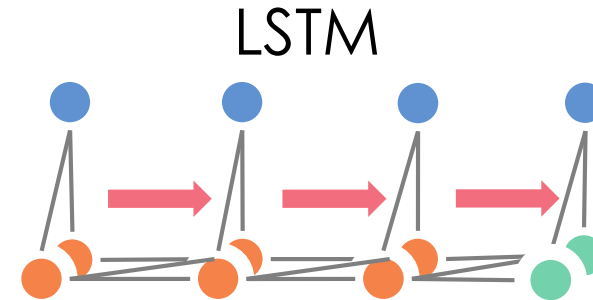
# Stealing!!!!!!

(use other people's network designs)

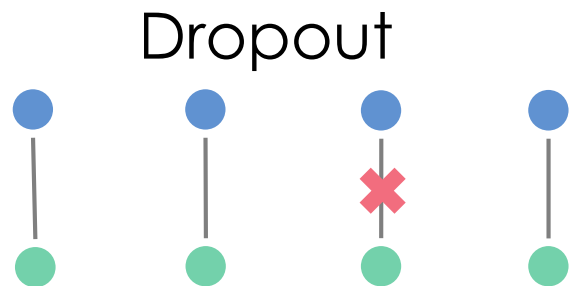
# Network layers



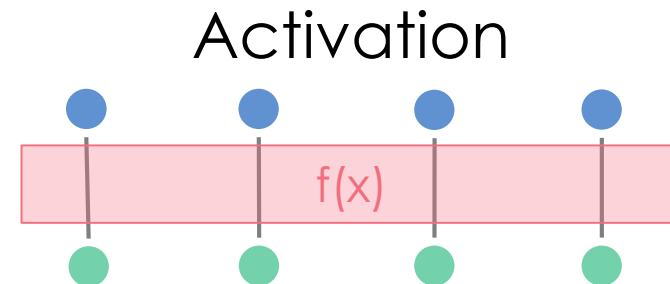
All inputs are fed into each regression



Each input fed into one set of outputs, along with previous output



Remove random values to avoid overfitting



Apply a function across the values

# Our simple network

- Input data
- LSTM – figure out the patterns
- Dense – get one output for each letter
- Activation – make scores add up to 1



*Stolen from RStudio Keras examples!*



# The code!

```
input <- layer_input(shape = c(max_length,num_characters))
```

**Set the input shape**  
*based on data*

```
output <-  
  input %>%  
  layer_lstm(units = 128) %>%  
  layer_dense(num_characters) %>%  
  layer_activation("softmax")
```

**Define the network**  
*(each layer is a line of code!)*

```
model <- keras_model(inputs = input, outputs = output) %>%  
  compile(  
    loss = 'categorical_crossentropy',  
    optimizer = "adam"  
  )
```

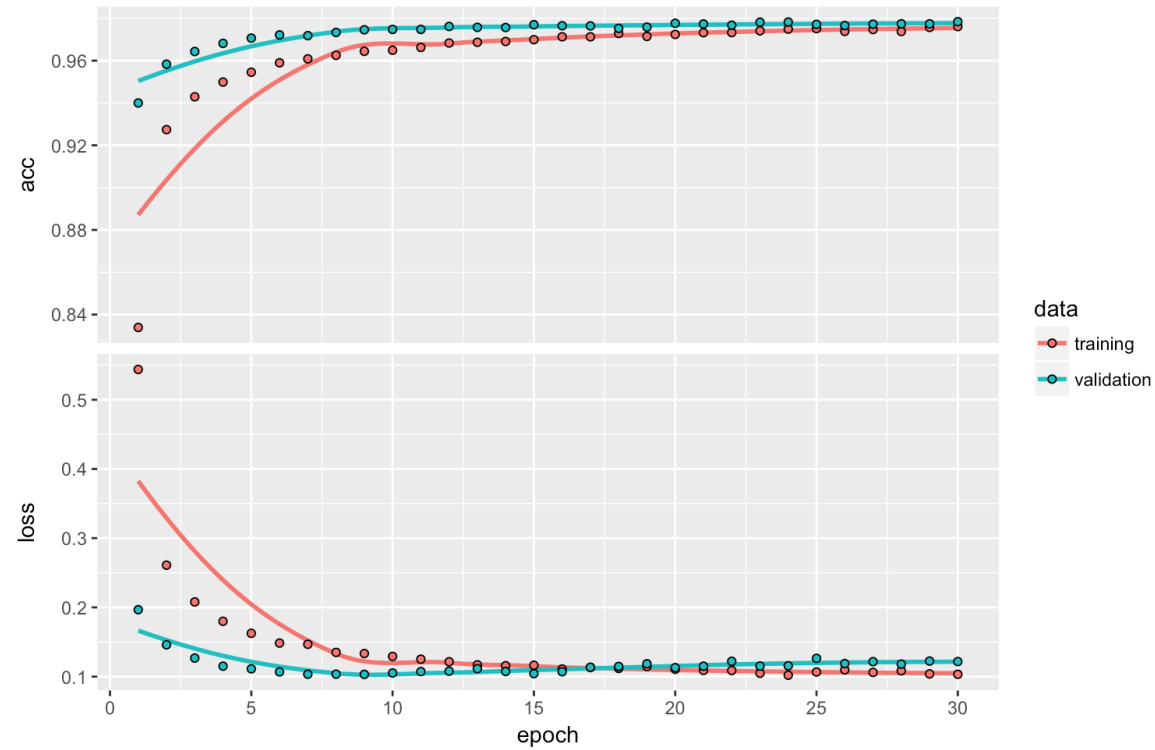
**Choose how to optimize it**  
*Choose loss based on output variable  
Setting optimizer to "adam" usually fine*

# Train the neural network!

- Choices:
  - Epochs - How many times to we want to fit data?
  - Batch Size - How many rows of data do we want to fit at once?
- Answers:
  - # epochs – Until it seems like it converges
  - Batch size – Doesn't super matter for small projects

```
fit_results <-  
  model %>%  
  keras::fit(  
    X,           Input X  
    y,           Target y  
    epochs = 25,  
    batch_size = 64  
  )
```

# This might take a while...



But RStudio gives you cool plots for your progress



# You're done! Save your work!

```
save_model_hdf5(model, "model.h5")
```



# How to use the model?

- Like everything else in R, use predict:  
`predict(model,previous_letters_data)`
- **For input letters, returns the probability of each letter being next**
- Define a function that:
  - Starts with a blank string
  - Predicts the next letter & updates
  - Predicts the next letter & updates (Keep going!)
  - Stop character! Stop!!!

## Input: E X A M

Next Letter	Probability
A	0.1
B	0.02
...	...
P	0.5
...	...
Z	0.001
(stop)	0.2

It works!!



# Inspiring others



Ryan Timpe  
@ryantimpe



Julia Silge  
@juliasilge

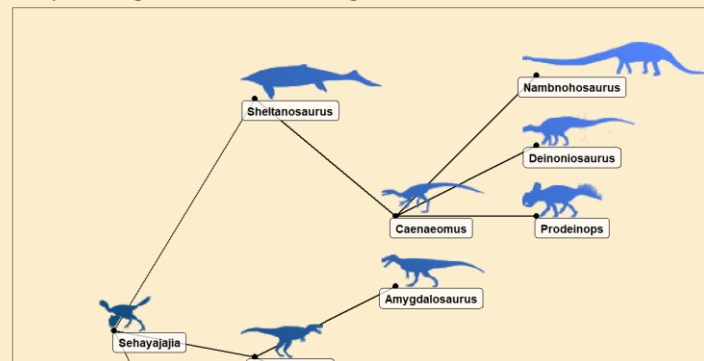


Namita Nandakumar  
@nnstats

## Dinosaur species

### Sehayajaja phylogenetic tree

Deep learning dinosaur names using Keras



## Jane Austen texts

the happiness of her sisters and had heard of the earnest of the servance of the consequence of the family of the very little to her friend to her family of the lady of the moment to the persuaded her to her sisters were allow to

## House Targaryen names

Aelora	Aelyx	Aerysa
Daeger	Daella	Daemion
Daenyra	Daeron	Dregon
Jaegor	Maegon	Maegys
Rhaegel	Rhaegon	Rhaenor
Vaella	Vaelon	Valera

# Conclusion: Do whatever motivates you!

*Any project that you learn from is worthwhile*



Learning this...



...taught me a foundation of what we use on major clients



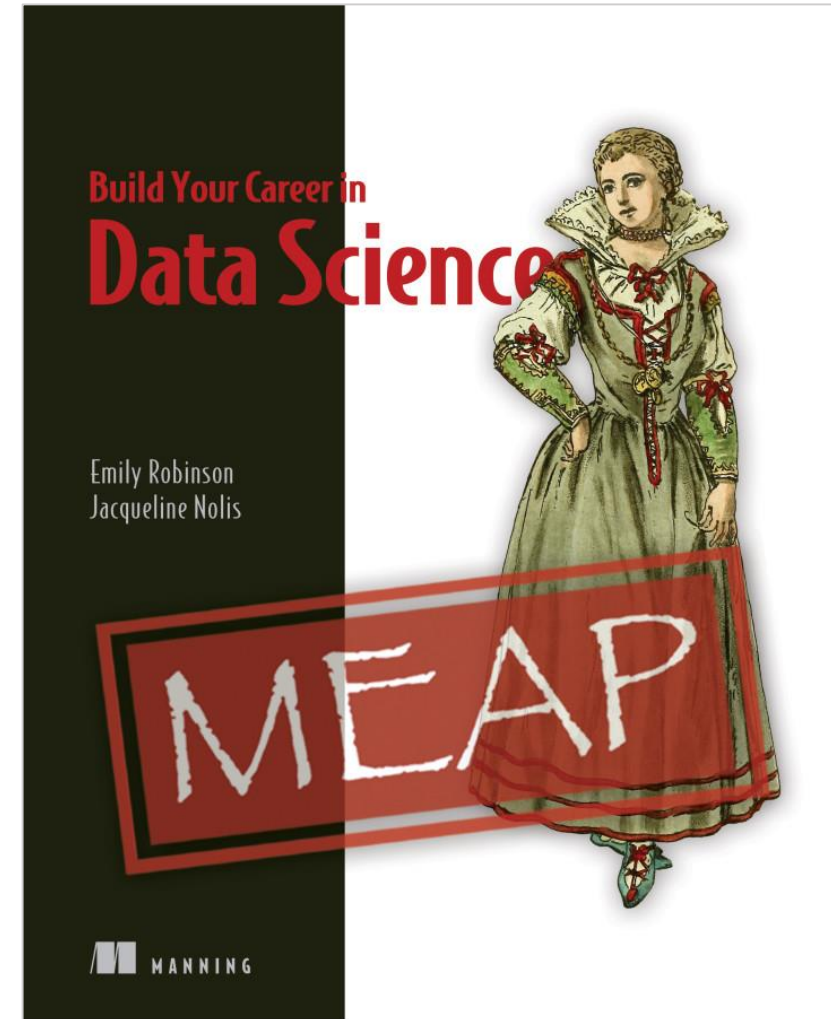


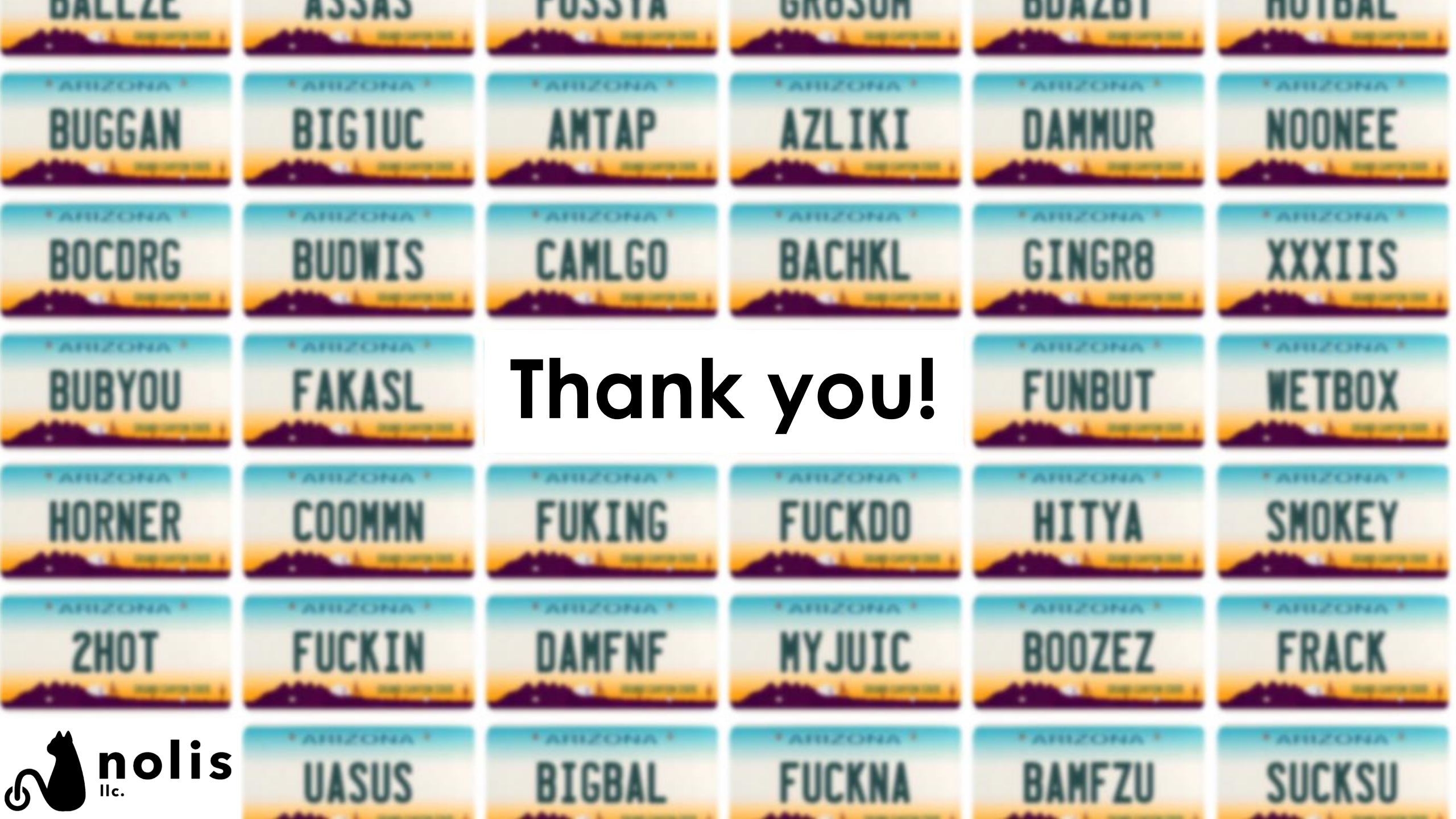
# Wrap'n it up

- Neural networks in R are not hard
- Can be treated like a super-duper linear regression
- Look online for how other people design them
- Run on your laptop to start!

# Calls to action

- View the code at:  
[github.com/jnolis/banned-license-plates](https://github.com/jnolis/banned-license-plates)
- View the slides at:  
<http://bit.ly/sdss-offensive-plates>
- Follow me on twitter [@skyetetra](https://twitter.com/skyetetra)
- Go to [nolisllc.com](https://nolisllc.com) for *professional consulting services*
- Check out me and Emily Robinson's book:  
Build Your Career In Data Science  
[bit.ly/datascibk](https://bit.ly/datascibk)





**Thank you!**