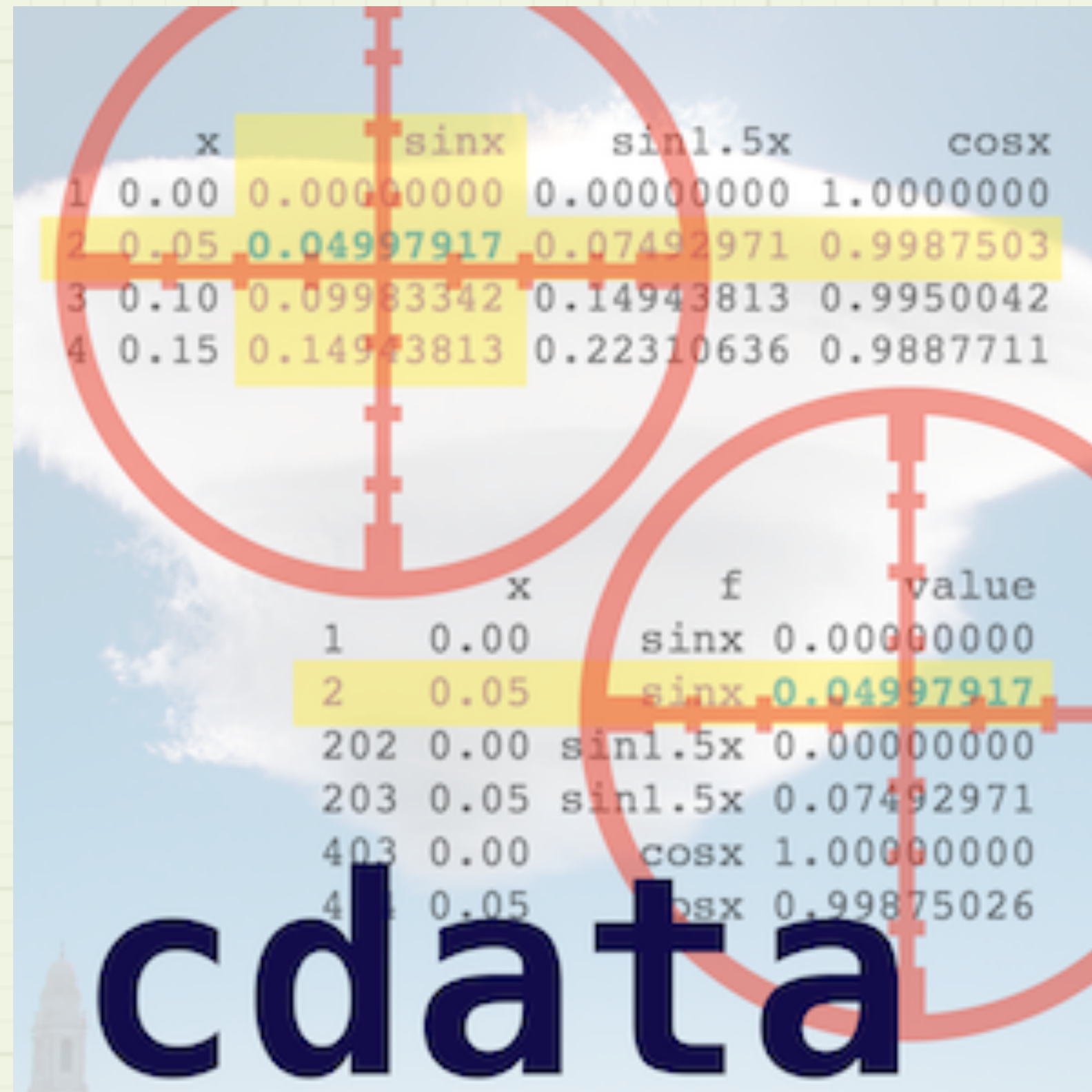


# cdata

## Fluid Data Transforms at Scale in R



John Mount  
Nina Zumel

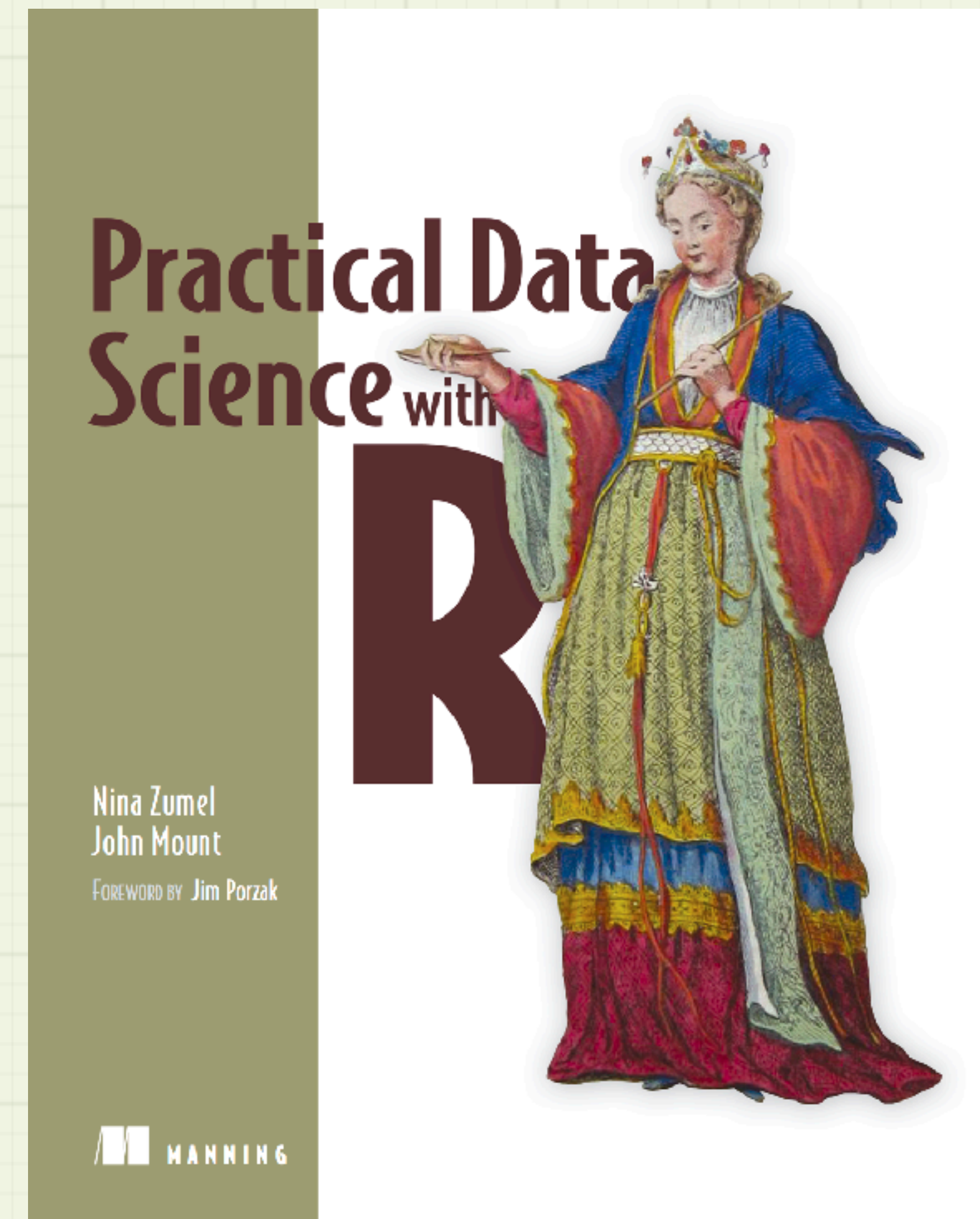


These slides:

<https://github.com/WinVector/cdata/blob/master/extras/cdata.pdf>

# Who Am I?

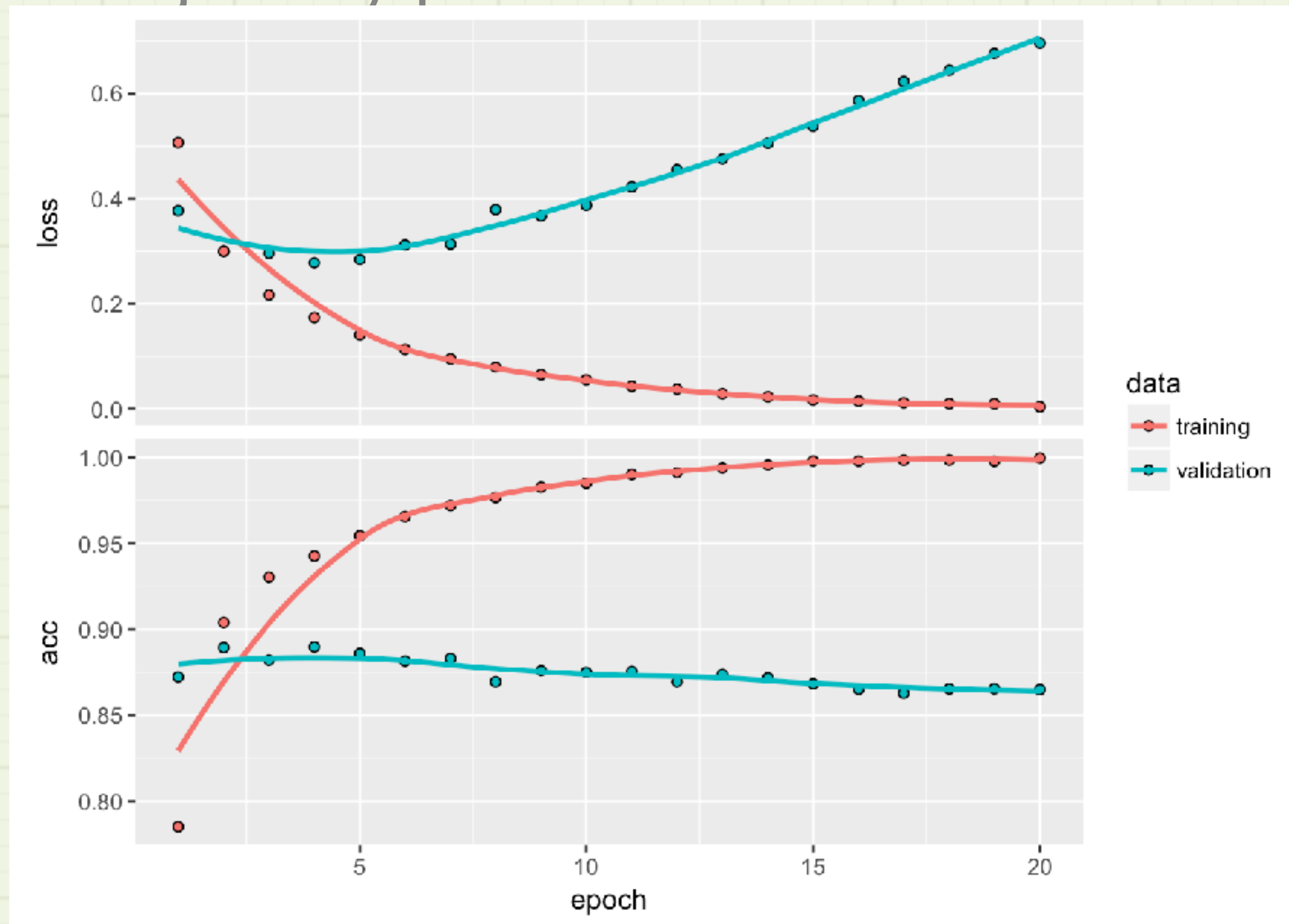
- John Mount
- Principal Consultant and Win-Vector LLC (data science consulting and training).
- One of the authors of *Practical Data Science with R* (Manning 2014).
- Co-author of a number of R packages
  - **vtreat**: statistically sound advanced data preparation
  - **wrapr**: sweet code tools for R
  - **cdata**: fluid data transforms at scale
  - (in development) **rquery**: reliable big data operators (piped **SQL**).
  - and many more
- Contributor to the [Win-Vector Blog](#), read by data scientists worldwide
- Frequent speaker on algorithms, statistics, machine learning and data science topics



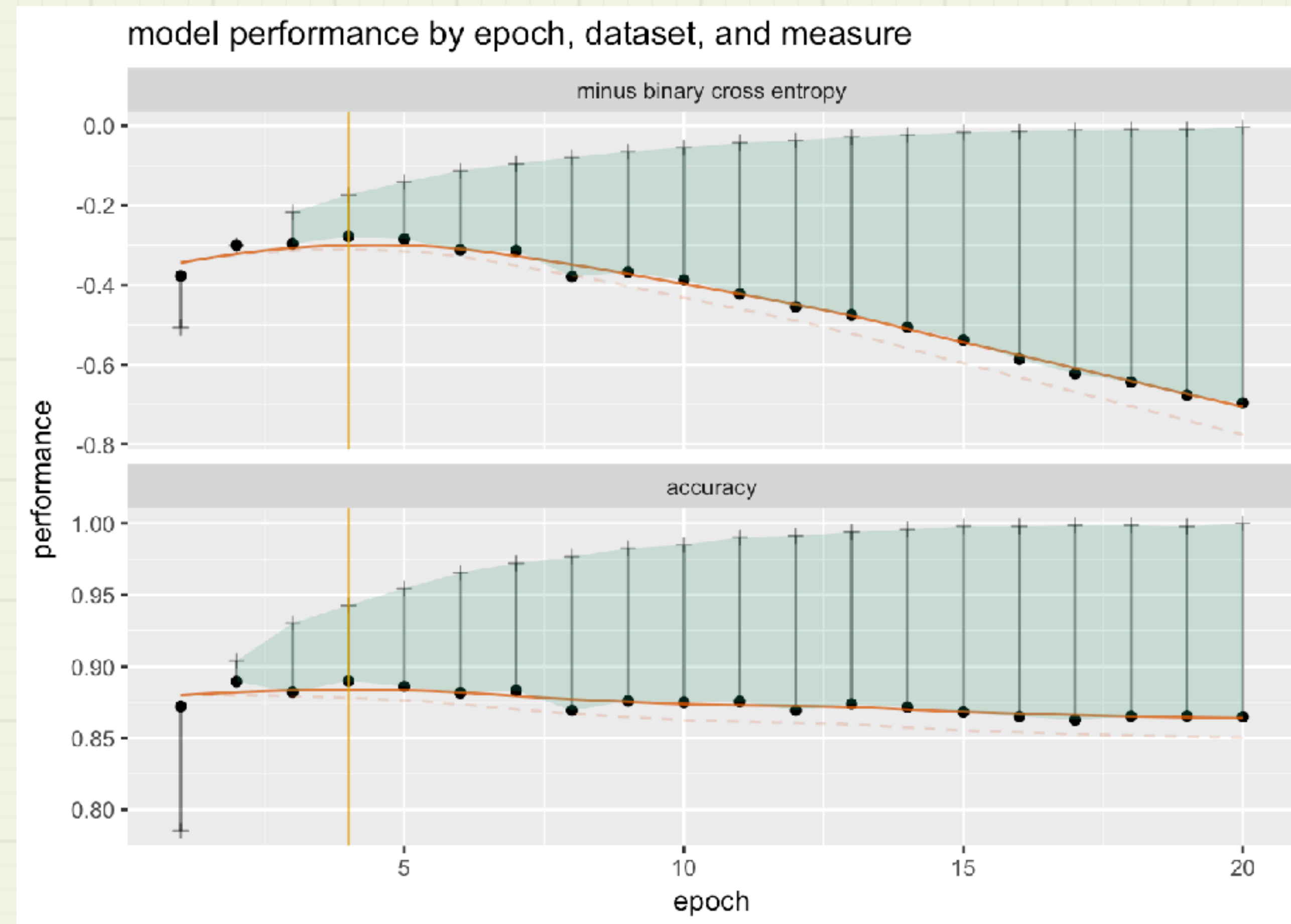


# Our Example Problem

Replace this **Keras** deep learning fit trajectory plot



With this plot

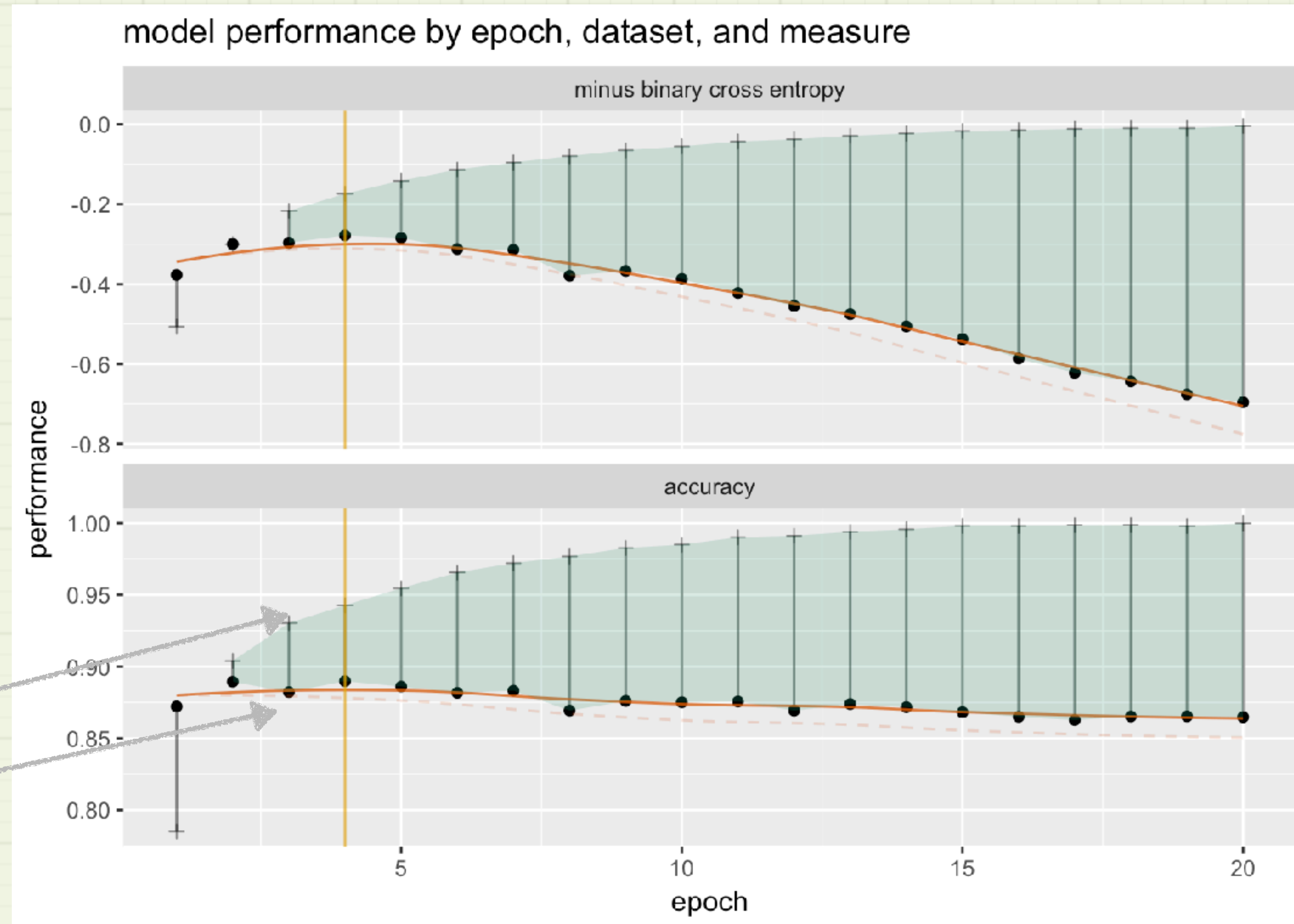


# Advantages of The New Presentation

- Up means better in all facets.
- Validation performance is rendered as the *only* horizontal curve.

Training performance

Test performance



# The issue

- Plot history data looks like the table to the right.
- **ggplot::geom\_ribbon()** needs training and validation results in same row.
- **ggplot::facet\_wrap()** needs different measures in different rows.

val_loss	val_acc	loss	acc	epoch
-0.3769818	0.8722	-0.5067290	0.7852000	1
-0.2996994	0.8895	-0.3002033	0.9040000	2
-0.2963943	0.8822	-0.2165675	0.9303333	3
-0.2779052	0.8899	-0.1738829	0.9428000	4
-0.2842501	0.8861	-0.1410933	0.9545333	5
-0.3119754	0.8817	-0.1135626	0.9656000	6

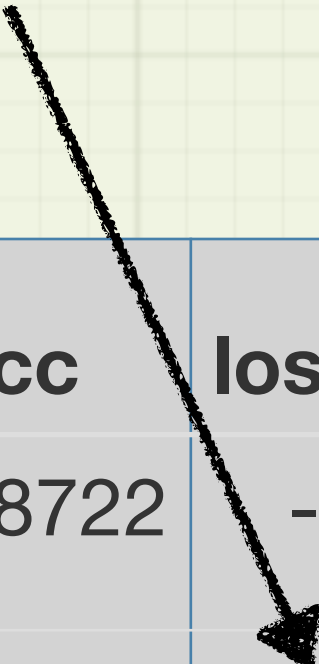
# We Need to re-Shape the Data

- Could do this with some combination of gather, cbind, rbind, and/or join.
- We are going to show a crystal clear "all in one step" tool for this: cdata (available on **CRAN**).
- This presentation will emphasize diagrammatic thinking and arguments.

# The Concept

- Data has **coordinates**.
- Information is grouped in **records**.
- Exact realization in a given table is an inessential implementation detail.

This cell is "the epoch 2 training loss", independent of how the table is formatted.



val_loss	val_acc	loss	acc	epoch
-0.3769818	0.8722	-0.5067290	0.7852000	1
-0.2996994	0.8895	<b>-0.3002033</b>	0.9040000	2
-0.2963943	0.8822	-0.2165675	0.9303333	3
-0.2779052	0.8899	-0.1738829	0.9428000	4
-0.2842501	0.8861	-0.1410933	0.9545333	5
-0.3119754	0.8817	-0.1135626	0.9656000	6



# Let's Look at One Row Record

Want to transform from this:

epoch	val_loss	val_acc	loss	acc
1	-0.3769818	0.8722	-0.506729	0.7852



To this:

epoch	measure	training	validation
1	minus binary cross entropy	-0.506729	-0.3769818
1	accuracy	0.785200	0.8722000



# We *Draw* The Transform

val_loss	val_acc	loss	acc
val_loss	val_acc	loss	acc



measure	training	validation
minus binary cross entropy	loss	val_loss
accuracy	acc	val_acc

epoch	val_loss	val_acc	loss	acc
1	-0.3769818	0.8722	-0.506729	0.7852



epoch	measure	training	validation
1	minus binary cross entropy	-0.506729	-0.3769818
1	accuracy	0.785200	0.8722000

# The Transforms

controlTable: cT

measure	training	validation
minus binary cross entropy	loss	val_loss
accuracy	acc	val_acc

epoch	val_loss	val_acc	loss	acc
1	-0.3769818	0.8722	-0.506729	0.7852

```
rowrecs_to_blocks(  
    .,   
    controlTable = cT,  
    columnsToCopy = "epoch")
```

```
blocks_to_rowrecs(  
    .,   
    controlTable = cT,  
    keyColumns = "epoch")
```

epoch	measure	training	validation
1	minus binary cross entropy	-0.506729	-0.3769818
1	accuracy	0.785200	0.8722000

# Control Table *is* the Transform

Control table is a picture of what the control table transforms a single row into our from (depending on direction of operator).

controlTable: cT

measure	training	validation
minus binary cross entropy	loss	val_loss
accuracy	acc	val_acc

First column behaves like `rownames()`, *except* it also carries the extra column name “measure”.

val_loss	val_acc	loss	acc
val_loss	val_acc	loss	acc

```
rowrecs_to_blocks(  
  .,  
  controlTable = cT,  
  columnsToCopy = NULL)
```

```
blocks_to_rowrecs(  
  .,  
  controlTable = cT,  
  keyColumns = NULL)
```

measure	training	validation
minus binary cross entropy	loss	val_loss
accuracy	acc	val_acc

# Back to Our Task

To reverse transform, need to specify `keyColumns` (instead of `columnsToCopy`) which identify which sets of rows go together to form blocks.

val_loss	val_acc	loss	acc	epoch
-0.3769818	0.8722	-0.5067290	0.7852000	1
-0.2996994	0.8895	-0.3002033	0.9040000	2
-0.2963943	0.8822	-0.2165675	0.9303333	3
-0.2779052	0.8899	-0.1738829	0.9428000	4
-0.2842501	0.8861	-0.1410933	0.9545333	5
-0.3119754	0.8817	-0.1135626	0.9656000	6

...



controlTable: cT

measure	training	validation
minus binary cross entropy	loss	val_loss
accuracy	acc	val_acc

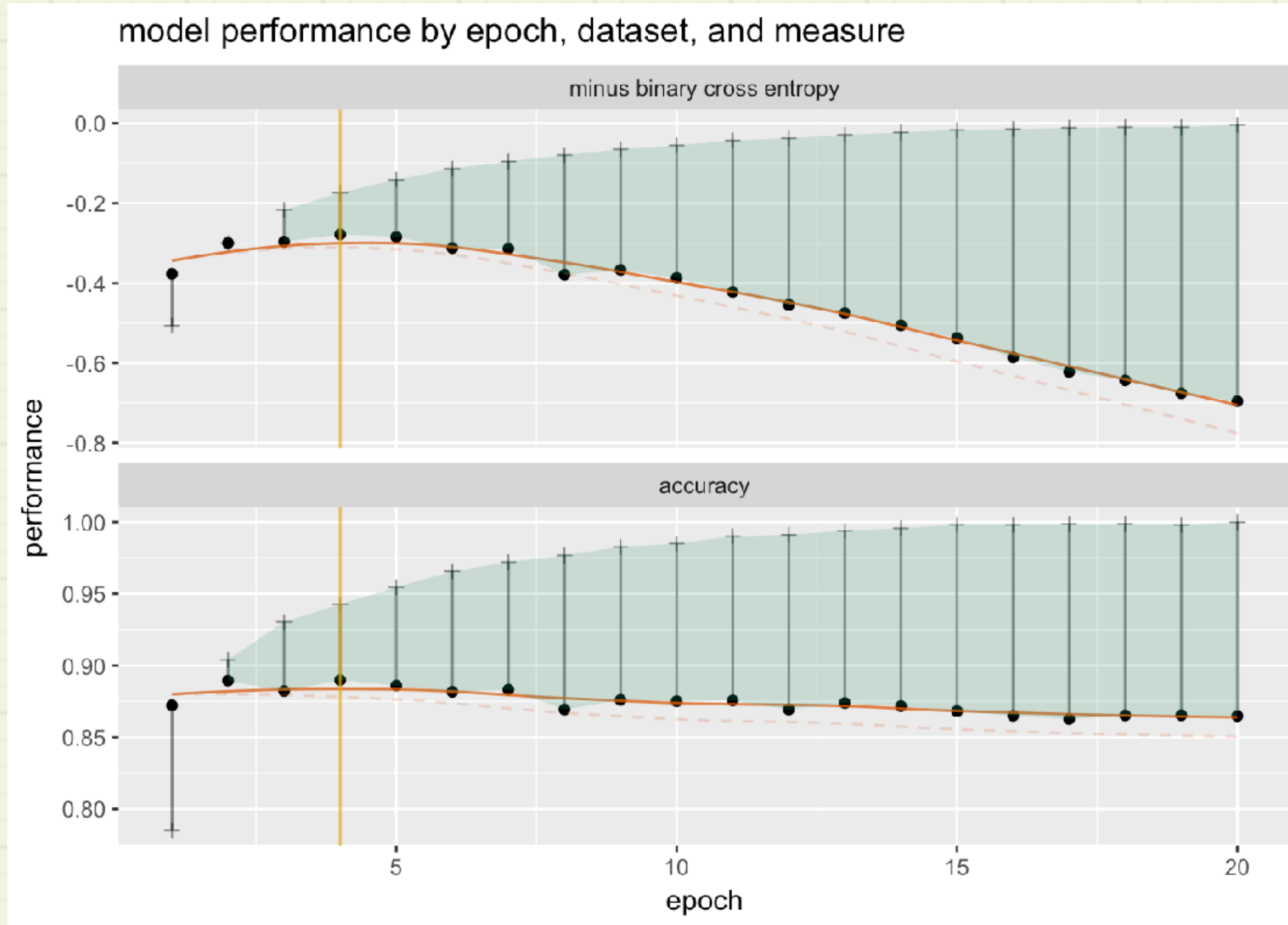
```
rowrecs_to_blocks(  
    •,  
    controlTable = cT,  
    columnsToCopy = "epoch")
```

epoch	measure	training	validation
1	minus binary cross entropy	-0.5067290	-0.3769818
1	accuracy	0.7852000	0.8722000
2	minus binary cross entropy	-0.3002033	-0.2996994
2	accuracy	0.9040000	0.8895000
3	minus binary cross entropy	-0.2165675	-0.2963943
3	accuracy	0.9303333	0.8822000

...

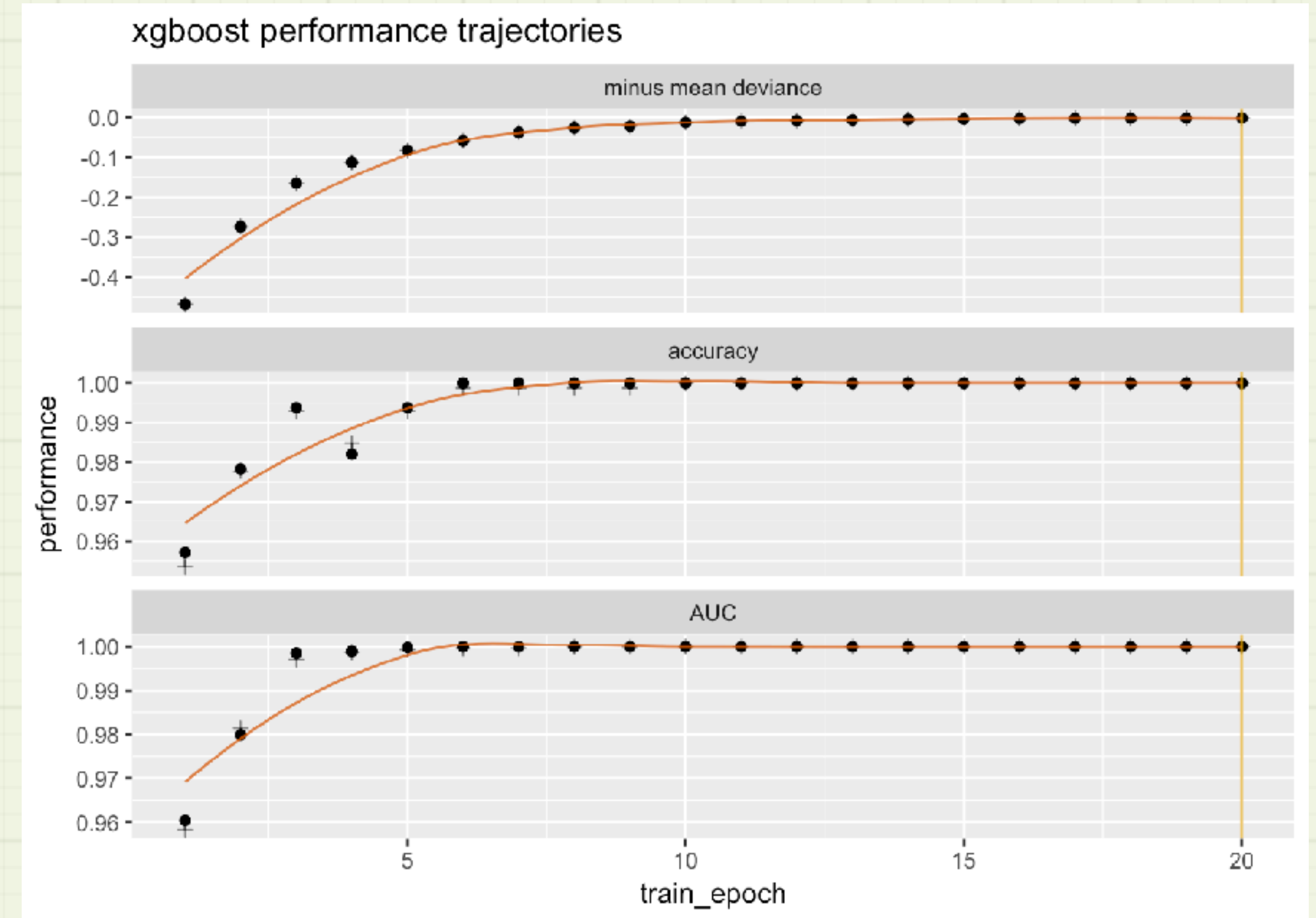


# And We are Ready to Plot



## Keras example

```
WVPlots::plot_Keras_fit_trajectory(  
  d,  
  title = "model performance by epoch, dataset, and measure")
```



## xgboost example

# Why Use cdata?

- *Clear* teachable theory:
  - Abstract data coordinates and abstract records
  - `blocks_to_rowrecs()` and `rowrecs_to_blocks()`
  - Diagrammatic theory: **draw the transform diagram!**
- *Powerful* transforms:
  - pivot/un-pivot, `tidyr::spread()`/`tidyr::gather()`, and one-hot-encode are all easy special cases.
  - Seamlessly move multiple values together.
- **cdata** is based on **SQL** and **DBI**
  - It can be *directly* applied to big data (via **Spark** or **PostgreSQL**).
  - Already have clients using this on **Spark** in production.

# Thank you!

- Links:
  - The **R** package  
<https://github.com/WinVector/cdata>
  - Fluid data reshaping with **cdata**  
<http://winvector.github.io/FluidData/FluidDataReshapingWithCdata.html>
  - Coordinatized data theory:  
<http://winvector.github.io/FluidData/RowsAndColumns.html>
- More:
  - Ready to go **Keras** plot:  
<http://winvector.github.io/FluidData/PlotExample/KerasPerfPlot.html>
  - These slides:  
<https://github.com/WinVector/cdata/blob/master/extras/cdata.pdf>
  - All the code behind this talk (with more links):  
<http://winvector.github.io/FluidData/PlotExample/PlotExample.html>