```
^{:clay {:kindly/options {:kinds-that-hide-code #{:kind/hiccup
                                                  :kind/md
                                                  :kind/html}}
         :quarto {:format {:revealjs {:theme :white
                                      :transition :none
                                      :transition-speed :fast}}

                  :fontsize "2em"
                  :mainfont "Helvetica"
                  :monofont "Roboto Mono"}
         :hide-info-line true
         :hide-ui-header true}}
(ns slides
  (:require [fastmath.random :as random]
                        [jon30.core :as core]
                        [jon30.data :as data]
                        [scicloj.cmdstan-clj.v1.api :as stan]
                        [scicloj.hanamicloth.v1.plotlycloth :as ploclo]
                        [scicloj.kindly.v4.kind :as kind]
                        [scicloj.metamorph.ml :as ml]
                        [scicloj.metamorph.ml.design-matrix :as dm]
                        [scicloj.metamorph.ml.regression]
                        [tablecloth.api :as tc]
                        [tablecloth.column.api :as tcc]
                        [tech.v3.tensor :as tensor]))
```

8-bit-sheep


Beginner since
1983 (SVI-318)

What we offer

Data solutions by
a select group of top experts

What we do

Identity solutions

Digital analytics

Data products

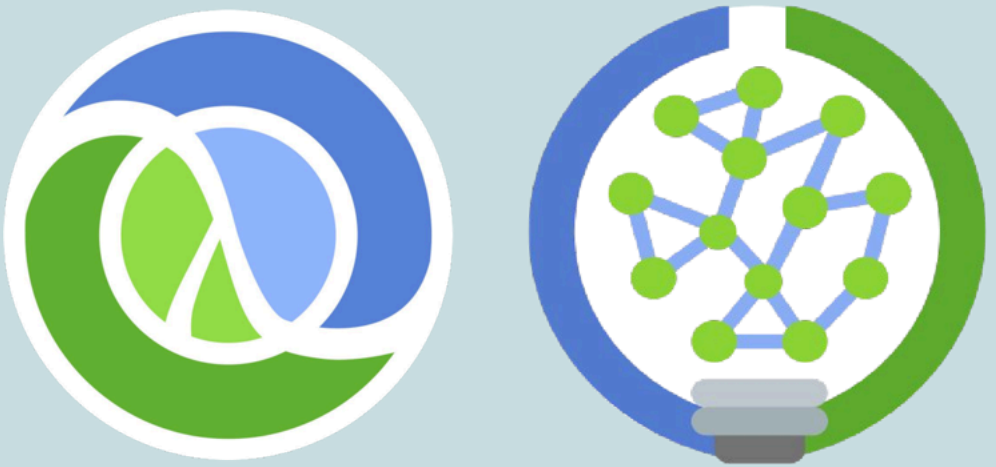
What we know

Coaching: Analytics
planning and management

Implementation: Development,
architecture analytics,
data and engineering

KP System

Geomatikk



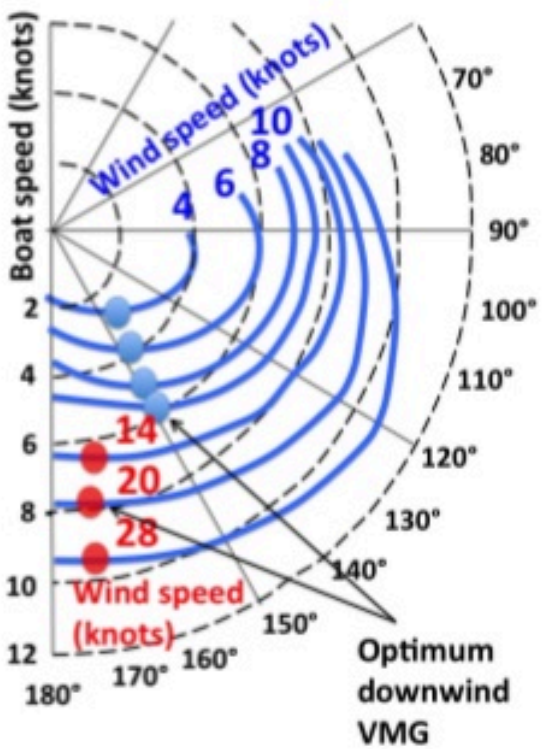
Daniel
Slutsky &
Community

The goal:

Predicting vessel velocity by wind conditions

What will we do with the results?

Polars!



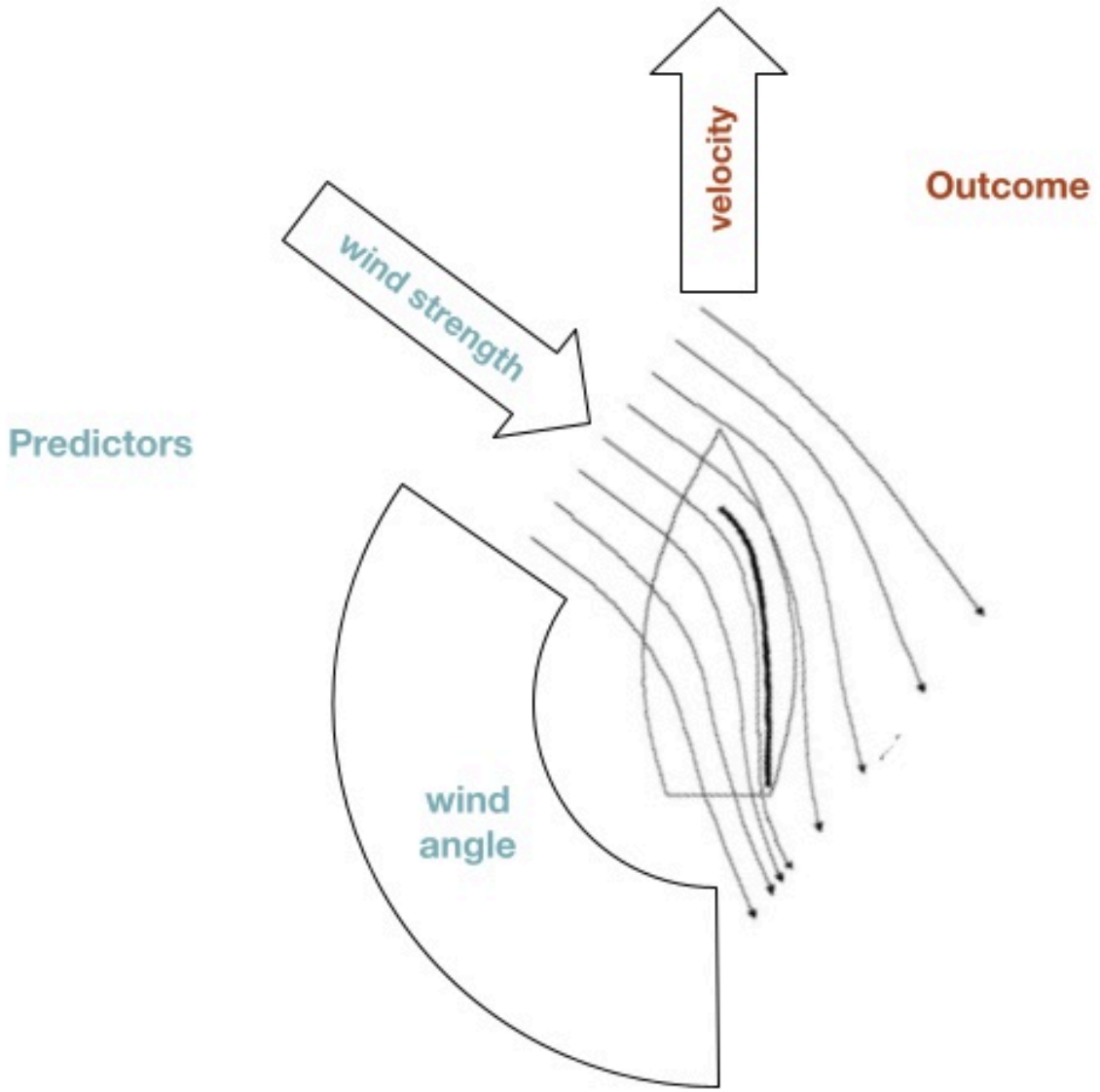
What are polar diagrams used for?

Racing focuses on optimizing performance and tactics during competitions.

Weather Routing and Planning uses polars to make long-distance cruising or offshore racing safer and more efficient.

Quantified Sailing applies data analysis to track, measure, and enhance overall performance.

Predicting velocity?




VAARALLINEN

Start simple

6 knots

6.9 mph

11 km/h

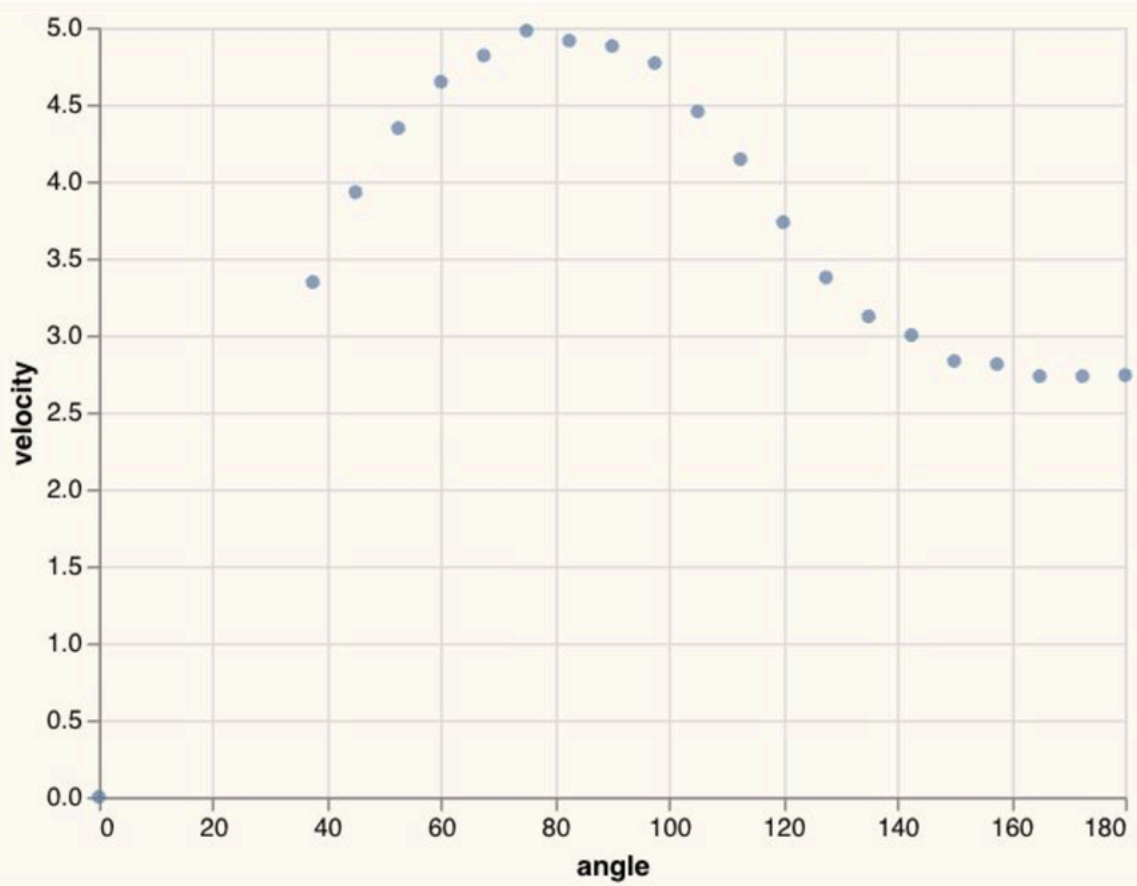
3 m/s

Synthetic data

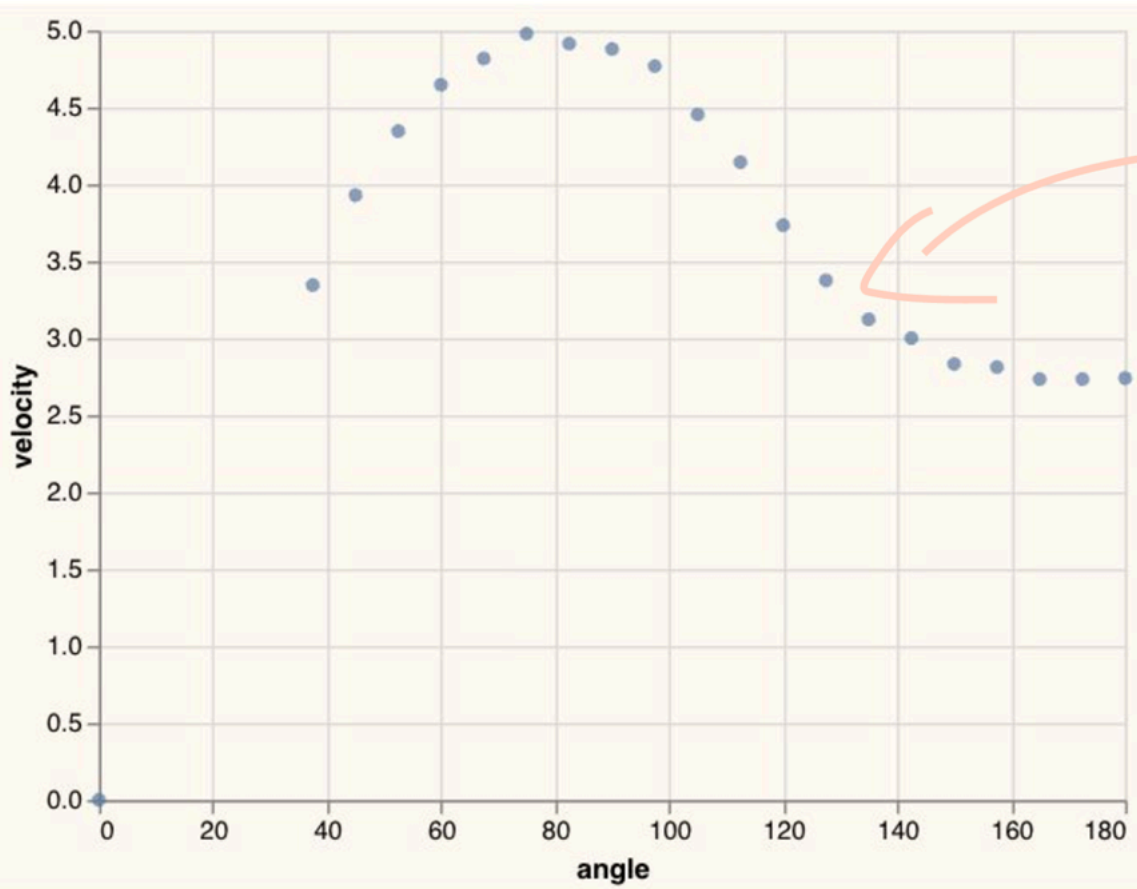
Synthetic data

:angle	:velocity	:wind
0.0	0.000	6
37.5	3.345	6
45.0	3.929	6
52.5	4.345	6
60.0	4.647	6
67.5	4.817	6
75.0	4.979	6
82.5	4.914	6
90.0	4.879	6
97.5	4.768	6
...	...	...
105.0	4.453	6
112.5	4.143	6
120.0	3.734	6
127.5	3.375	6
135.0	3.122	6
142.5	3.000	6
150.0	2.832	6
157.5	2.812	6
165.0	2.734	6
172.5	2.734	6
180.0	2.740	6

```
(-> (wind-strength 6)
    (haclo/plot haclo/point-chart
                {:=x :angle
                 :=y :velocity}))
```



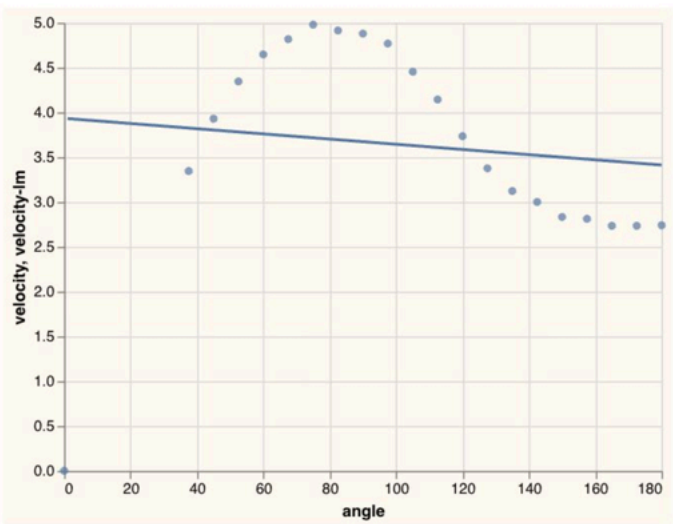
```
(-> (wind-strength 6)
    (haclo/plot haclo/point-chart
                {:=x :angle
                 :=y :velocity}))
```



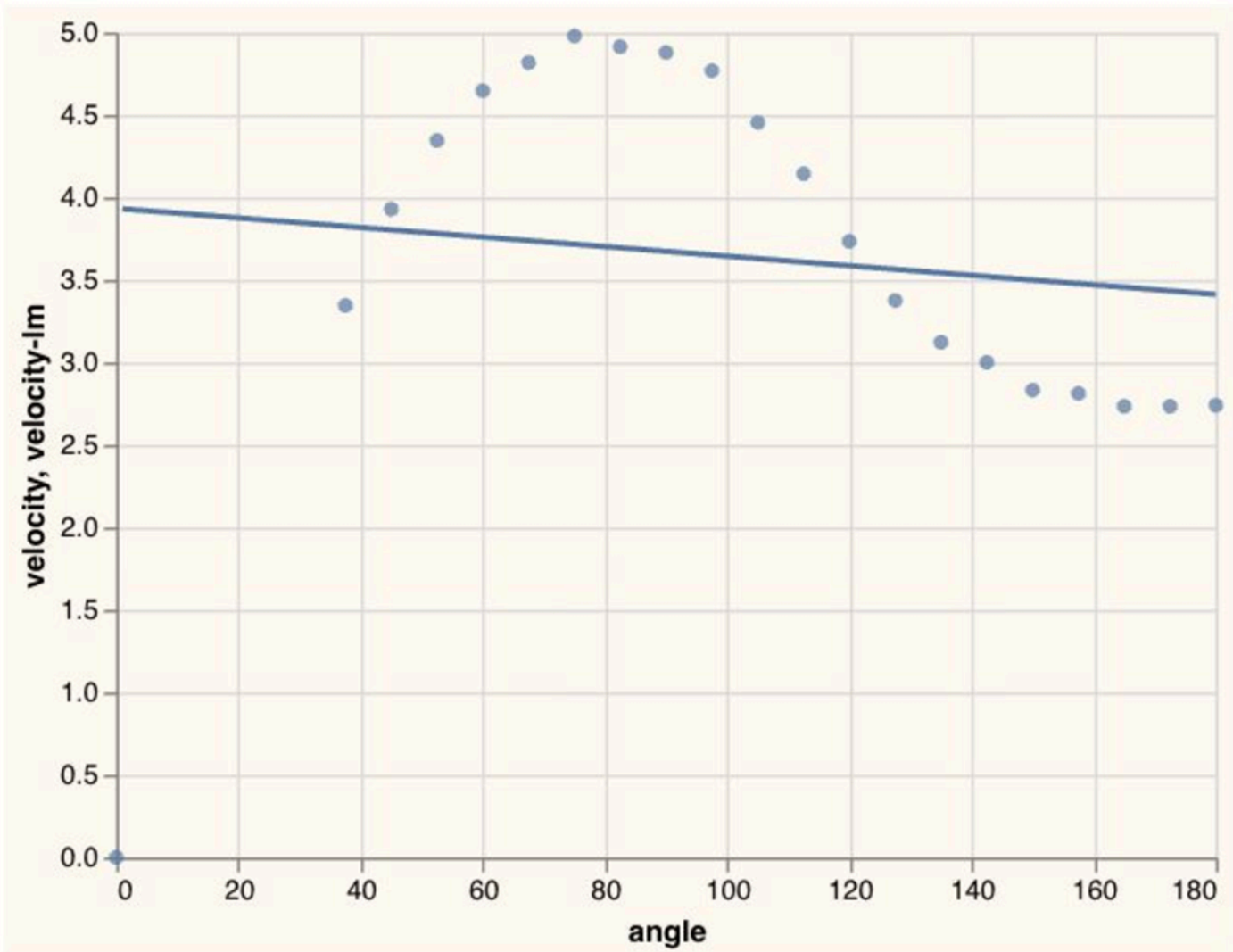
Regression

$$\text{Velocity} = \alpha_0 + \alpha_1 * \text{Angle} + \text{Noise}$$

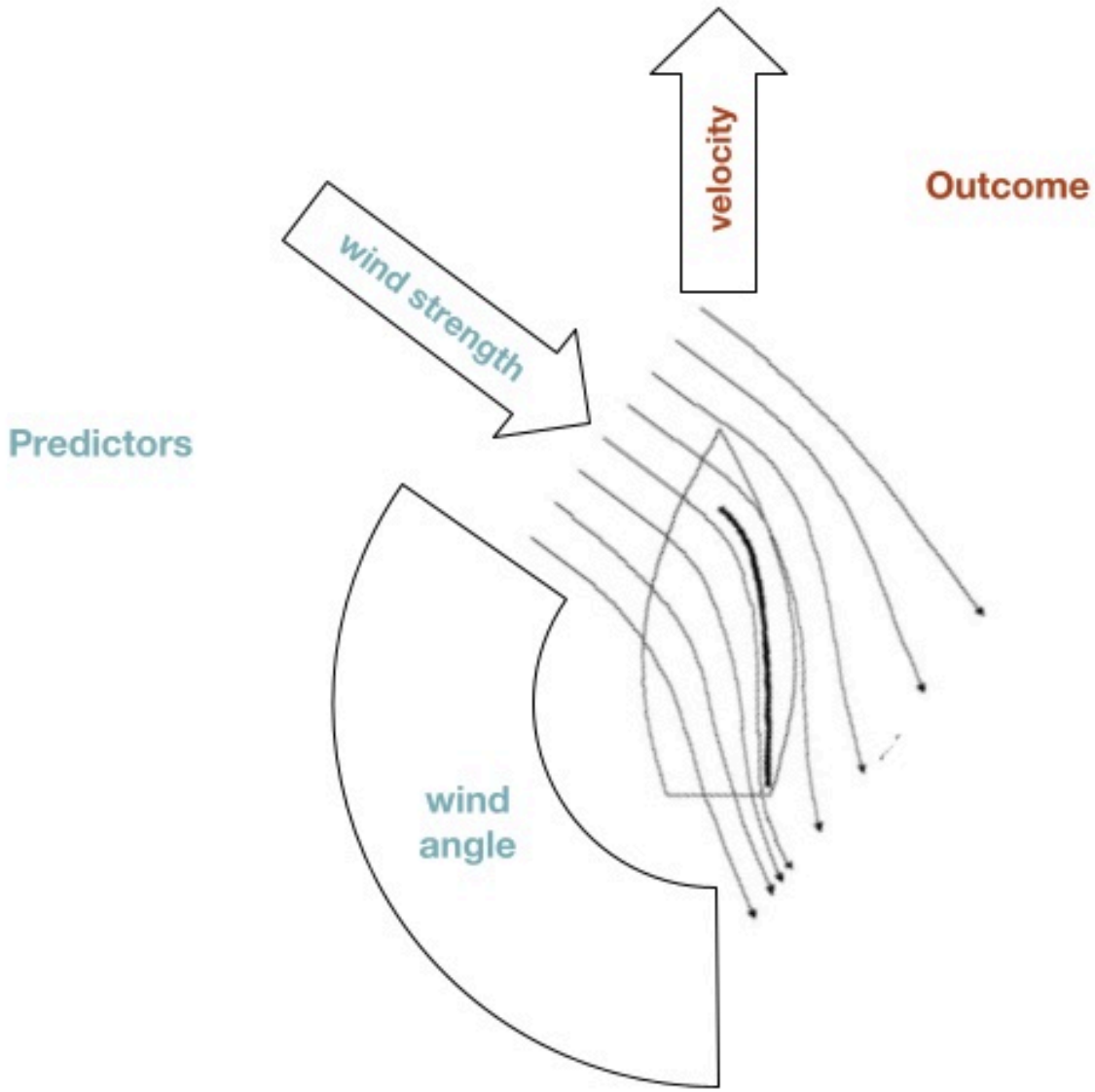
```clojure
(-> (wind-strength 6)
    (haclo/base
     {:=x :angle})
    (haclo/layer-point
     {:=y :velocity})
    (haclo/update-data  (fn [_]
                          (->
                           (wind-strength 6)
                           ;; Like R's formula: velocity ~ angle
                           ;; Here for demo, not necessary
                           (dm/create-design-matrix
                            [:velocity]
                            [[:angle '(identity angle)]])
                           (ml/train {:model-type :fastmath/ols})
                           (->>> (ml/predict (tc/dataset {:angle angles})))
                           (tc/add-column :angle angles)
                           (tc/rename-columns {:velocity :velocity-lm})))))
    (haclo/layer-line {:=y :velocity-lm}))
```



```
(-> (wind-strength 6)
    (haclo/base
     {:=x :angle})
    (haclo/layer-point
     {:=y :velocity})
    (haclo/update-data  (fn [_]
                          (->
                           (wind-strength 6)
                           ;; Like R's formula: velocity ~ angle
                           ;; Here for demo, not necessary
                           (dm/create-design-matrix
                            [:velocity]
                            [[:angle '(identity angle)]])
                           (ml/train {:model-type :fastmath/ols})
                           (->> (ml/predict (tc/dataset {:angle angles})))
                           (tc/add-column :angle angles)
                           (tc/rename-columns {:velocity :velocity-lm})))))
    (haclo/layer-line {:=y :velocity-lm}))
```


velocity, velocity-lm
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100
120
140
160
180
angle

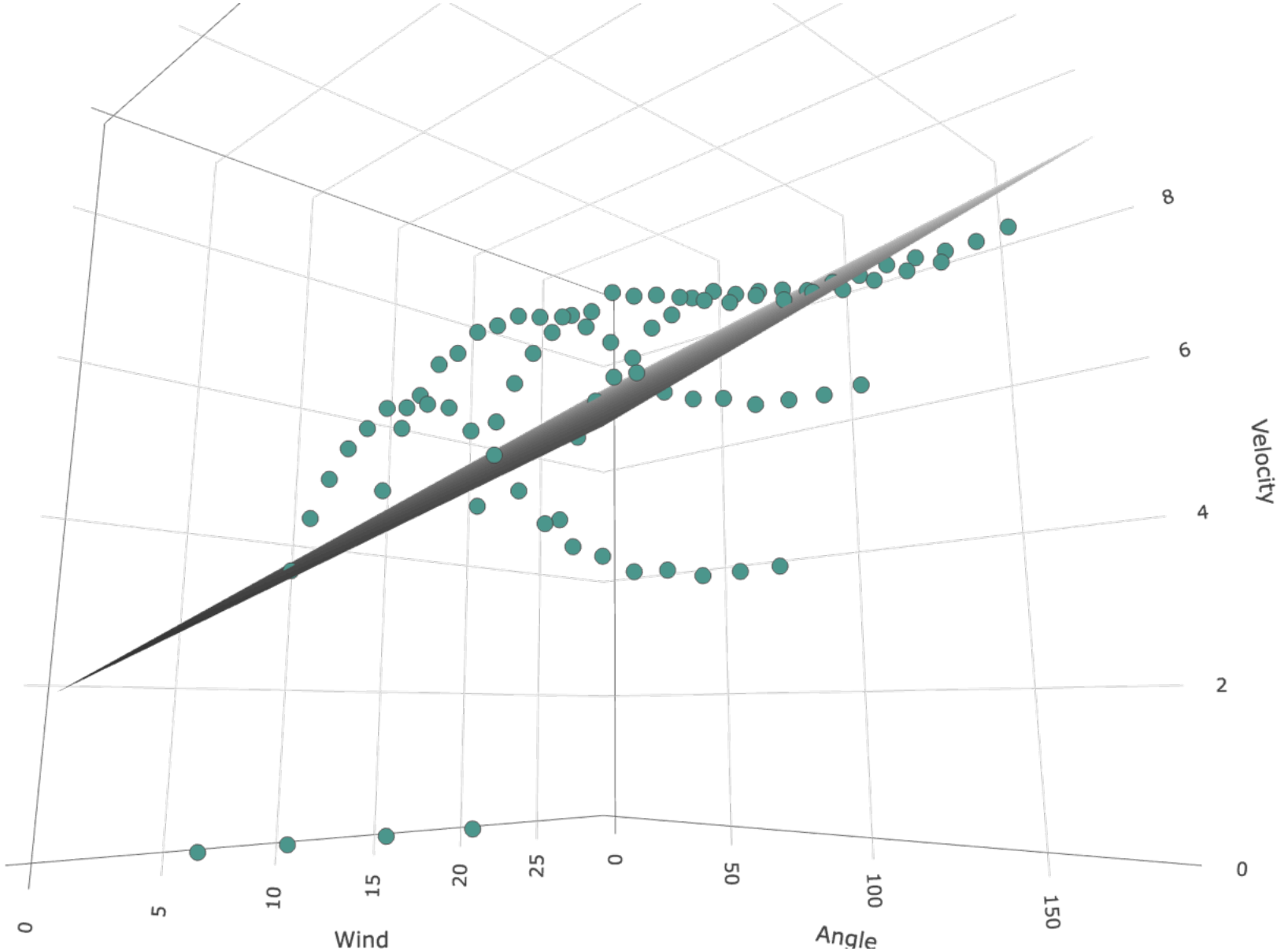
Multivariate


velocity
Outcome
wind strength
Predictors
wind
angle

$$\text{Velocity} = \alpha_0 + \alpha_1 * \text{Wind} + \alpha_2 * \text{Angle} + \text{Noise}$$

```
(kind/plotly
 {:data [(-> {:type :surface
              :colorscale "Greys"
              :cauto false
              :zmin 0
              :z z-trace-for-surface})
         (-> {:type :scatter3d
              :mode :markers
              :marker {:size 6
                       :line {:width 0.5
                              :opacity 0.8}}
              :x (tcc/- (:x training-data-trace)
                        min-angle)
              :y (tcc/- (:y training-data-trace)
                        min-wind)
              :z (:z training-data-trace)})]
  :layout {:width 600
           :height 700}}))
```

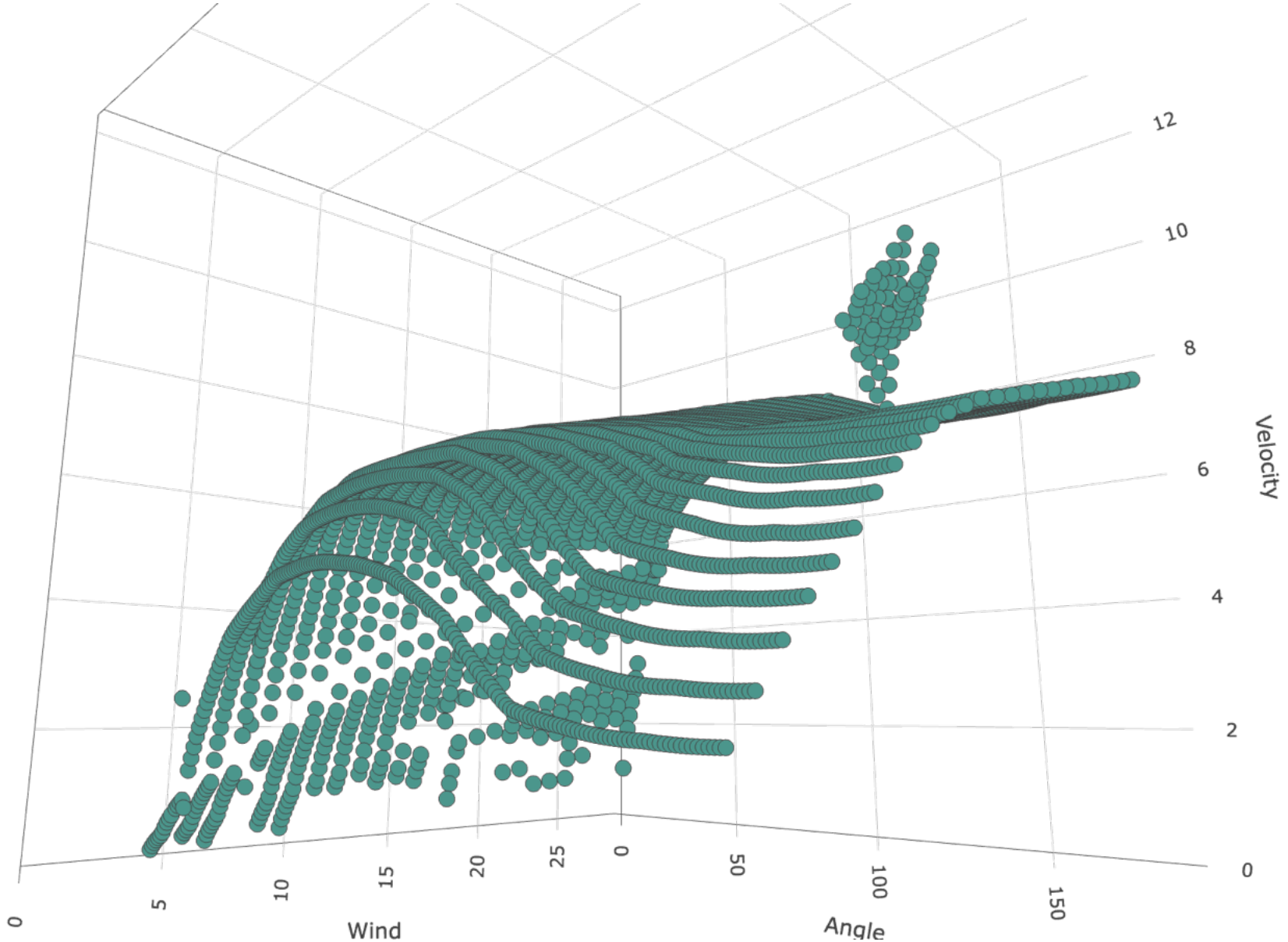
Multivariate regression



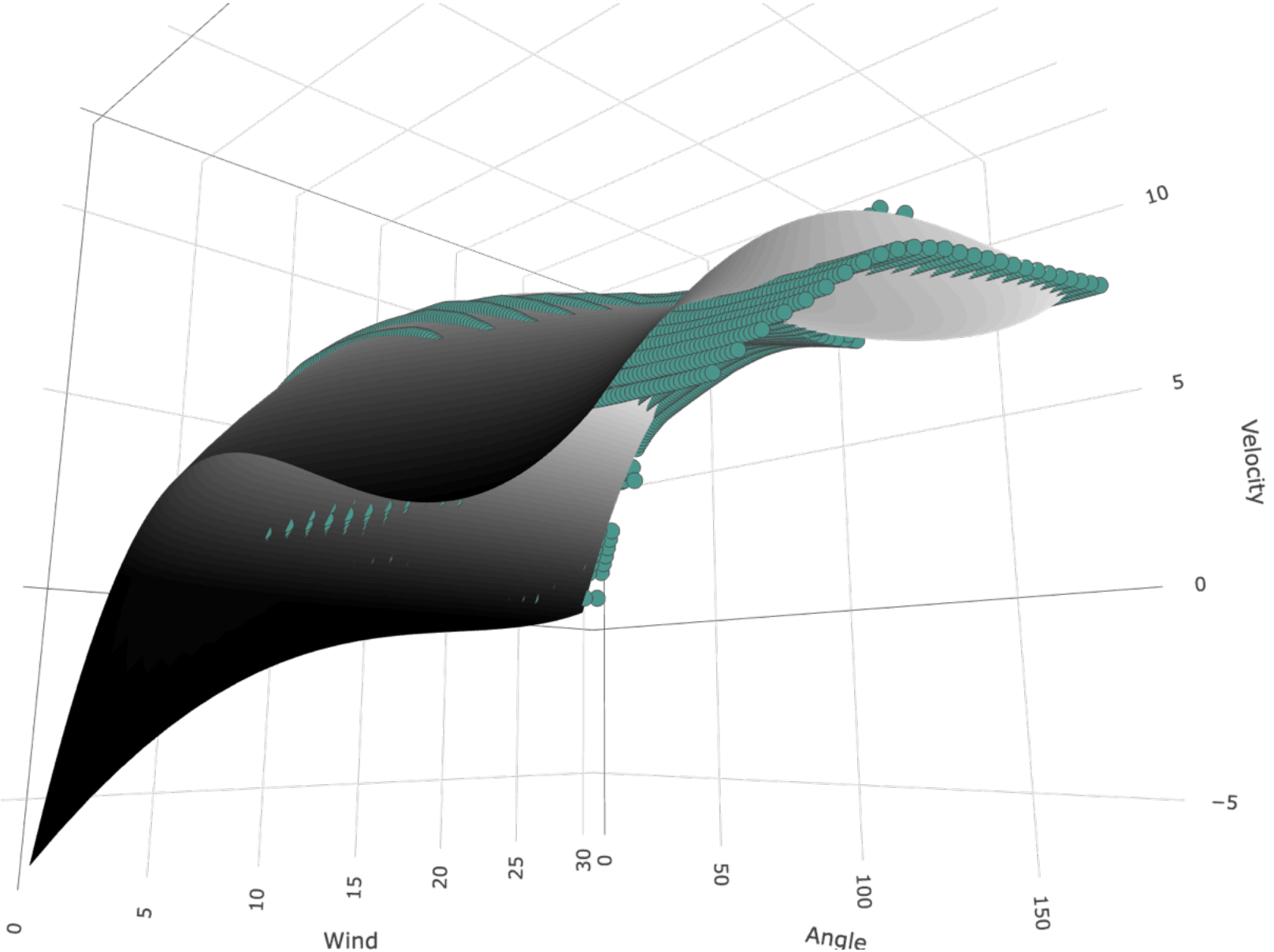
```
[[:velocity]
 [[:angle '(identity angle)]
  [:angle2 '(* angle angle)]
  [:angle3 '(* angle angle angle)]
  [:wind '(identity wind)]
  [:wind2 '(* wind wind)]
  [:wind3 '(* wind wind wind)]]]
```

$$\text{Velocity} = \alpha_0 + \alpha_1 * \text{Wind} + \alpha_2 * \text{Wind}^2 + \alpha_3 * \text{Wind}^3$$
$$\alpha_4 * \text{Angle} + \alpha_5 * \text{Angle}^2 + \alpha_6 * \text{Angle}^3 + \text{Noise}$$

Mo data, mo problems



Better model



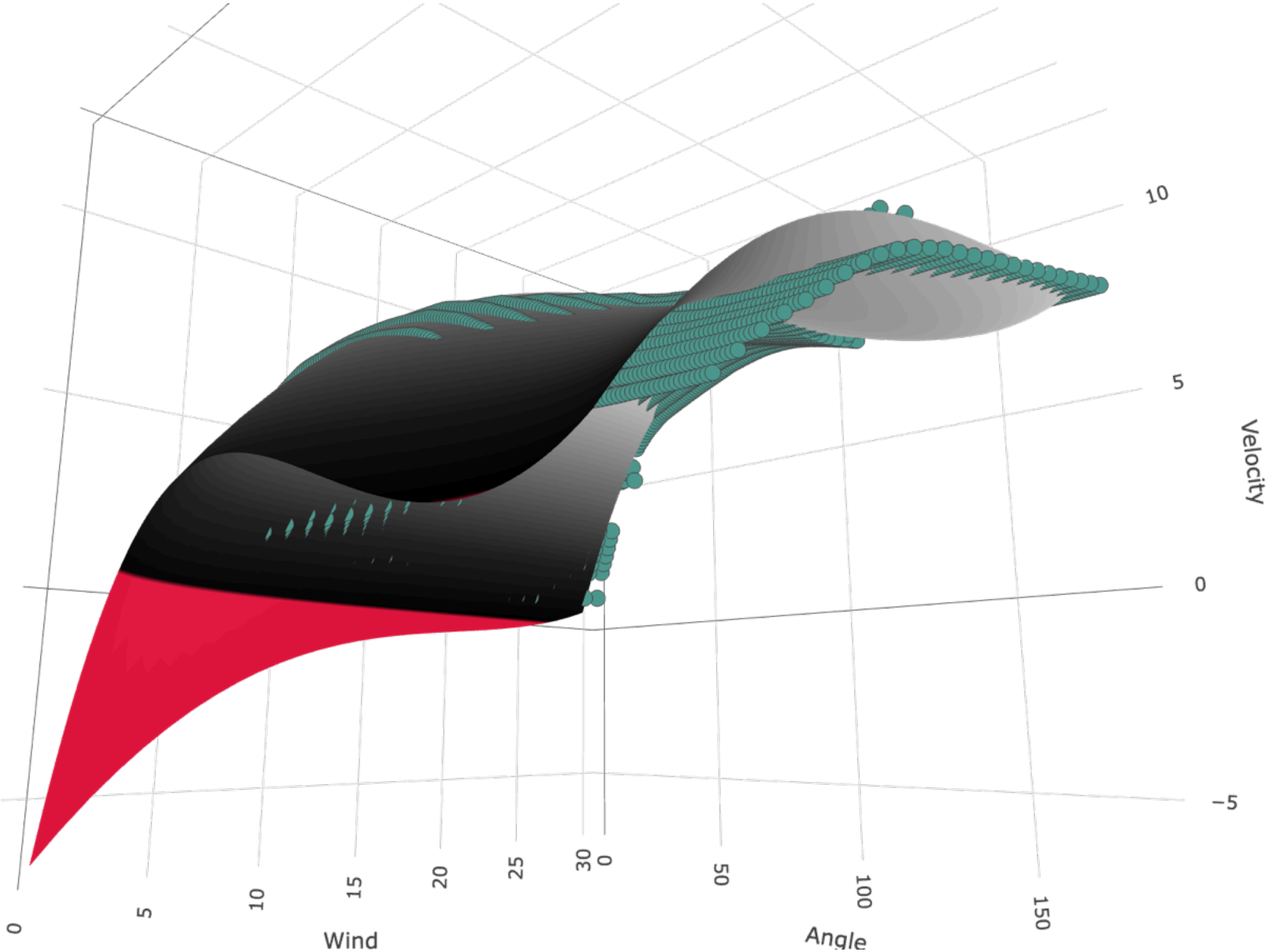
Custom gradient

```
^:kind/hidden
(def color-boundries
  [0.011111111111
   0.222222222222
   0.333333333333
   0.444444444444
   0.555555555556
   0.666666666667
   0.777777777778
   0.888888888889
   1.0])
```

Custom colourscale

```clojure
^:kind/hidden
(def custom-color-scale
  (into [[0.0 "rgb(220, 20, 60)"]]
        (mapv (fn [c n]  [c (str "rgb(" n
                                 ", " n
                                 ", " n
                                 ")")])
              color-boundries
              (->> color-boundries
                   count
                   (/ 255) ;; sorry Stuart
                   int
                   (range 1 255)))))
```

Analysis



What next?

- Combine data
- Assumptions
- Uncertainty
- An intuitive mental model

Bayesian

Bayesian Statistics

Bayesian Statistics

- Everything is a variable with uncertainty
- Like a datalog query

$$\text{Velocity} = \alpha_0 + \alpha_1 * \text{Wind} + \alpha_2 * \text{Wind}^2 + \alpha_3 * \text{Wind}^3$$
$$\alpha_4 * \text{Angle} + \alpha_5 * \text{Angle}^2 + \alpha_6 * \text{Angle}^3 + \text{Noise}$$

STAN in the house

```
data {
    int n;
    vector[n] angle;
    vector[n] angle2;
    vector[n] angle3;
    vector[n] wind;
    vector[n] wind2;
    vector[n] wind3;
    vector[n] velocity;
    array[n] int<lower=0,upper=1> should_observe;
}
parameters {
    real a0;
    real a1_angle;
    real a2_angle;
    real a3_angle;
    real a1_wind;
    real a2_wind;
    real a3_wind;
    real<lower=0> sigma;
}
transformed parameters {
    vector[n] mu;
    mu = a0 + a1_angle * angle + a2_angle * angle2 + a3_angle * angle3 + a1_wind * wind + a2_wind * wind
}
```

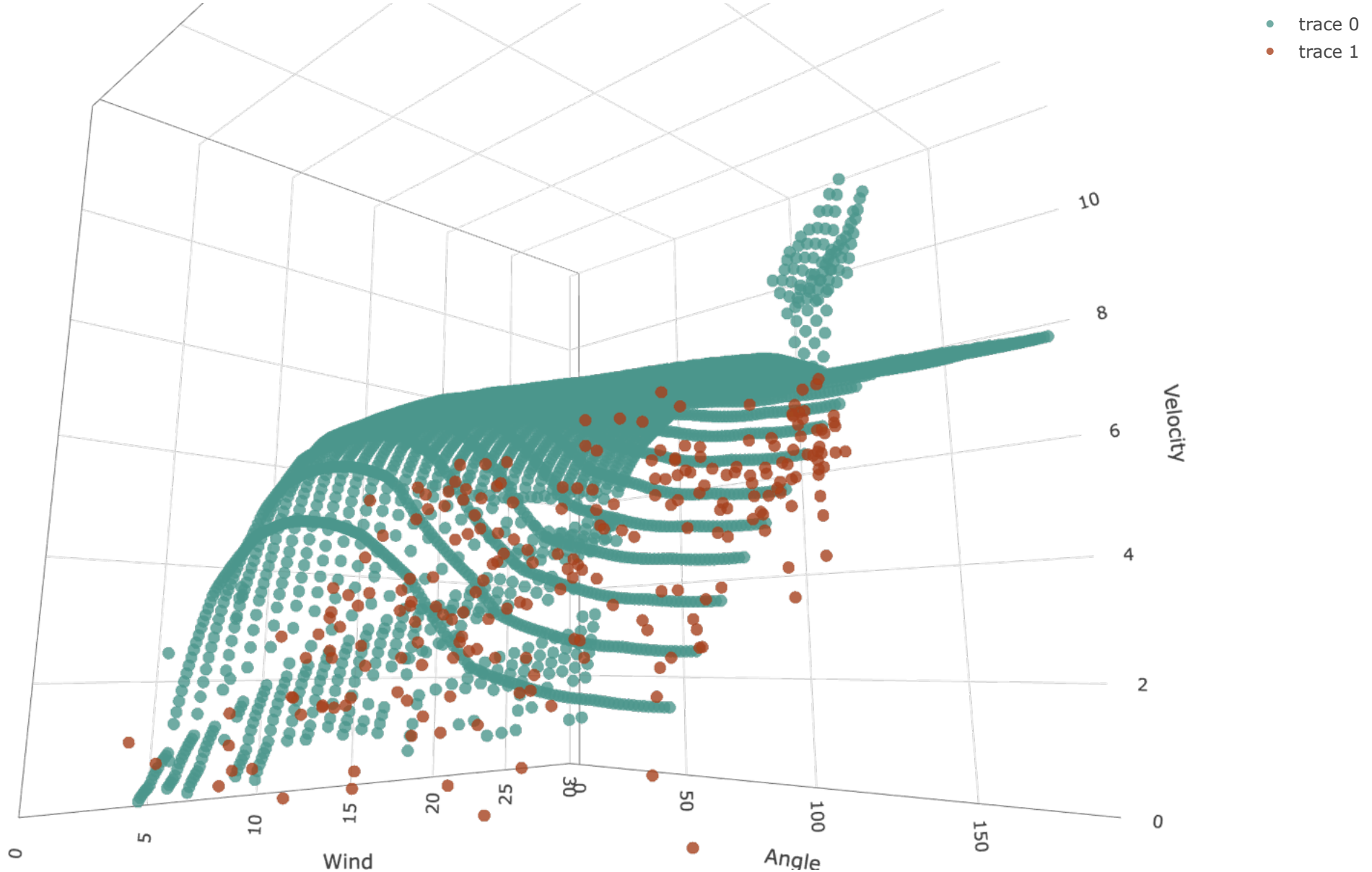
Compiling the model

```
(def jon-polynomial-model
  (delay
    (stan/model core/jon-polynomial-model-code)))
```

Training the model

```
(-> full-training-data
    (tc/drop-columns [:part])
    (->> (into {}))
    (update-vals vec)
    (assoc :n (tc/row-count full-training-data))
    (#(stan/sample @jon-polynomial-model
                   %
                   {:num-samples 200}))
    :samples)
```

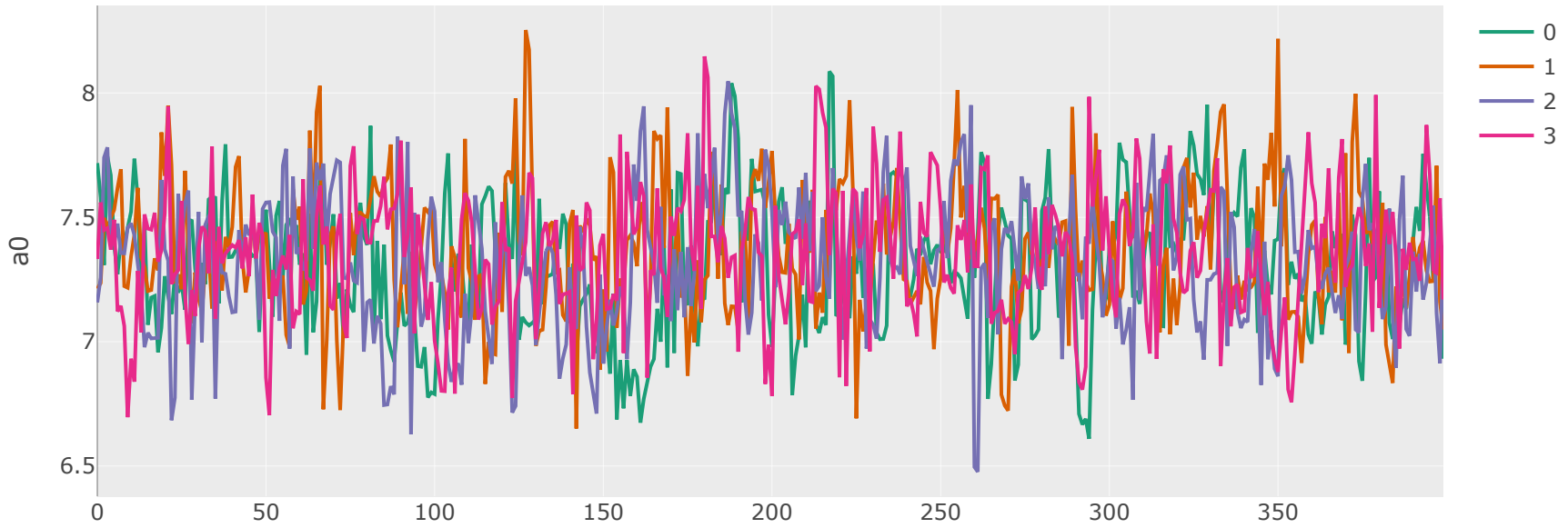
Real measurements + syntethetic data

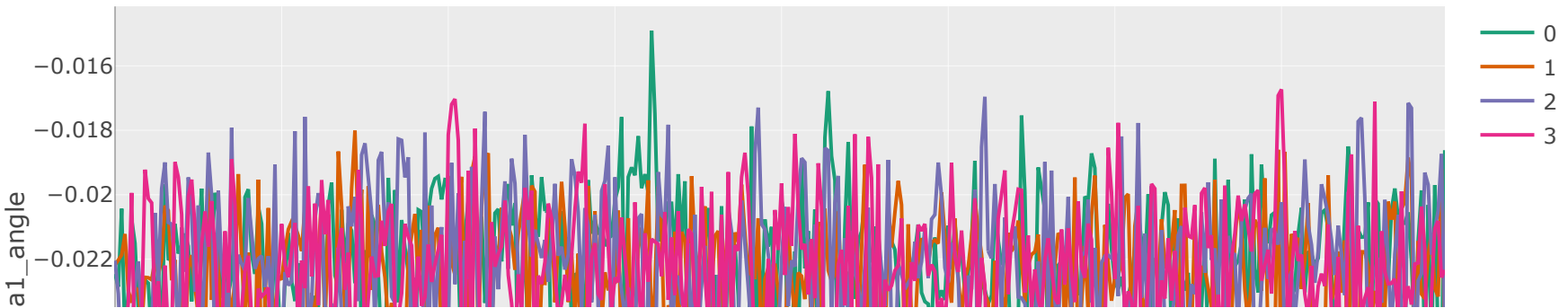


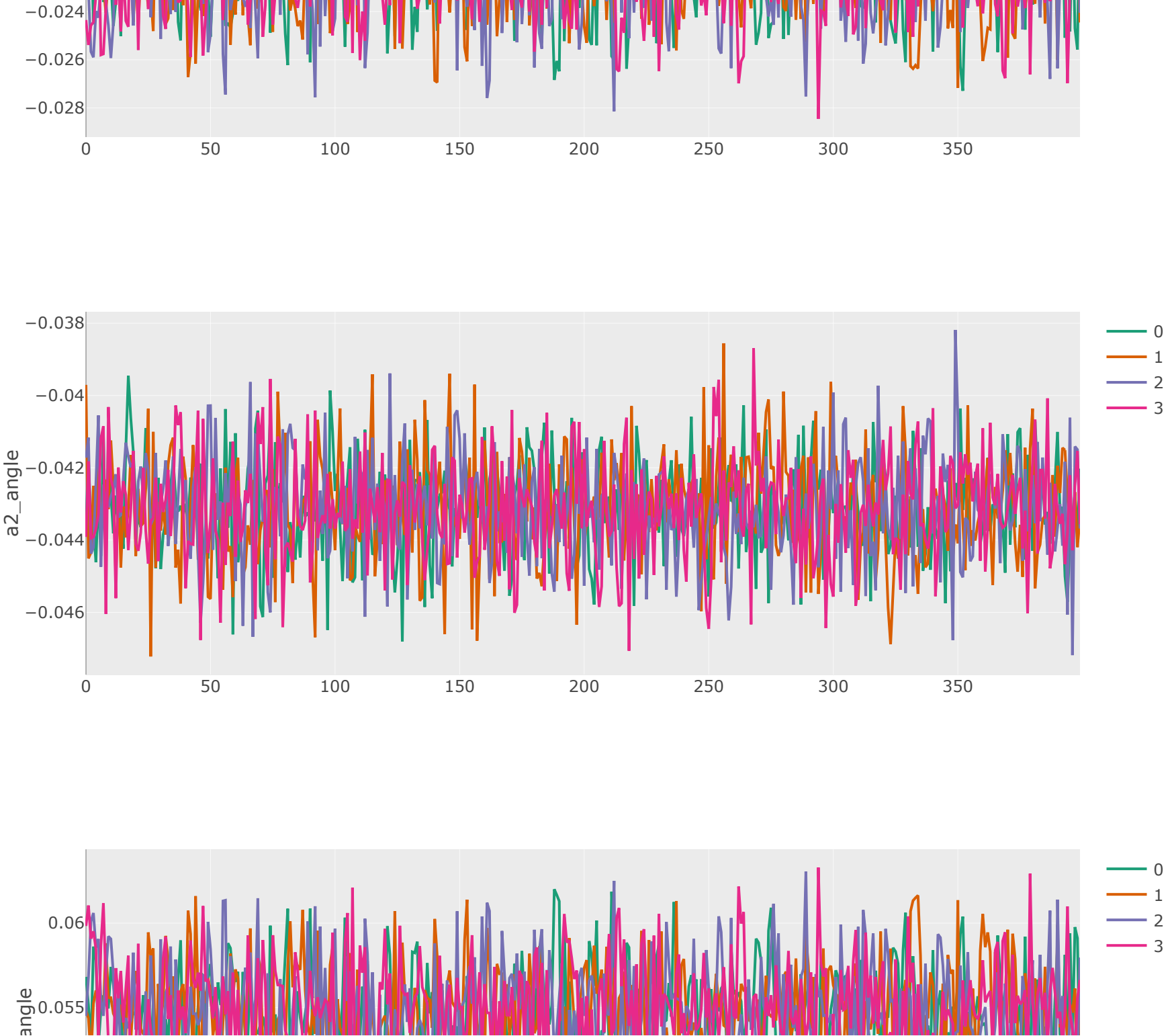
Creating plots for diagnostics

```
    (for [k [:a0
             :a1_angle :a2_angle :a3_angle ;; :a4_angle
             :a1_wind :a2_wind :a3_wind    ;; :a4_wind
             :sigma]]
      (-> samples
          (ploclo/layer-point {:=x :i
                               :=y k
                               :=color :chain
                               :=color-type :nominal})
          ploclo/plot))
```

Doing diagnostic

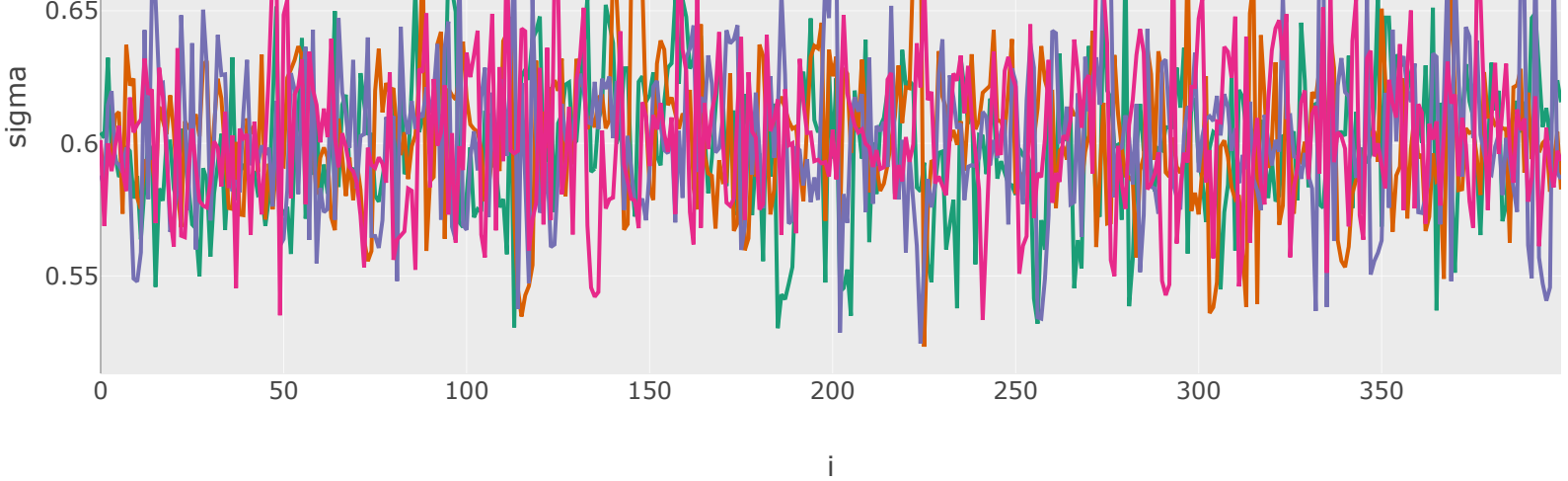




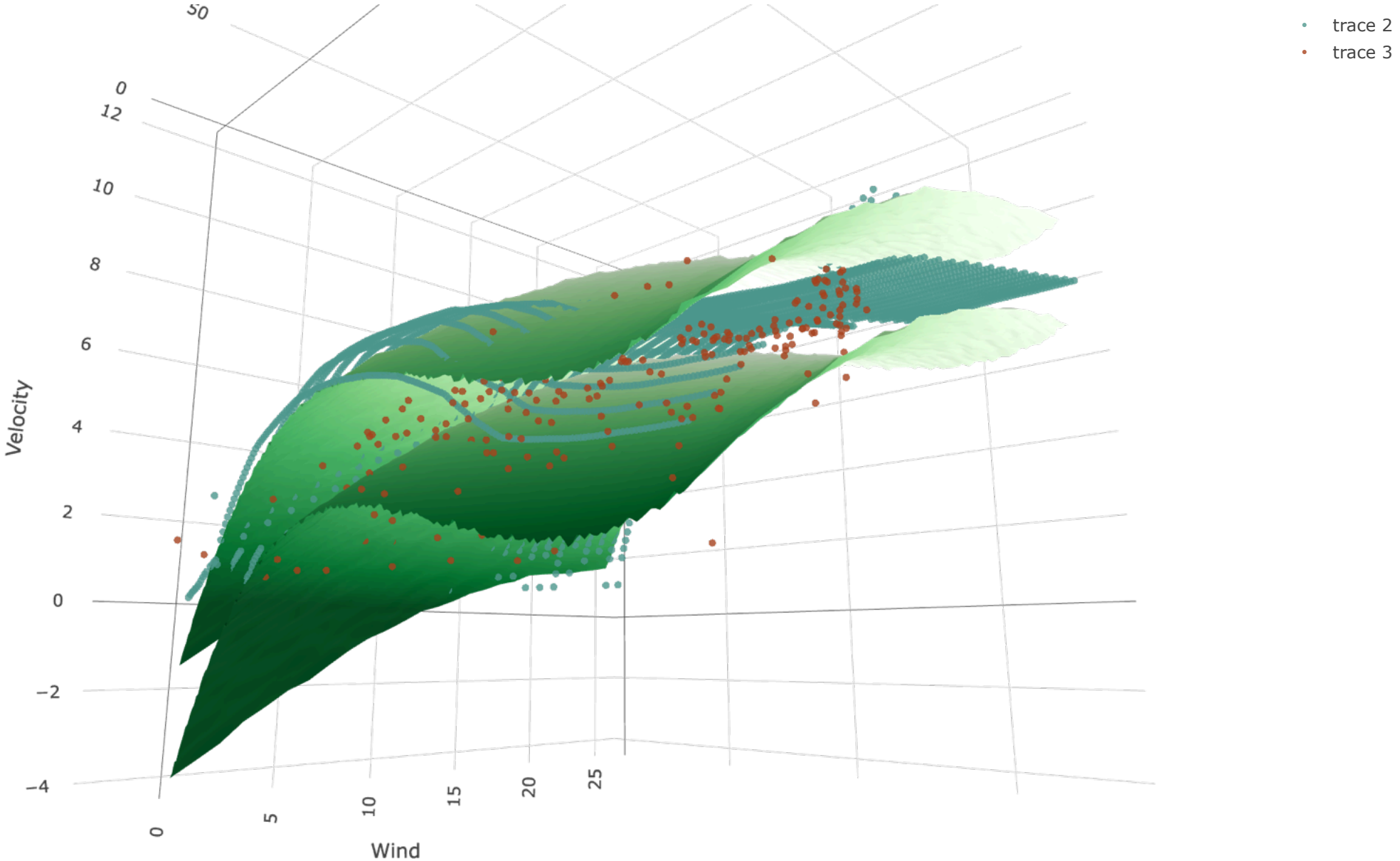

−0.024
−0.026
−0.028
0
50
100
150
200
250
300
350
−0.038
−0.04
−0.042
−0.044
−0.046
a2_angle
0
1
2
3
0.06
0.055
angle
0
1
2
3


0.05
0.045
0
50
100
150
200
250
300
350
a1_wind
0.16
0.14
0.12
0.1
0.08
0
50
100
150
200
250
300
350
0
1
2
3
−0.1
−0.12
0
1
2
3


a2_wind
−0.14
−0.16
−0.18
0
50
100
150
200
250
300
350
a3_wind
0.2
0.15
0.1
0.05
0
50
100
150
200
250
300
350
0
1
2
3
0.7
0
1
2
3

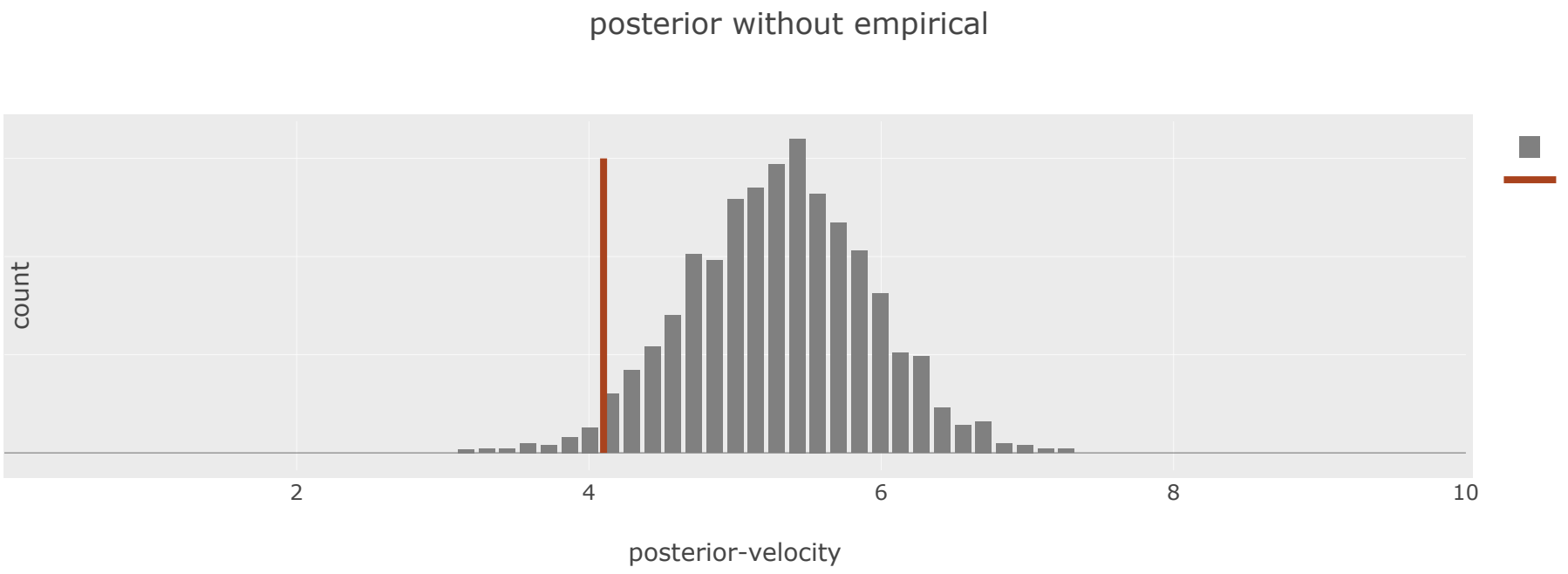

sigma
0.65
0.6
0.55
0
50
100
150
200
250
300
350
i

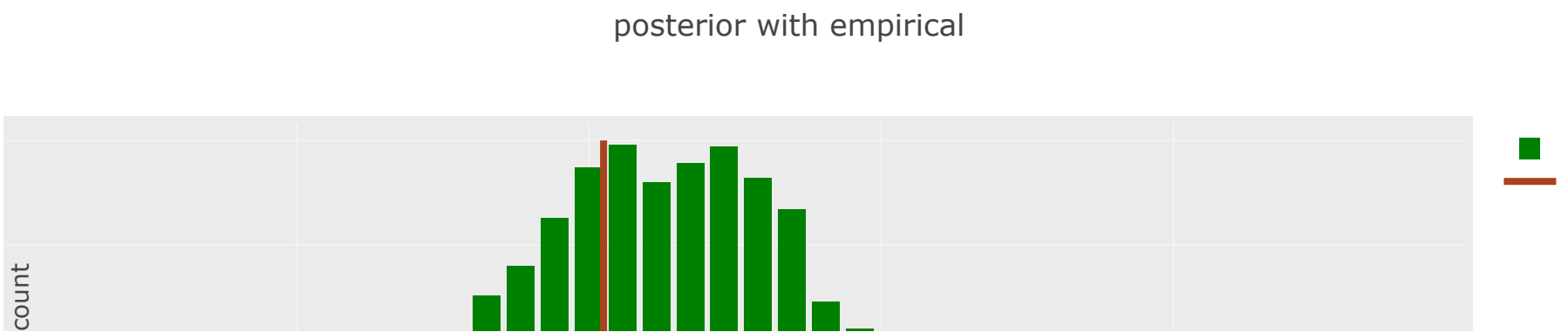
Updating the synthetic model with the measurements

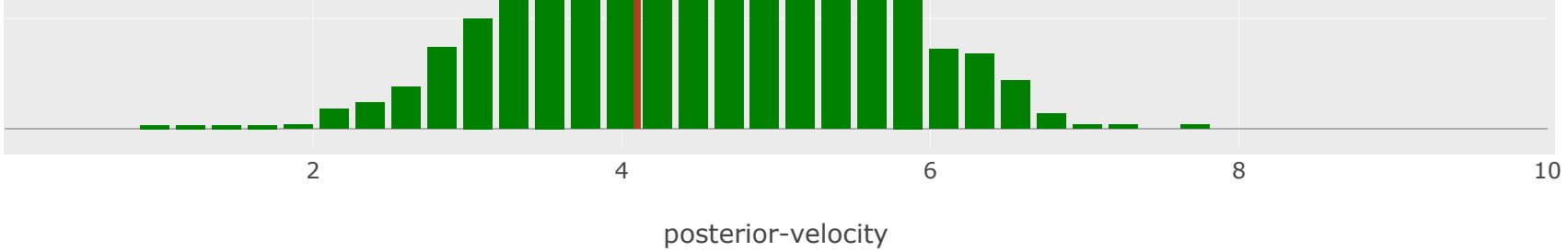


Comparing synthetic with empirical

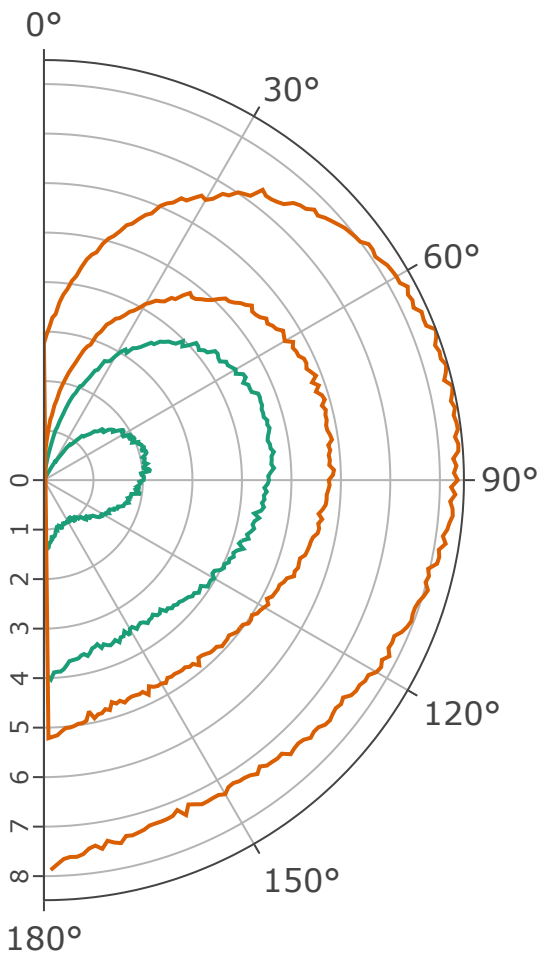
empirical example #151 angle:175.5 wind: 10.6 velocity:4.1






2
4
6
8
10
posterior-velocity

drumroll.. Polars



Here be dolphins!

- Clojure data science tools are here!

Selected Dependencies for this project

- Tablecloth
- Tech.ml
- Metamorph
- Fastmath
- PythonVPP
- Hanami
- Hanamicloth and Plotlycloth
- Cmdstan-clj
- Clay
- Noj, released today!

Noj

- “Noj” all Clojure data science libraries and documentation in one place!
- Released today! Noj v2 alpha FTW!

Opportunities!

- Open source mentoring!
- The website: https://scicloj.github.io/

Some heroes

- Generateme
- Chris Nuernberger
- Jon Anthony
- Carsten Behring
- Kira McLean
- Timothy Pratley
- Daniel Slutsky (also the co-author of of this presentation)
- The Scicloj Community!

Thank you!

Project code: https://github.com/skallinen/jon30