```clojure
(ns drifthappens.predreprod1
  (:require ;[clojure.math :as m]
            [fastmath.vector :as fvec]
            [fastmath.matrix :as fmat]
            [fastmath.core :as fm]
            [scicloj.kindly.v4.kind :as kind]
            [scicloj.tableplot.v1.plotly :as plotly]
            [tablecloth.api :as tc]
            [drifthappens.wrightfisher :as wf]
            [utils.plotly :as uplot]
            [utils.math :as umath]
            [utils.misc :as umisc])
  (:import [fastmath.vector Vec2 Vec3 Vec4]
           [fastmath.matrix Mat2x2 Mat3x3 Mat4x4]))
```

```clojure
(defn make-prob-states
  [tran-mats init-state]
  (mapv (fn [m] (fmat/mulv m init-state))
        tran-mats))
```

```clojure
(def fit-B 1.0)
```

```clojure
(def big-fit-A 1.01)
```

large size has sel

```clojure
(def small-fit-A 1.0)
```

small size is pure drift

```clojure
(def increments (iterate (partial + 16) 0))
```

```clojure
(def generations (map inc increments))
```

```clojure
(def half-generations (map (fn [n] (inc (/ n 2))) increments))
```

```clojure
(comment
  (take 10 increments)
  (take 10 generations)
```

```
    (take 10 half-generations)
    (select-keys [1 5 7 9] [0 2 3])
  )
```

```
  (def num-gens 20)
```

should be even;

```
  (def small-N 10)
```

```
  (def big-N 2000)
```

```
  (def half-small-N (/ small-N 2))
```

```
  (def half-big-N (/ big-N 2))
```

```
  (def small-pop-init (wf/mkvec (concat (repeat half-small-N 0.
```

```
  (def small-drift-mat (wf/right-mult-tran-mat small-fit-A fit-
```

```
  (def small-tran-mats (umath/make-mat-powers small-drift-mat (
```

```
  (def small-prob-states (make-prob-states small-tran-mats smal
```

```
  (def small-plots (mapv uplot/plot-both small-prob-states))
```

```
  (def big-pop-init (wf/mkvec (concat (repeat half-big-N 0.0) [
```

```
  (def big-drift-mat (wf/right-mult-tran-mat big-fit-A fit-B (d
```

use fit-B for fit-A to make them equal

```
  (def big-tran-mats (umath/make-mat-powers big-drift-mat (take
(def big-prob-states (make-prob-states big-tran-mats big-pop-i
```

```
(def big-plots (mapv uplot/plot-lines big-prob-states))
```

small-plots big-plots

```
(def N (dec (count big-pop-init)))
```

```
(def M (dec (count small-pop-init)))
```

Shrinking and expanding matrices. Note M and N are swapped in the next two

```
(def predat-drift-mat (wf/right-mult-tran-mat small-fit-A fit-
```

```
(def reprod-drift-mat (wf/right-mult-tran-mat big-fit-A fit-B
```

use fit-B for fit-A to make them equal Combine into one square matrix:

```
(def pred-reprod-mat (fmat/mulm reprod-drift-mat predat-drift-
```

We use half-generations here because each step involves two sampling processes. So each generation is analogous to two generations in the small and big models.

```
(def pred-reprod-tran-mats (umath/make-mat-powers pred-reprod-
```

```
(def pred-reprod-prob-states (make-prob-states pred-reprod-tra
```

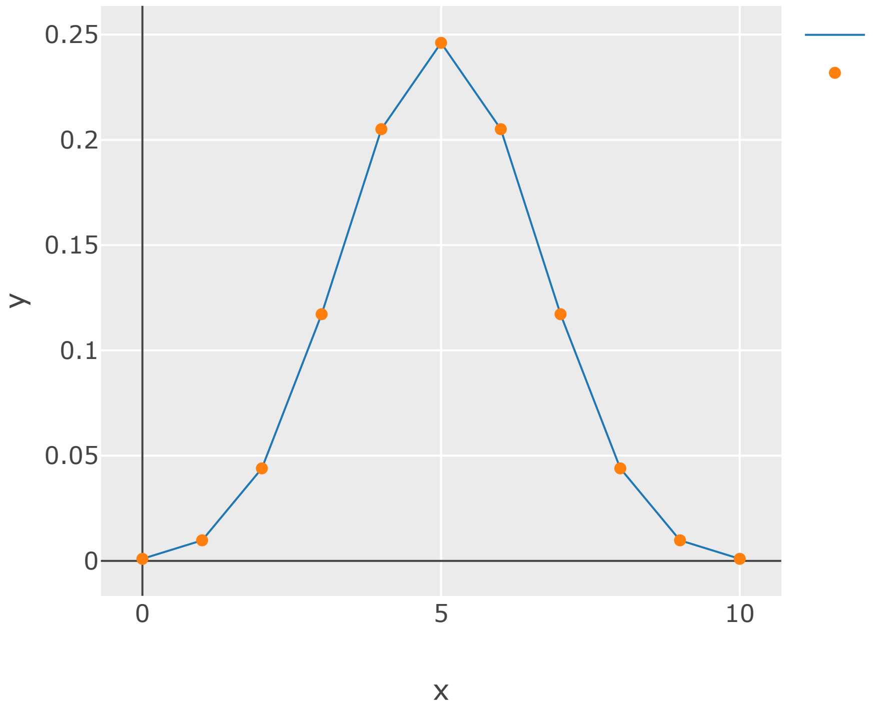
```
(def pred-reprod-plots (mapv uplot/plot-both pred-reprod-prob-
```

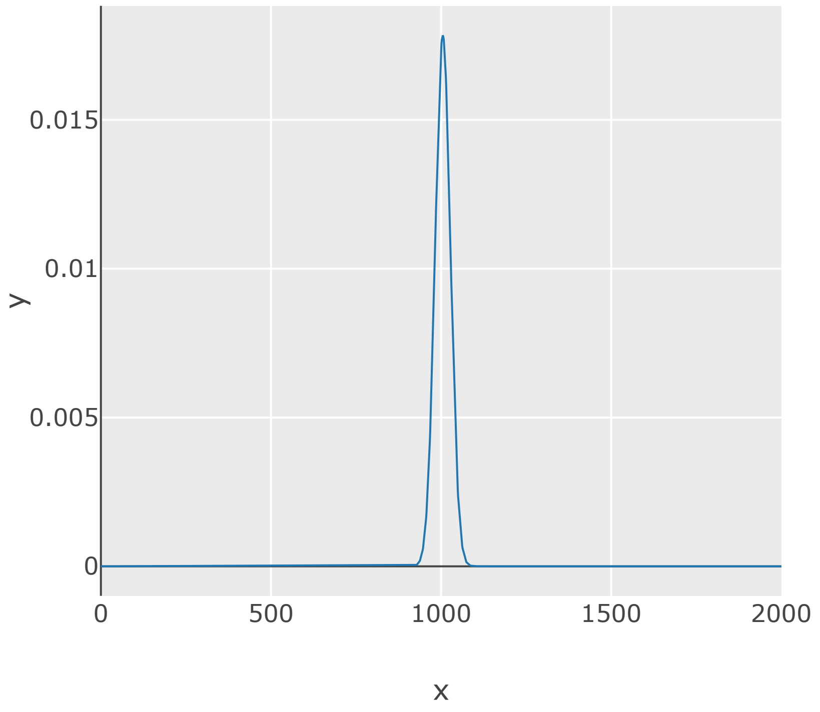
pred-reprod-plots

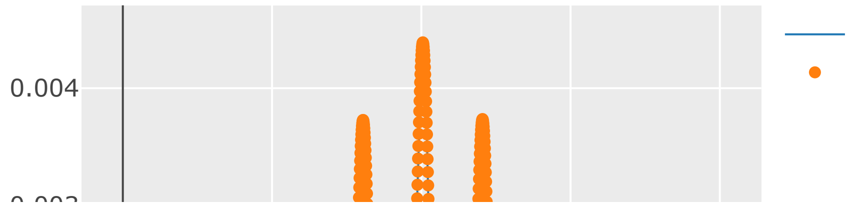
```
(def small-big-combo-plots (interleave small-plots big-plots p
```

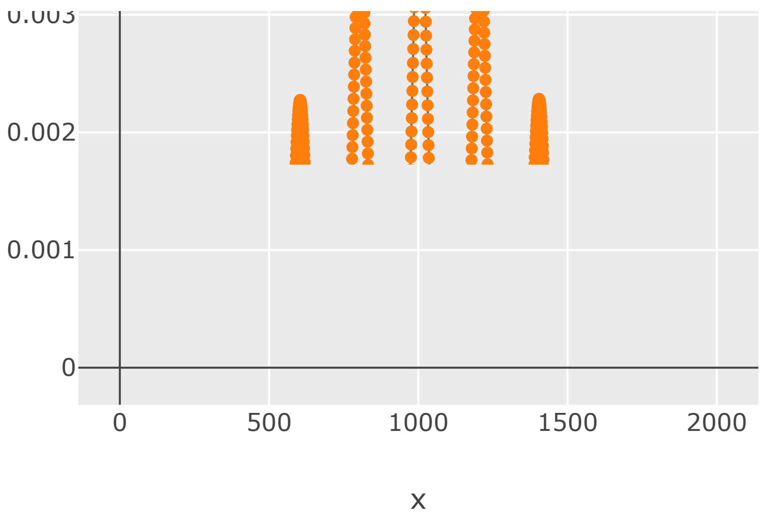
```
small-big-combo-plots
```

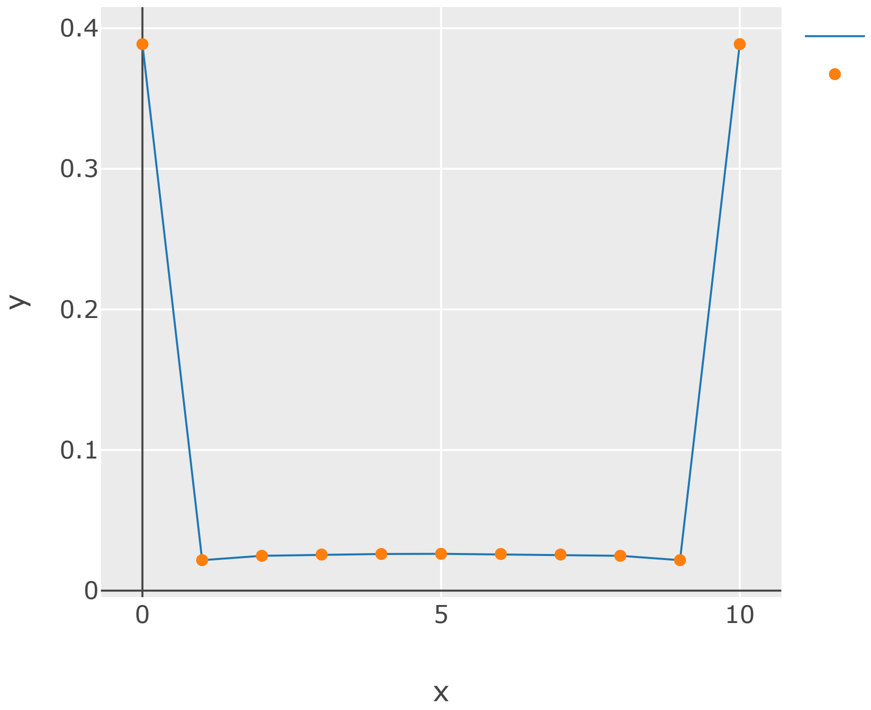
(

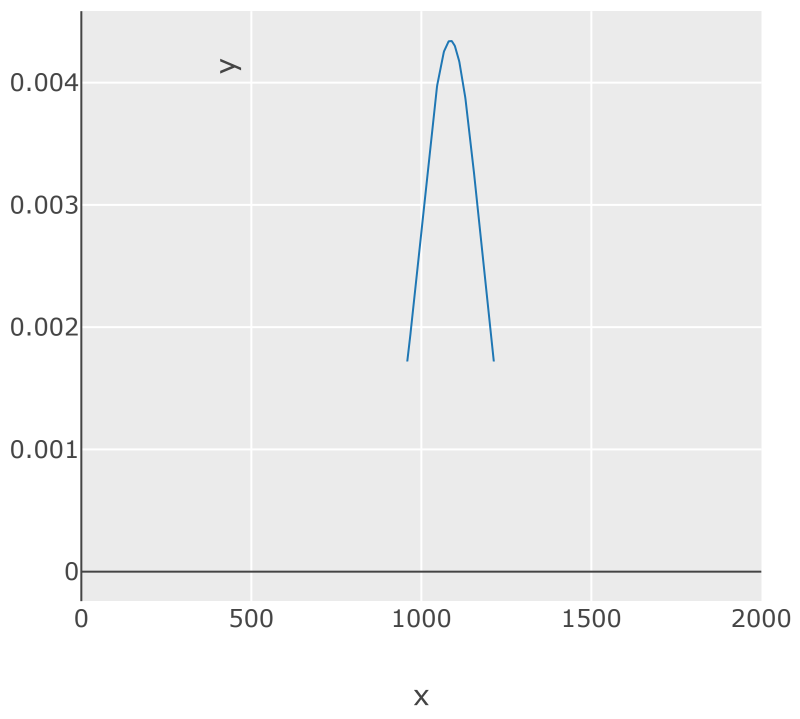

0.25
0.2
0.15
0.1
0.05
0
y
0
5
10
x

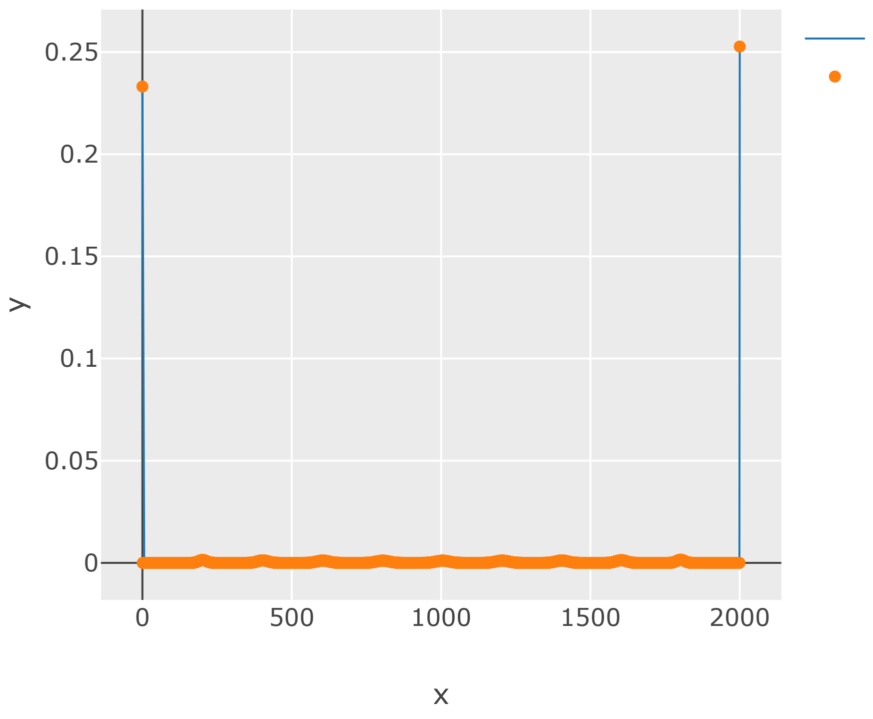

0.015
0.01
0.005
0
y
0
500
1000
1500
2000
x


0.004

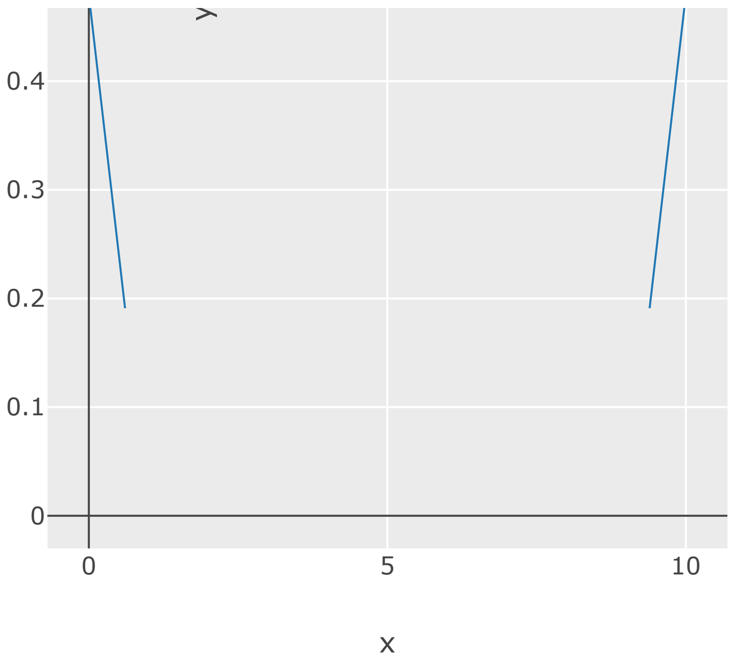

0.003
0.002
0.001
0
0
500
1000
1500
2000
x


0.4
0.3
0.2
0.1
0
y
0
5
10
x

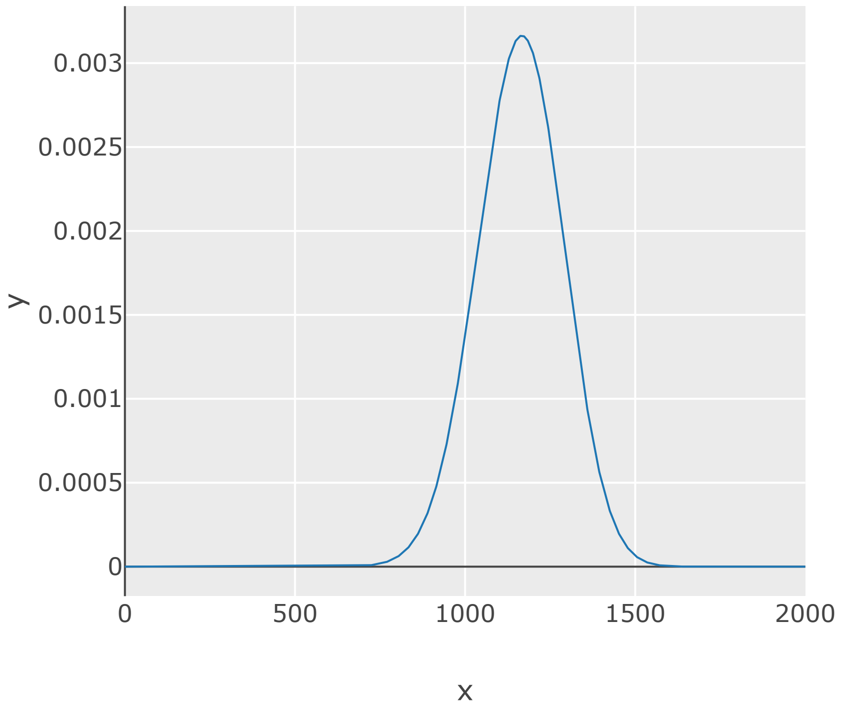

y
0.004
0.003
0.002
0.001
0
0
500
1000
1500
2000
x


0.25
0.2
0.15
0.1
0.05
0
y
0
500
1000
1500
2000
x

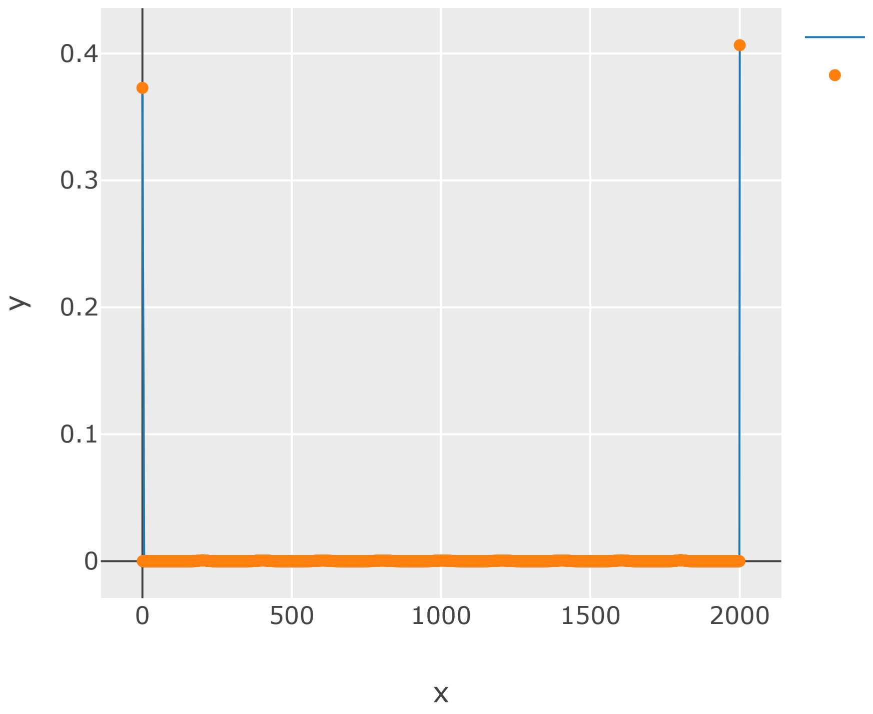
0.5

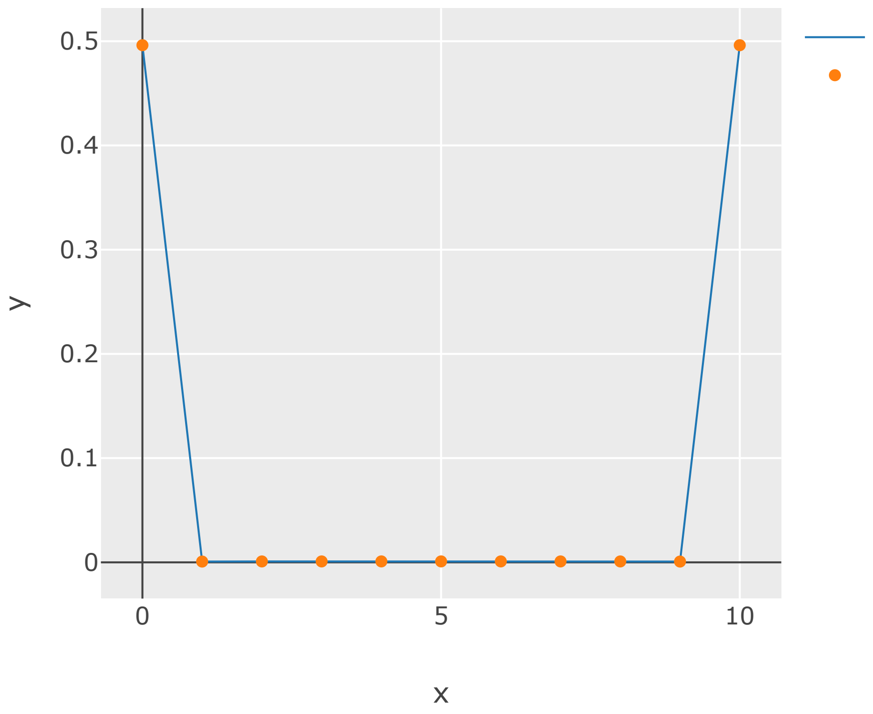


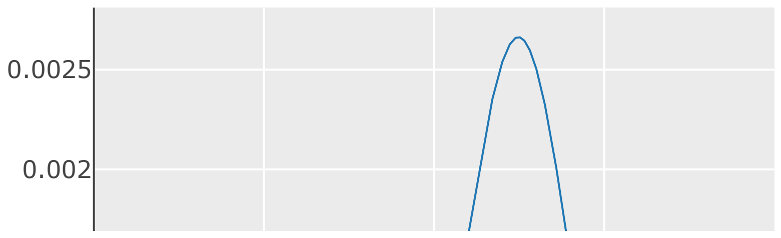
y
0.4
0.3
0.2
0.1
0
0
5
10
x

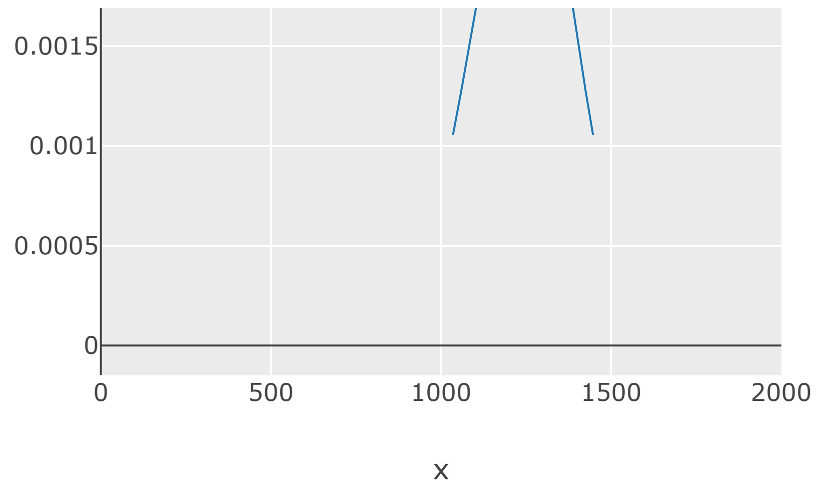


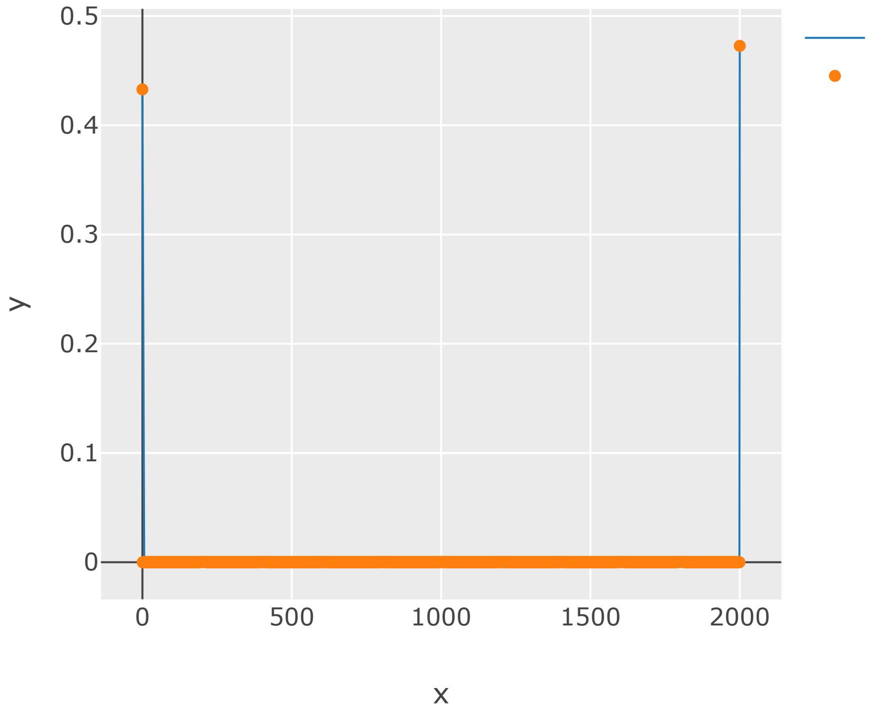
y
0.003
0.0025
0.002
0.0015
0.001
0.0005
0
0
500
1000
1500
2000
x

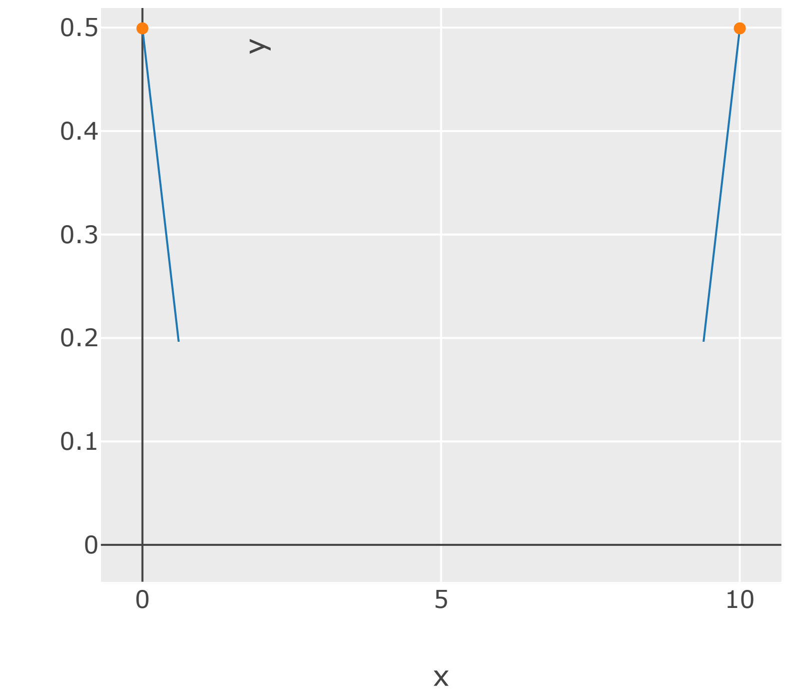

0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

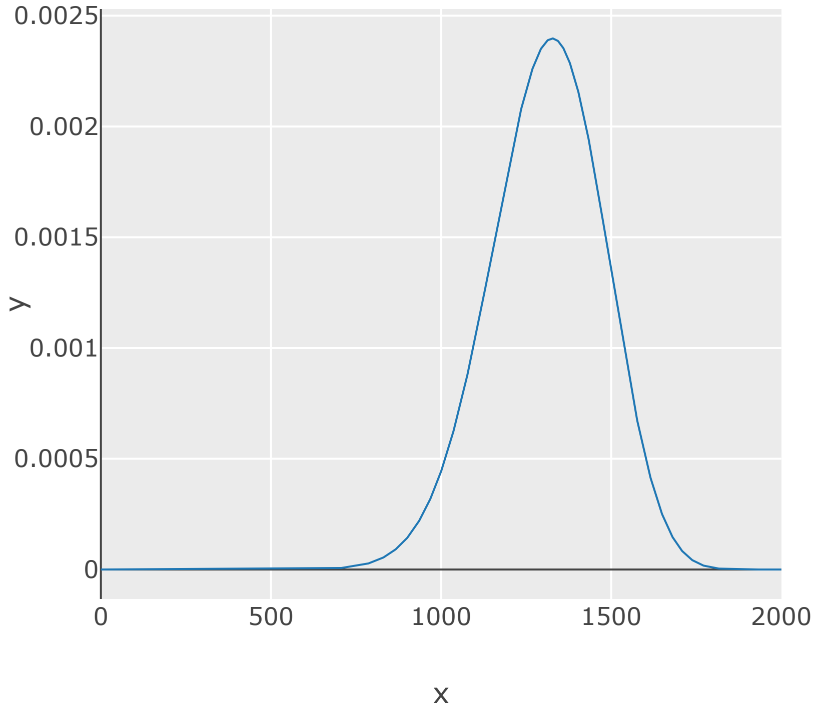

0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

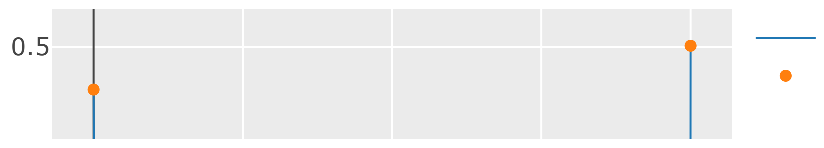

0.0025
0.002

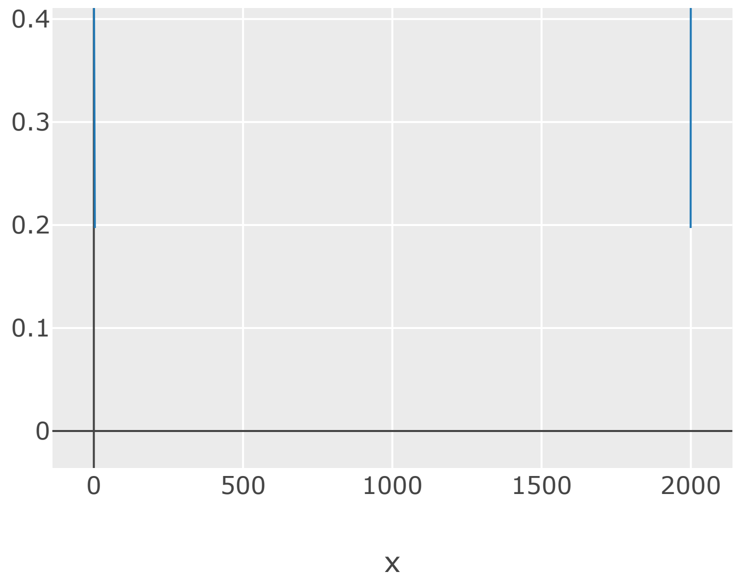

0.0015
0.001
0.0005
0
0
500
1000
1500
2000
x

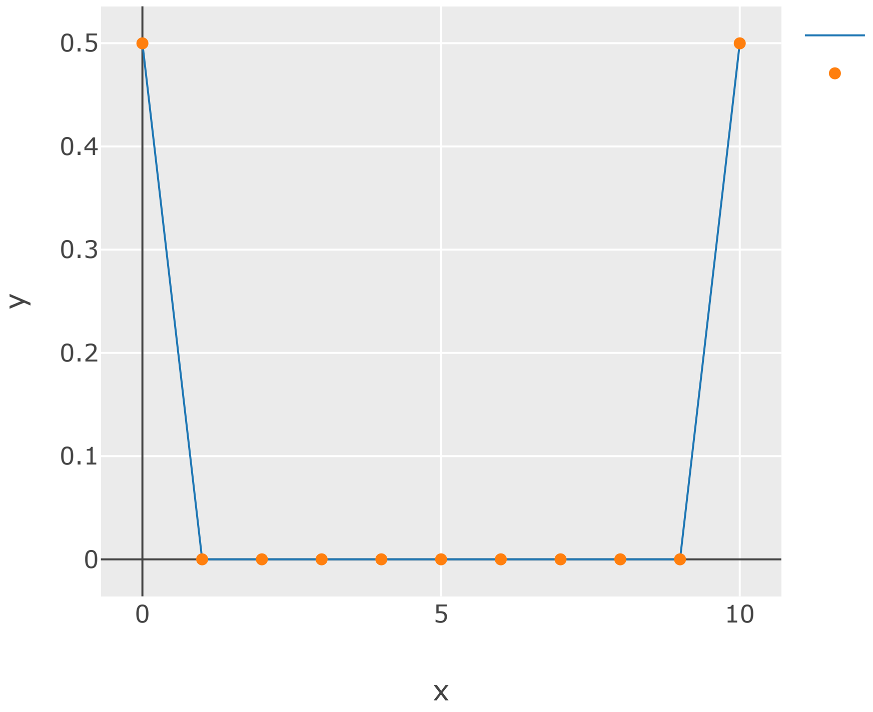

0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

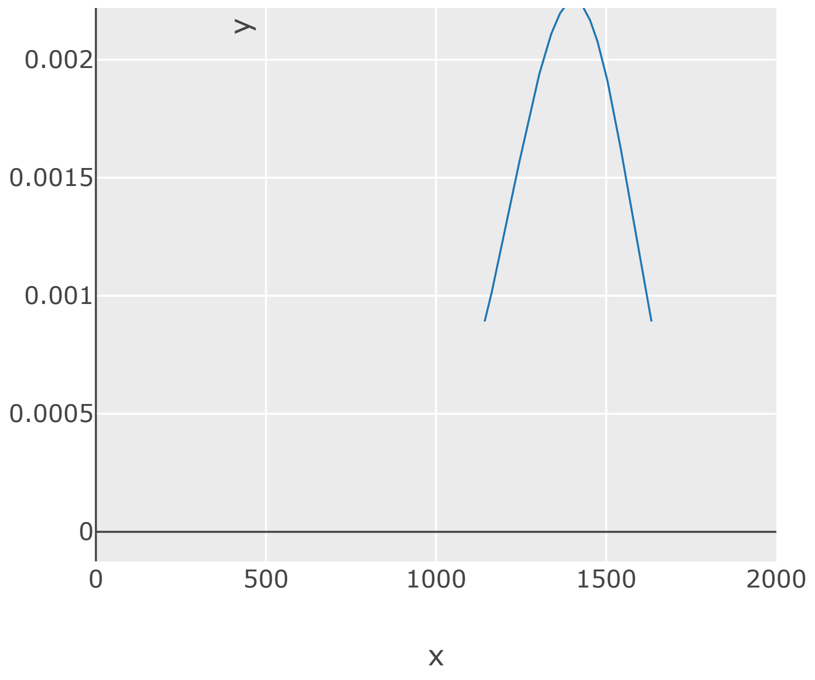

0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

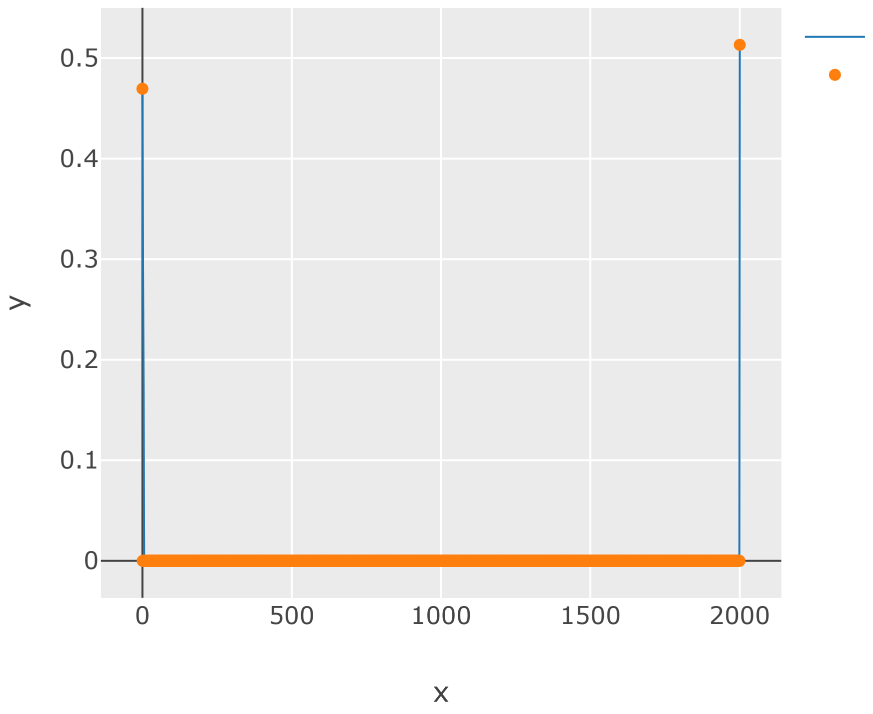

0.0025
0.002
0.0015
0.001
0.0005
0
y
0
500
1000
1500
2000
x

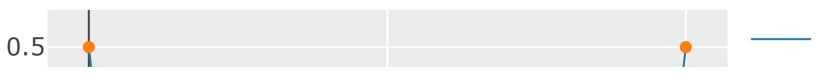

0.5

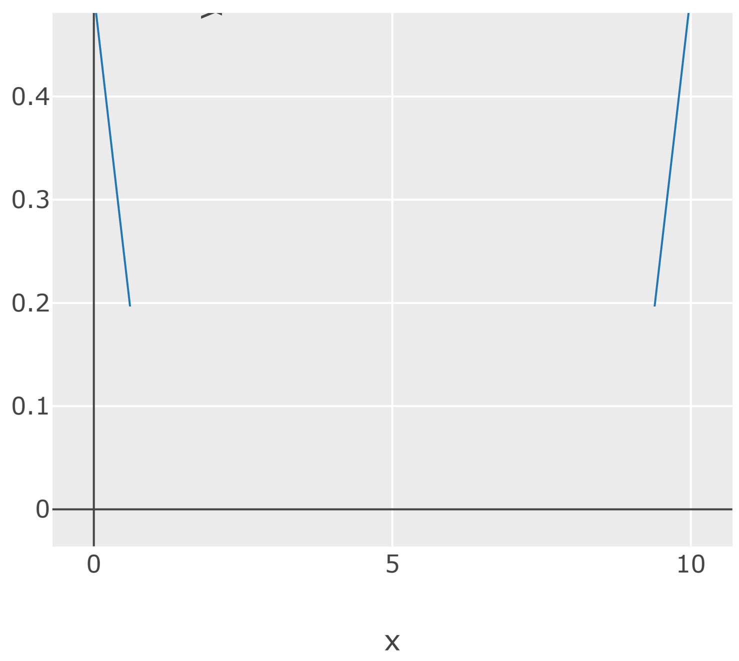

0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

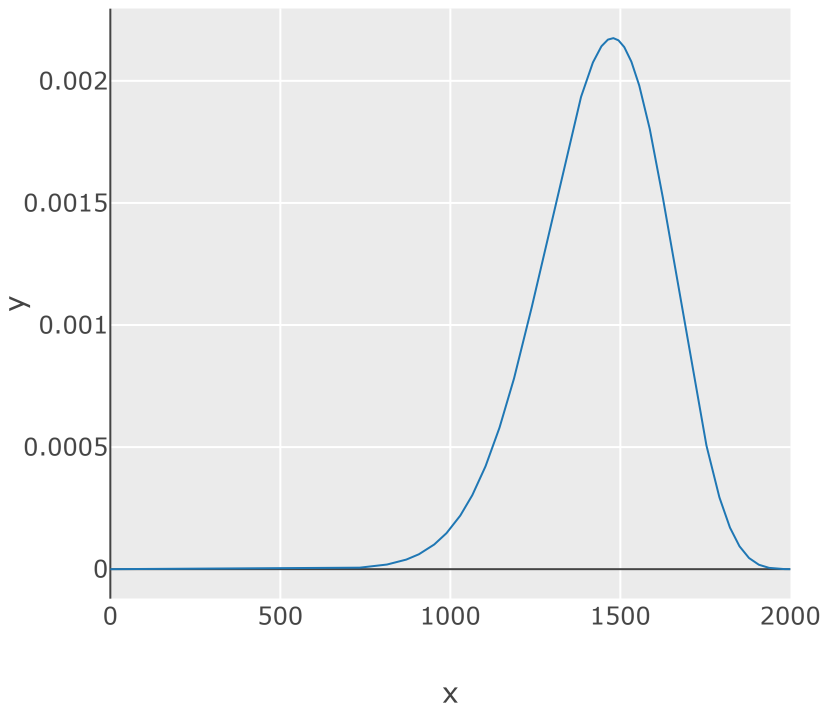

y
0.002
0.0015
0.001
0.0005
0
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

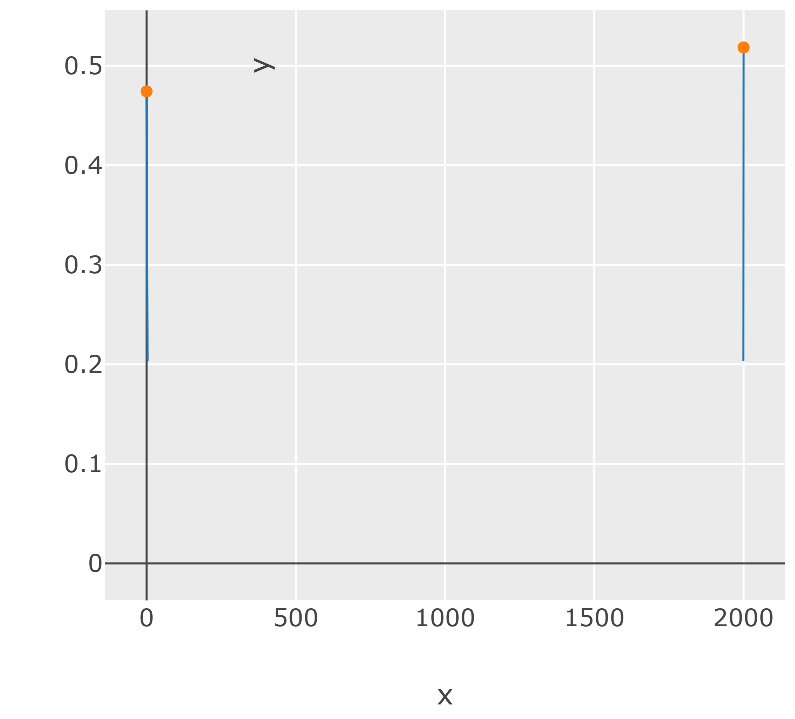

0.5

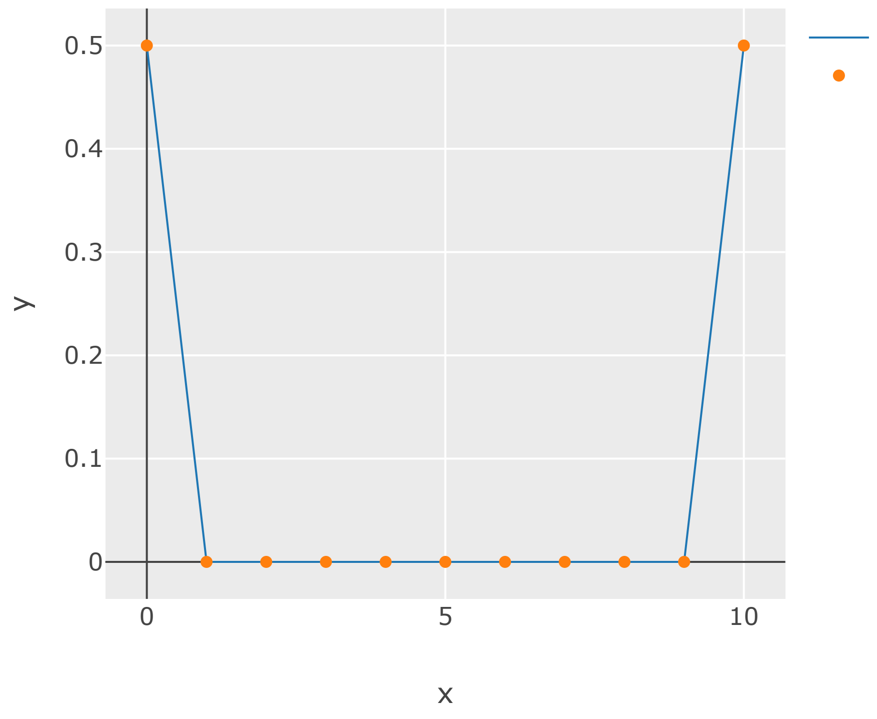


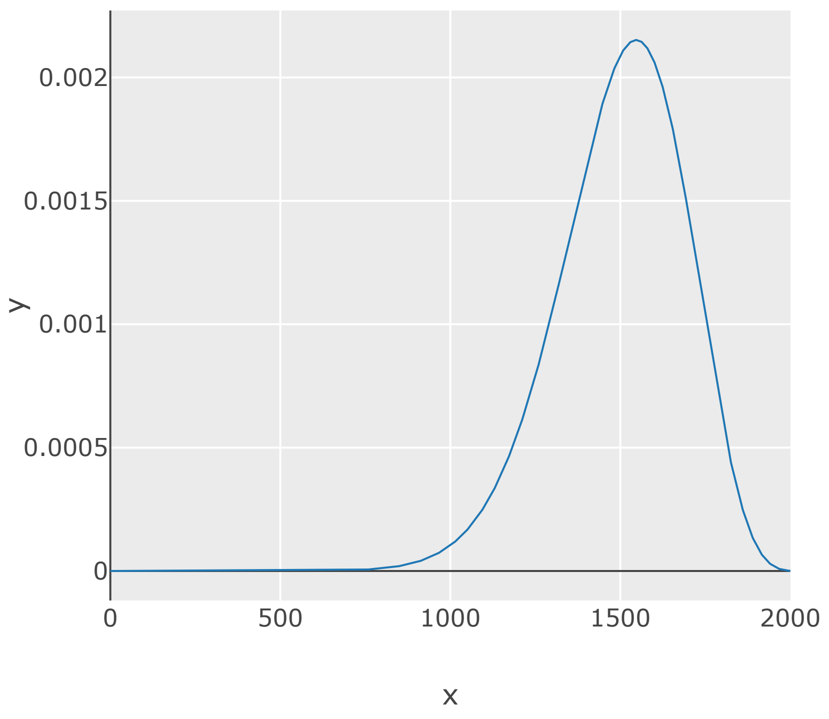
0.4
0.3
0.2
0.1
0
0
5
10
x



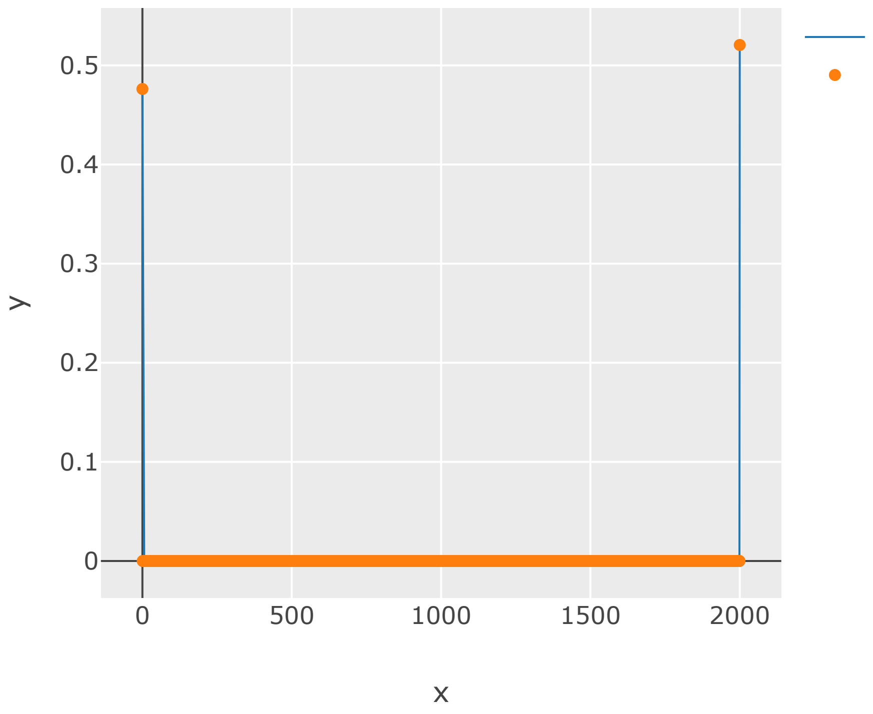
0.002
0.0015
0.001
0.0005
0
y
0
500
1000
1500
2000
x


y
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
x


y
0.5
0.4
0.3
0.2
0.1
0
0
5
10
x

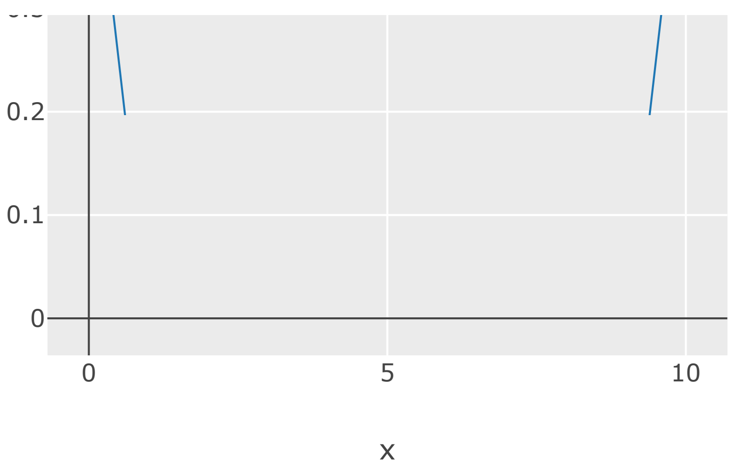


0.002
0.0015
0.001
0.0005
0
y
0
500
1000
1500
2000
x

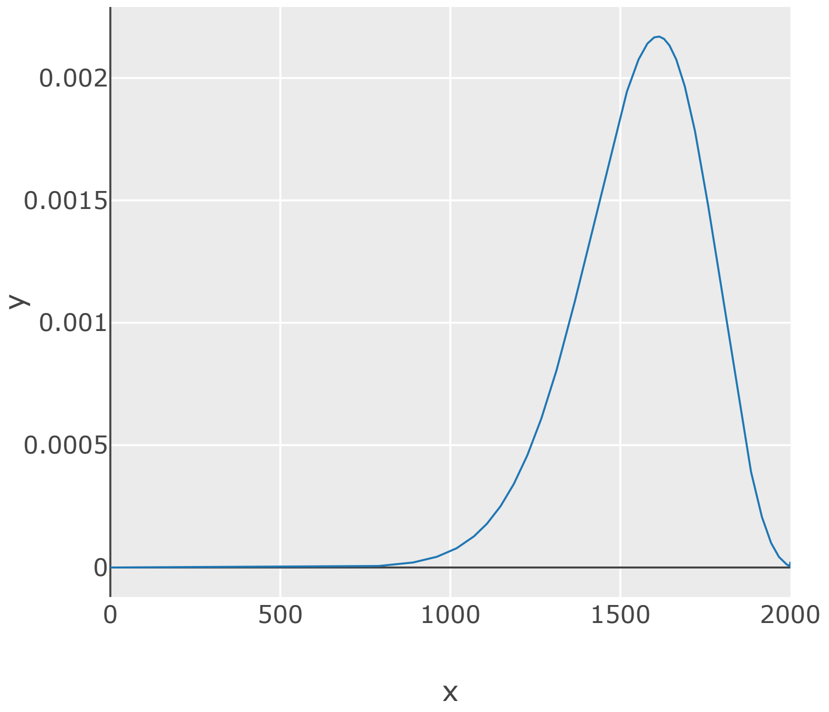


0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

0.5
0.4
0.3

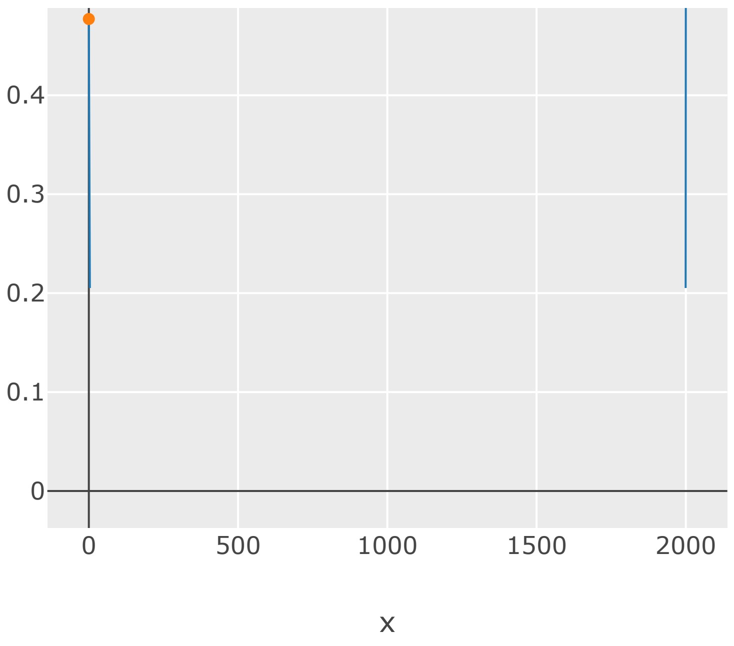


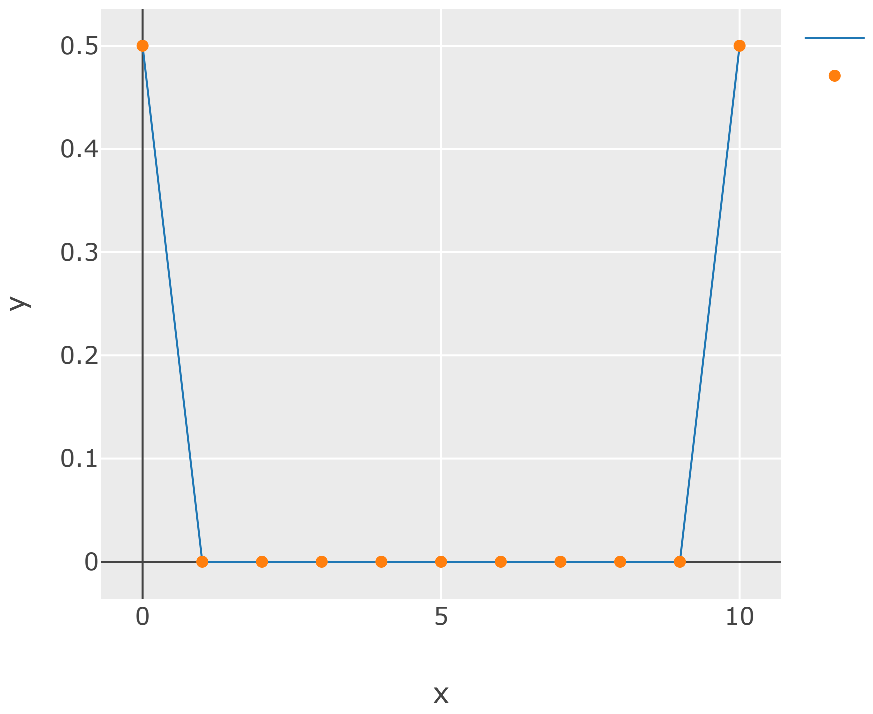
0.2
0.1
0
0
5
10
x



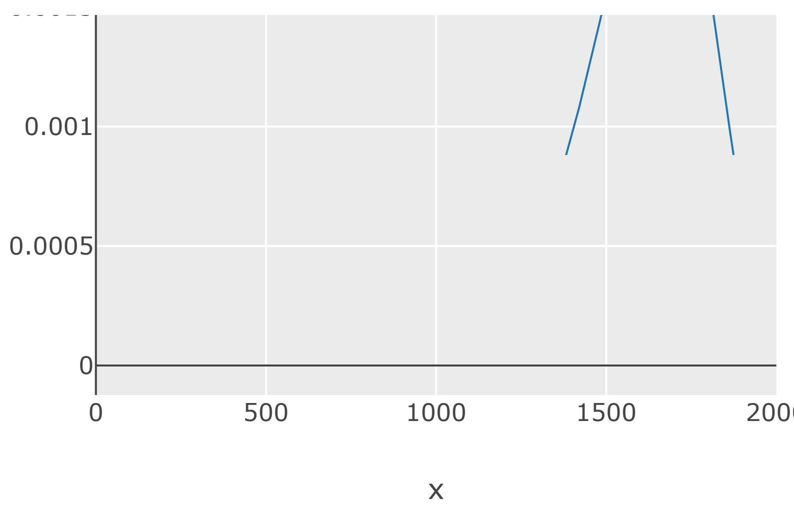
0.002
0.0015
0.001
0.0005
0
y
0
500
1000
1500
2000
x

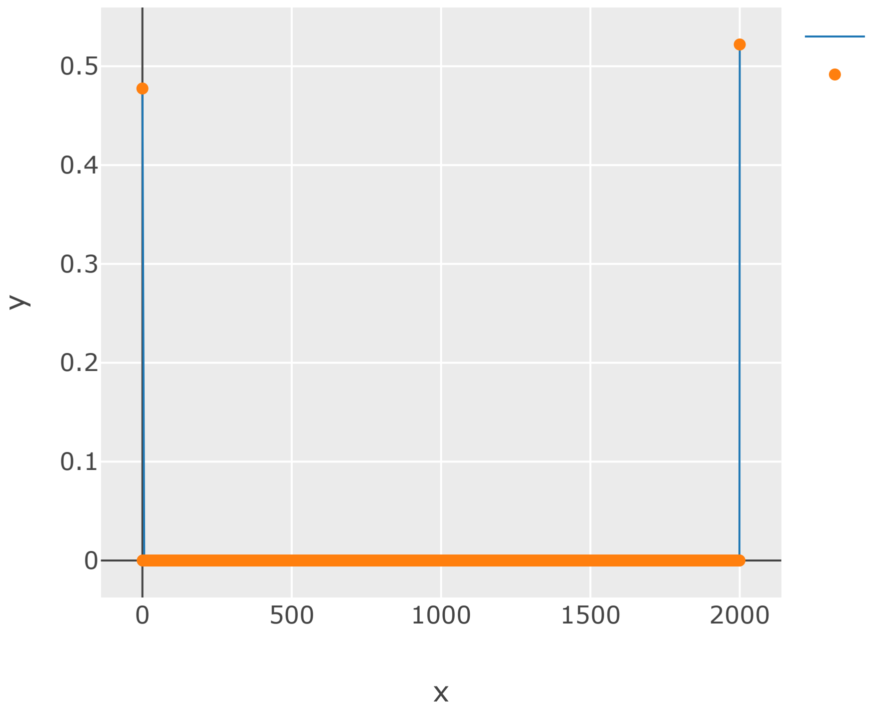
0.5


0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
x

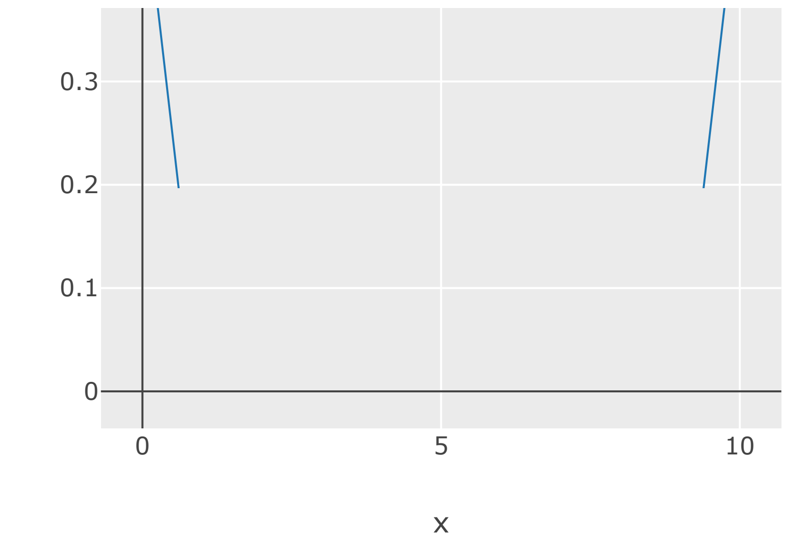

0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

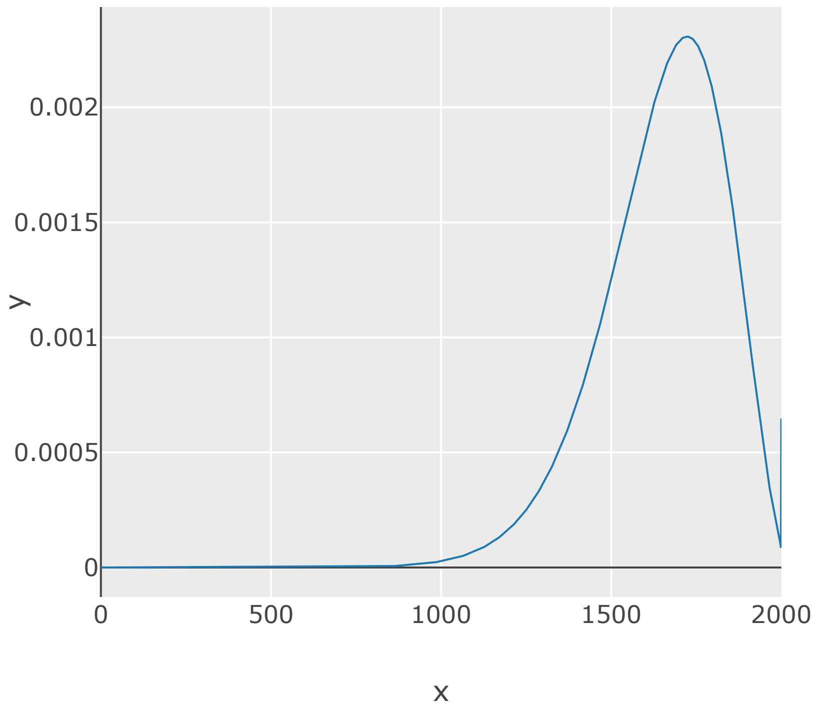
0.002
0.0015

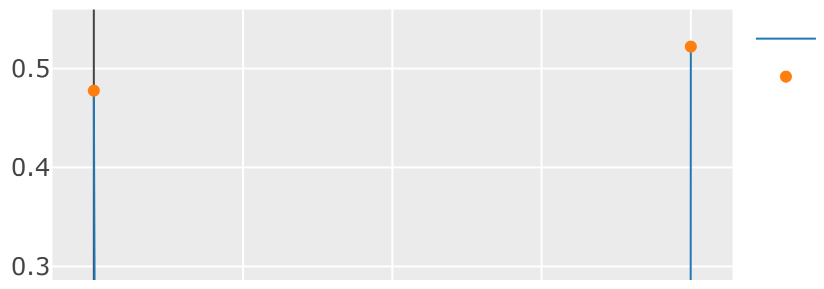

0.001
0.0005
0
0
500
1000
1500
2000
x

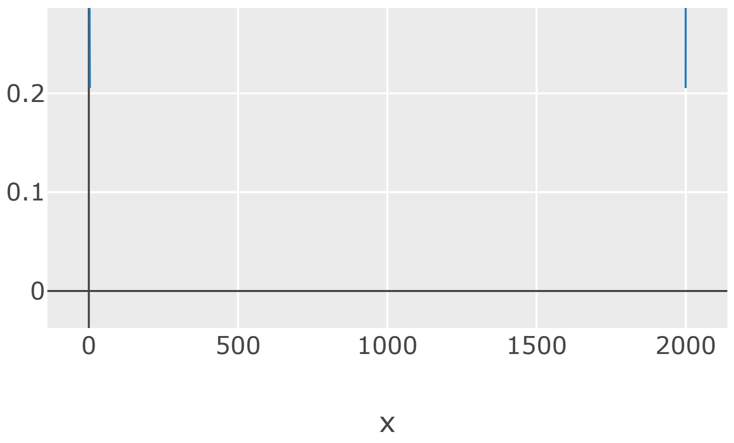

0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

0.5
0.4

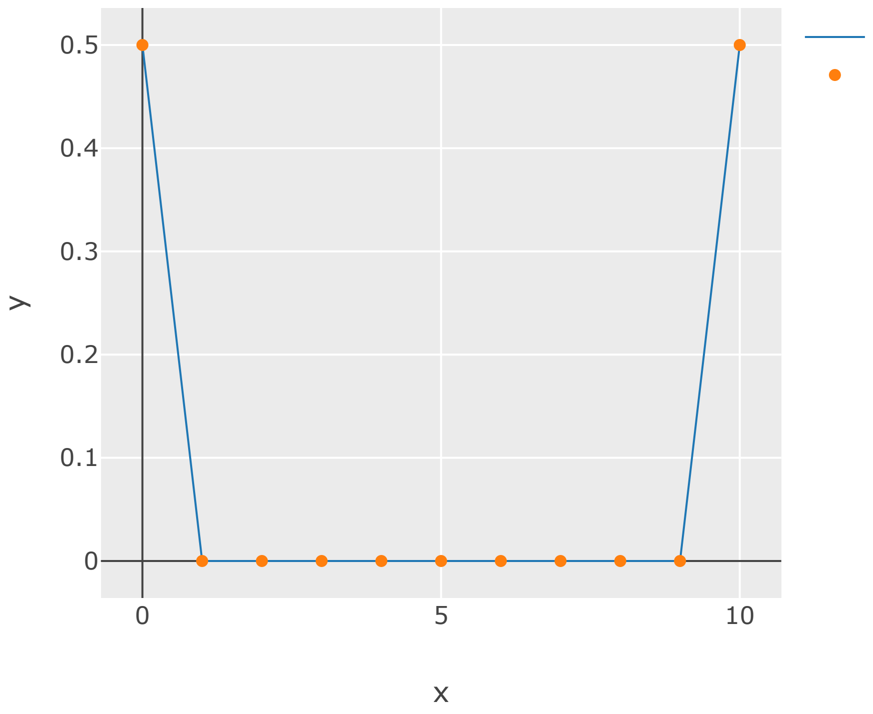

0.3
0.2
0.1
0
0
5
10
x


0.002
0.0015
0.001
0.0005
0
y
0
500
1000
1500
2000
x

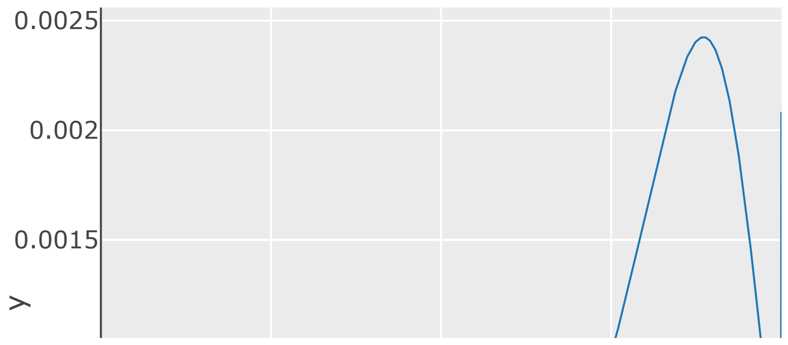

0.5
0.4
0.3



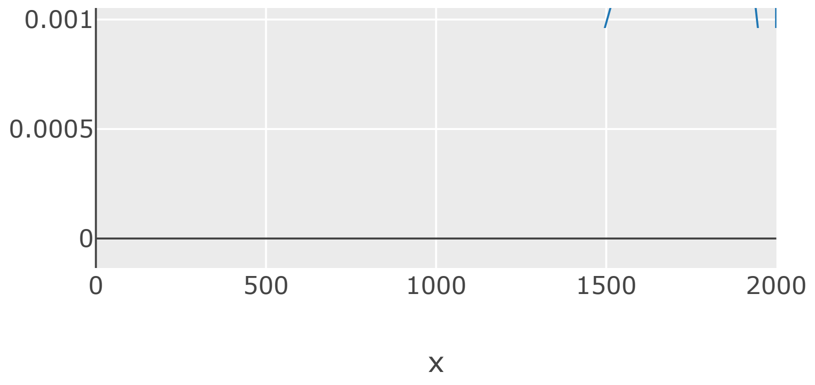
0.2
0.1
0
0
500
1000
1500
2000
x

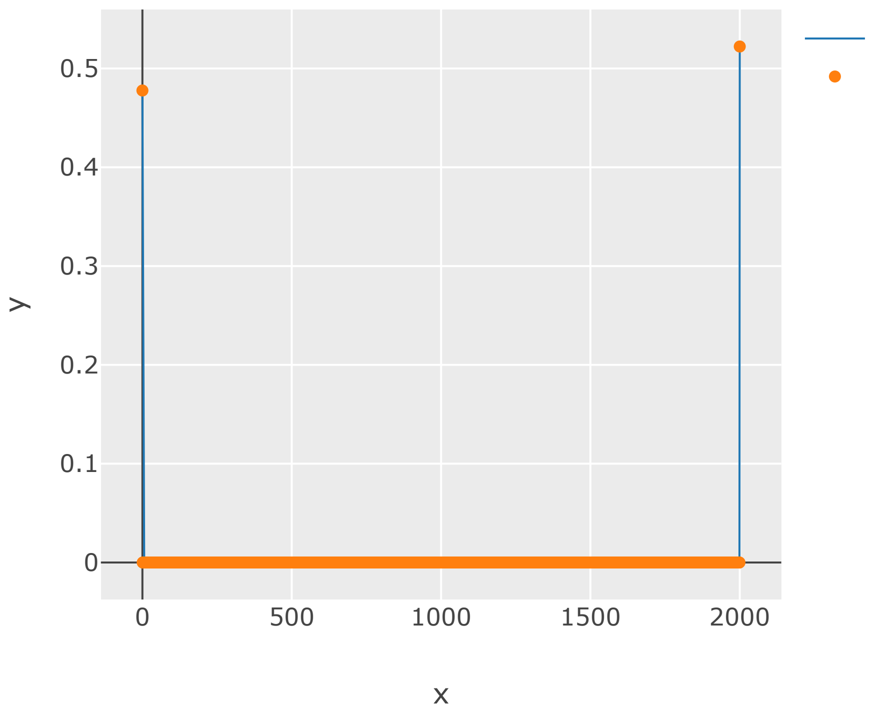


0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

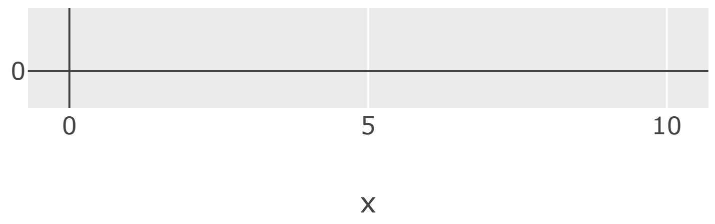


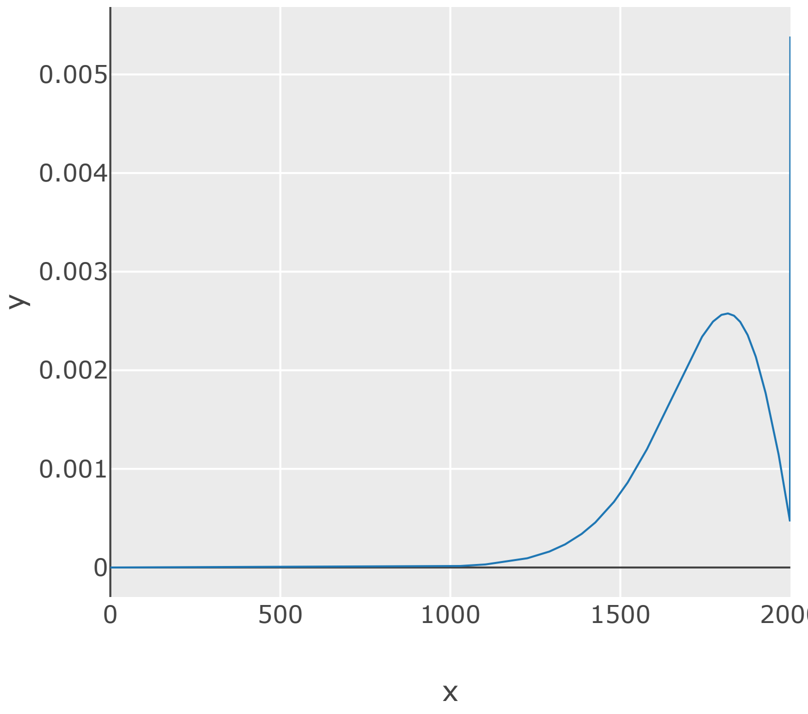
0.0025
0.002
0.0015
y

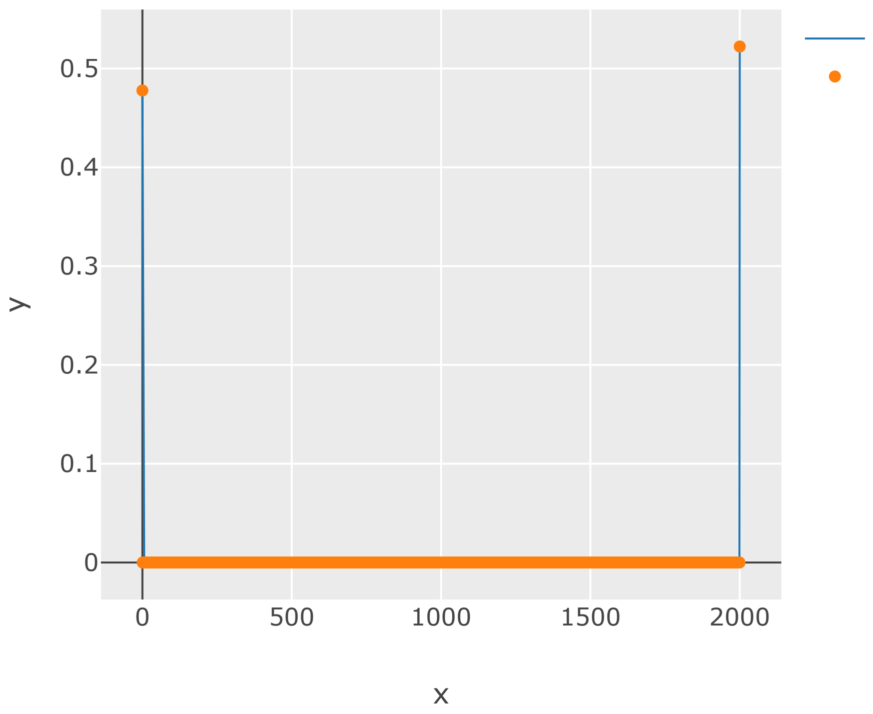

0.001
0.0005
0
0
500
1000
1500
2000
x

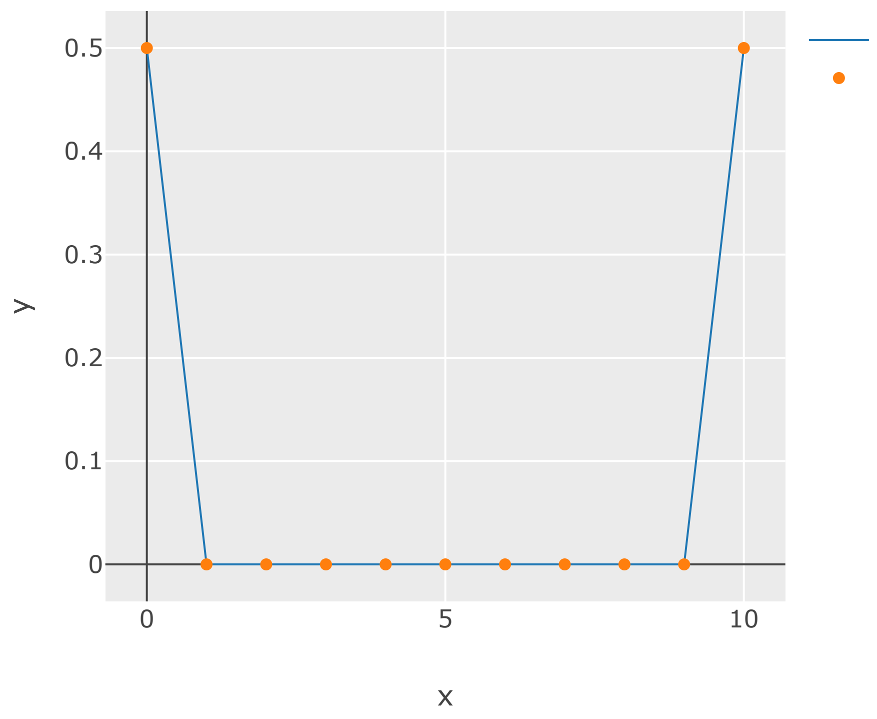

0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

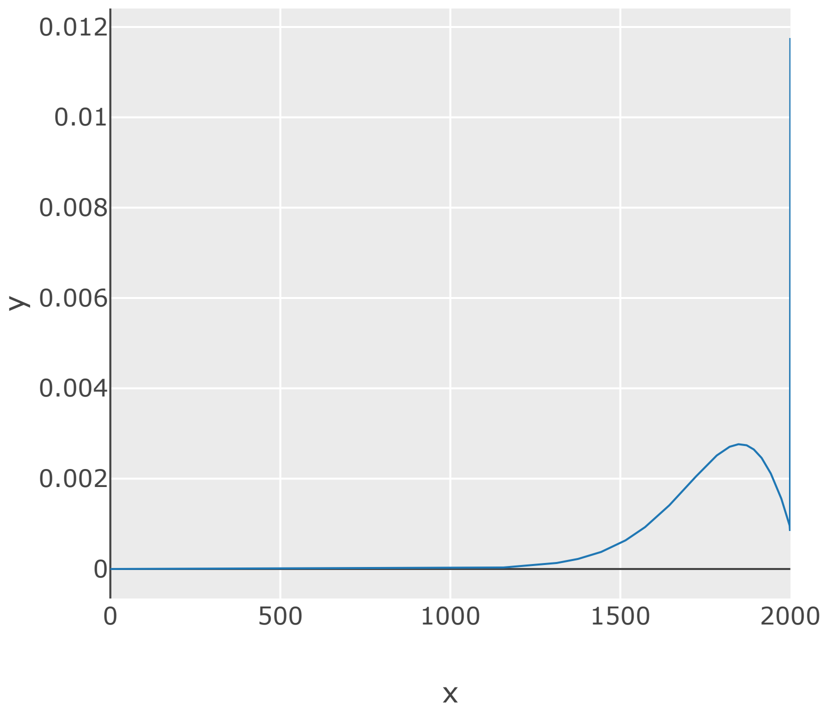
0.5
0.4
0.3
0.2
0.1
y

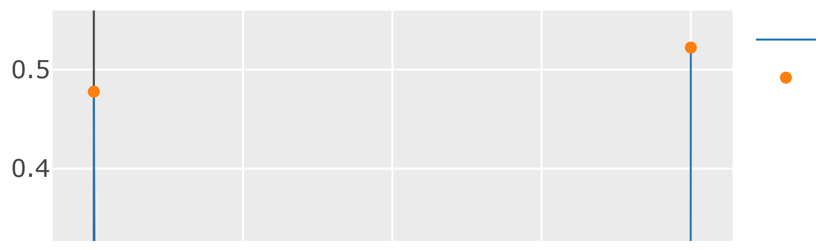

0
0
5
10
x

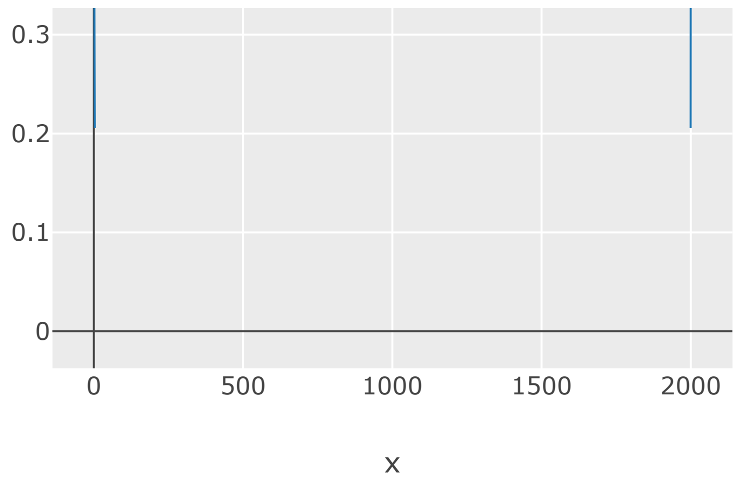

0.005
0.004
0.003
0.002
0.001
0
y
0
500
1000
1500
2000
x

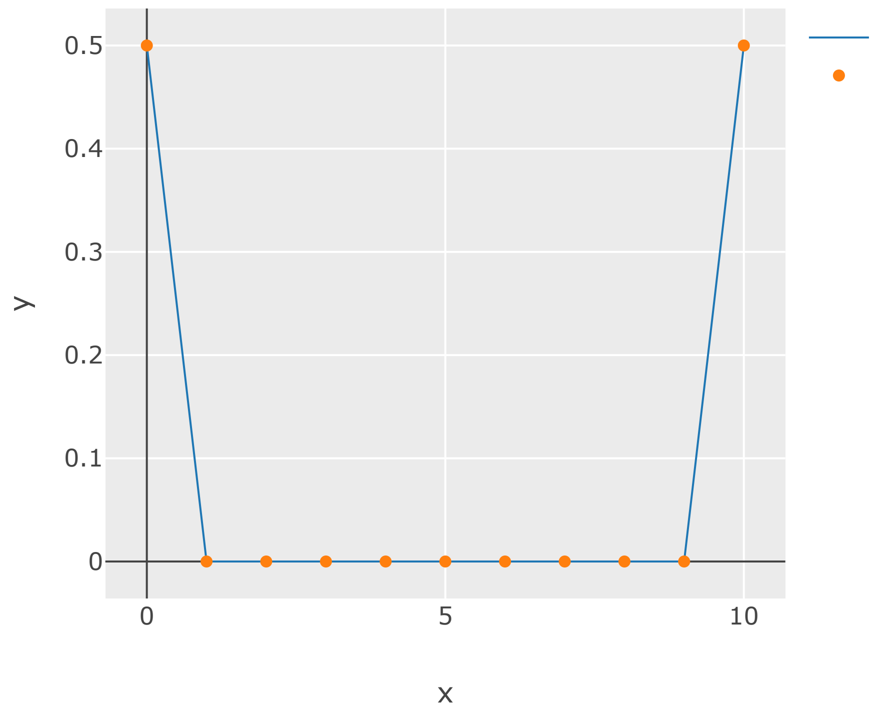

0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

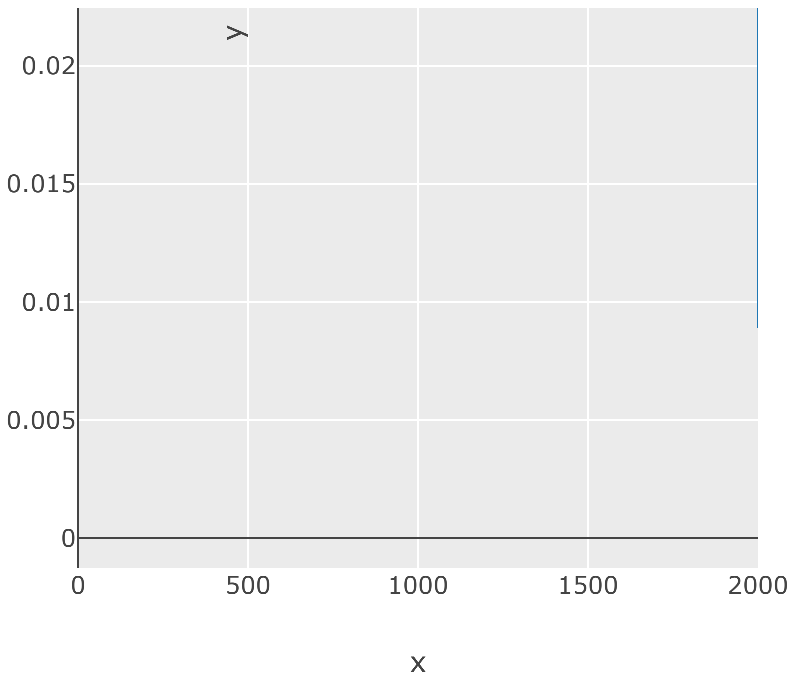

0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

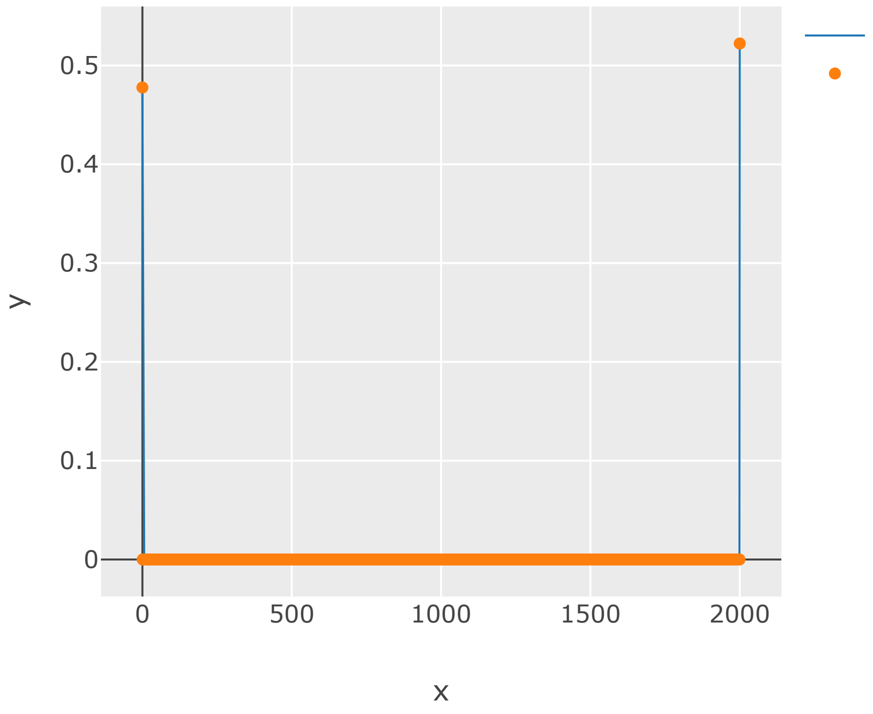

0.012
0.01
0.008
0.006
0.004
0.002
0
y
0
500
1000
1500
2000
x


0.5
0.4

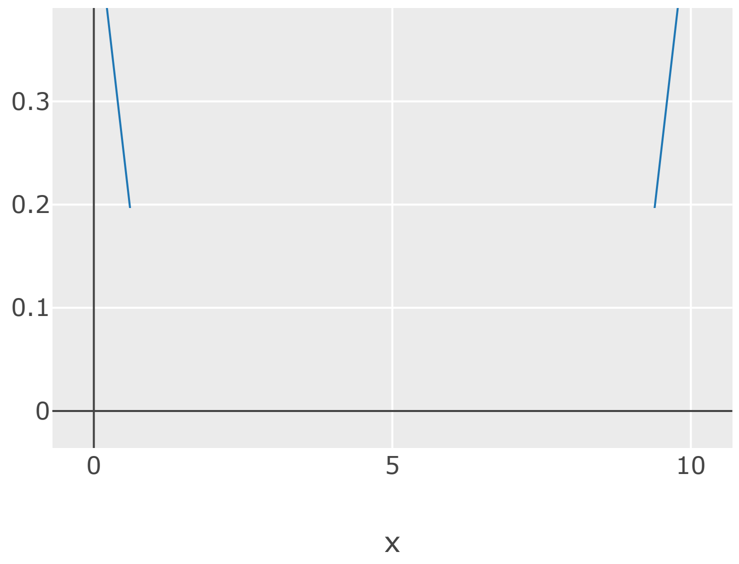

0.3
0.2
0.1
0
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

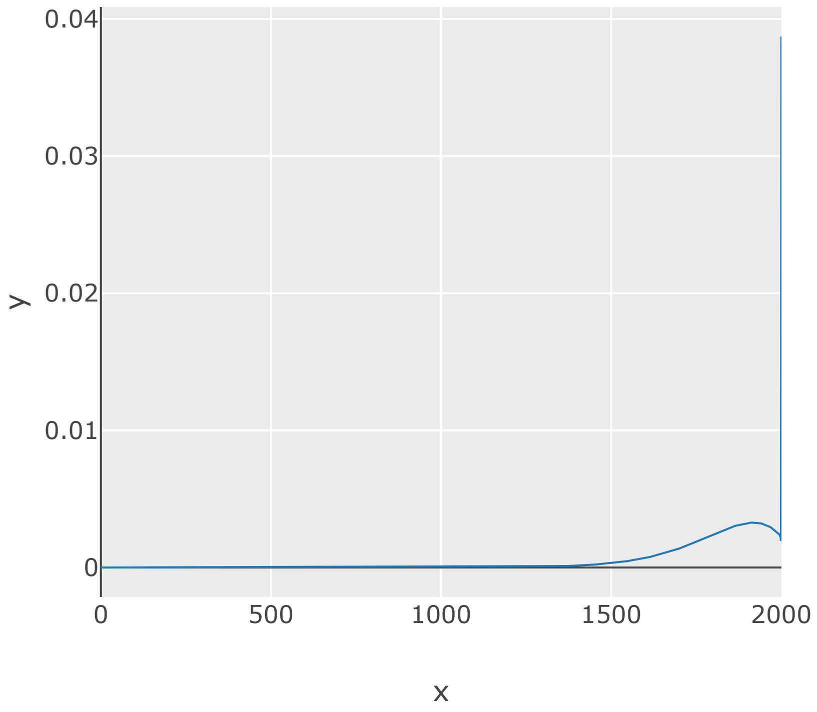

y
0.02
0.015
0.01
0.005
0
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x

0.5
0.4

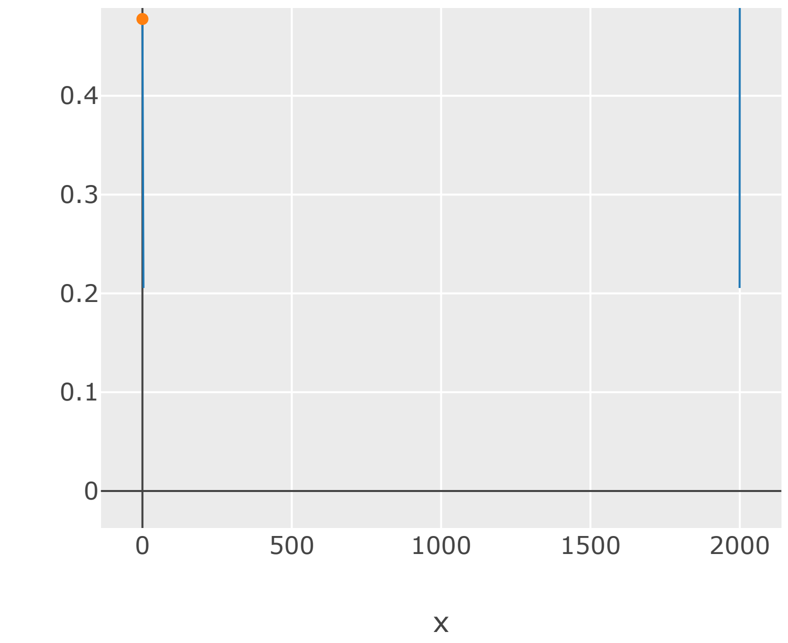


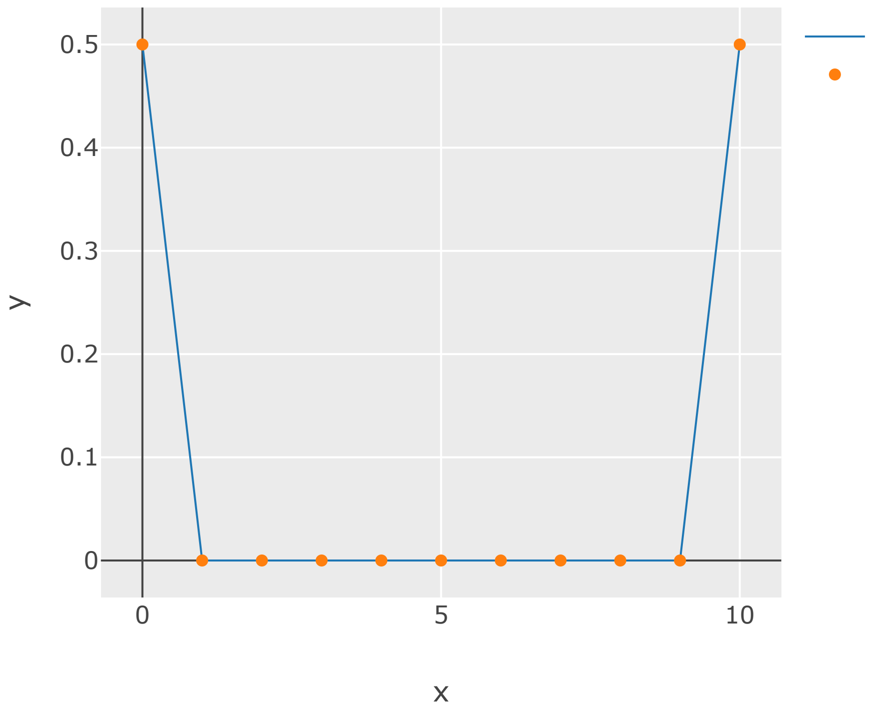
0.3
0.2
0.1
0
0
5
10
x

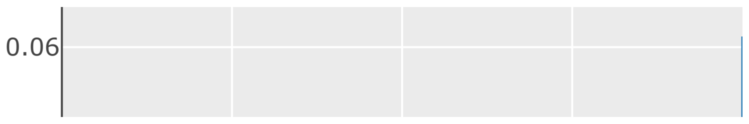


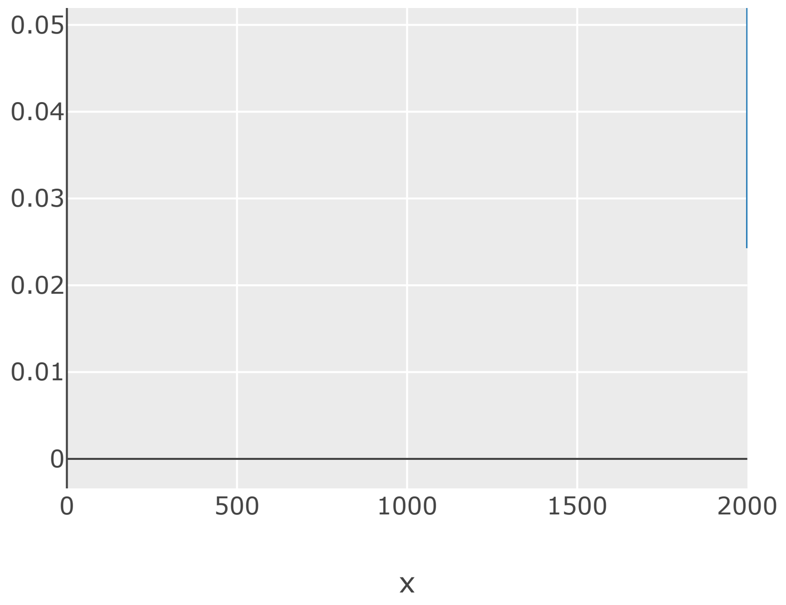
0.04
0.03
0.02
0.01
0
y
0
500
1000
1500
2000
x

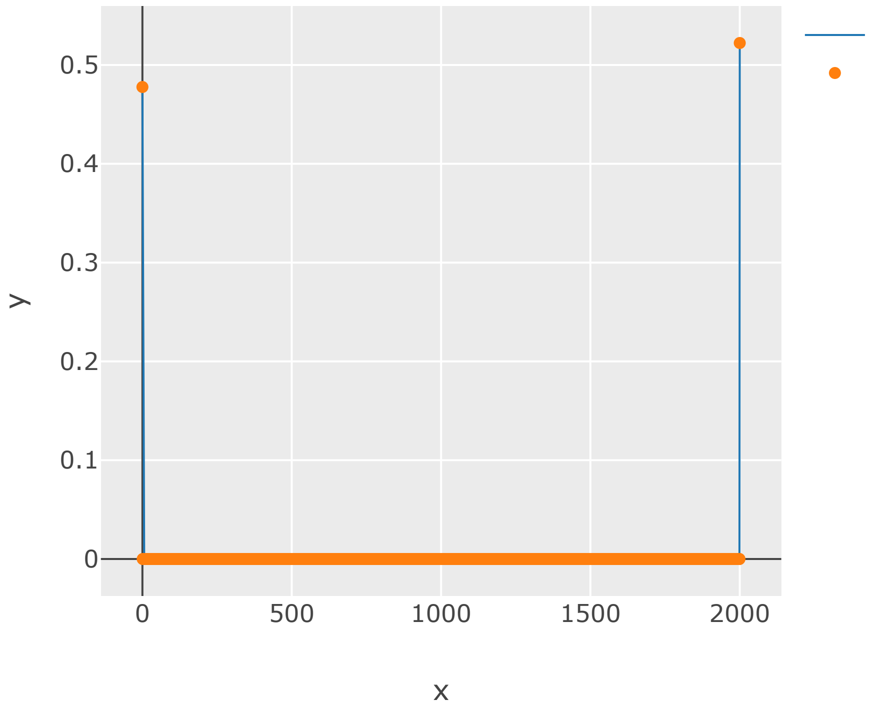
0.5

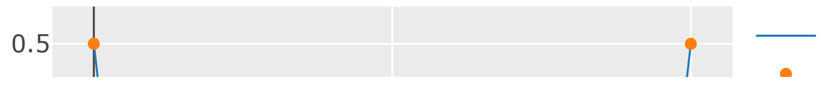

0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
x

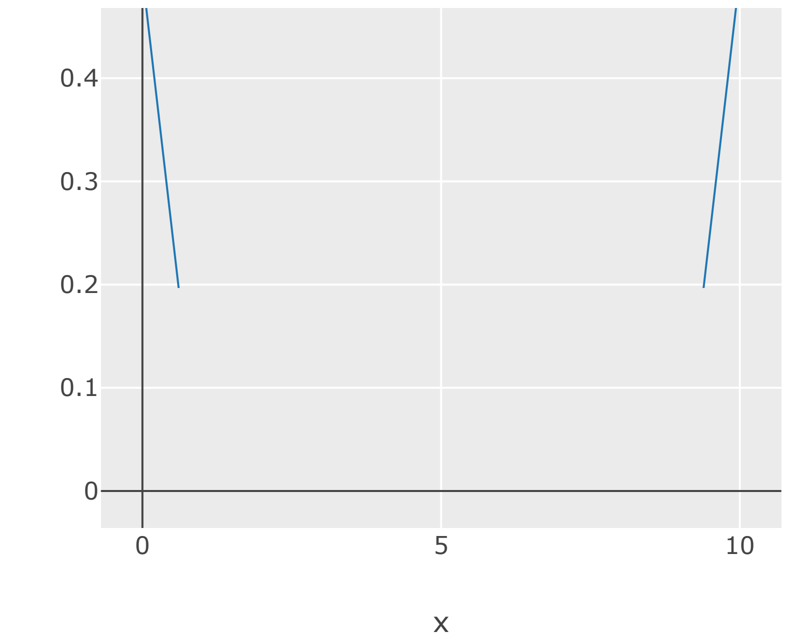

0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x

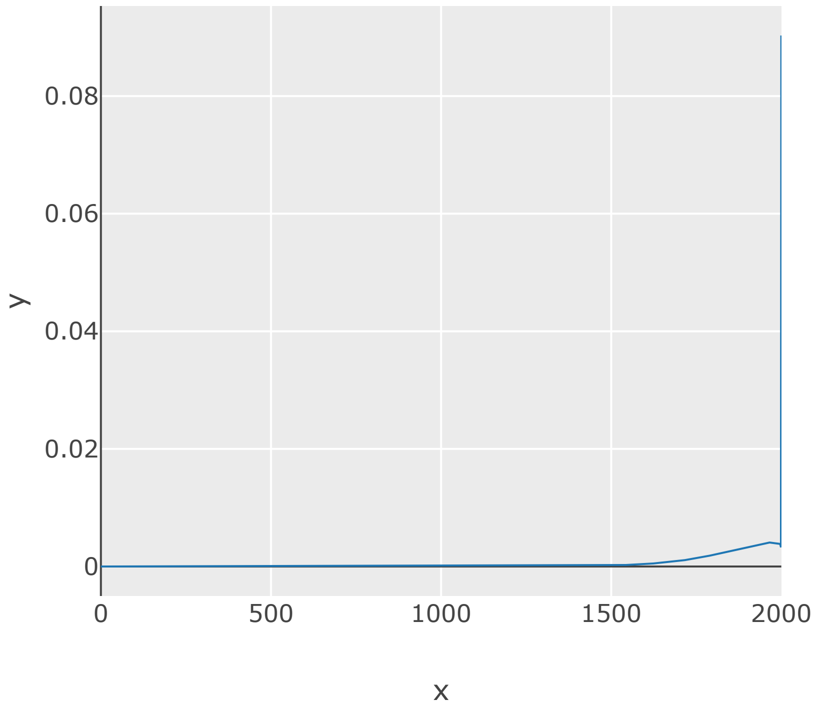

0.06


0.05
0.04
0.03
0.02
0.01
0
0
500
1000
1500
2000
x

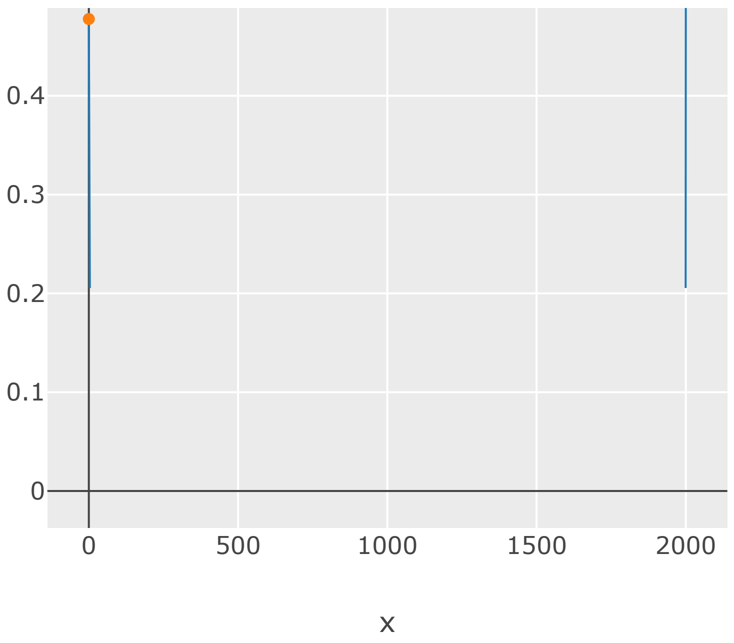

0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x


0.5

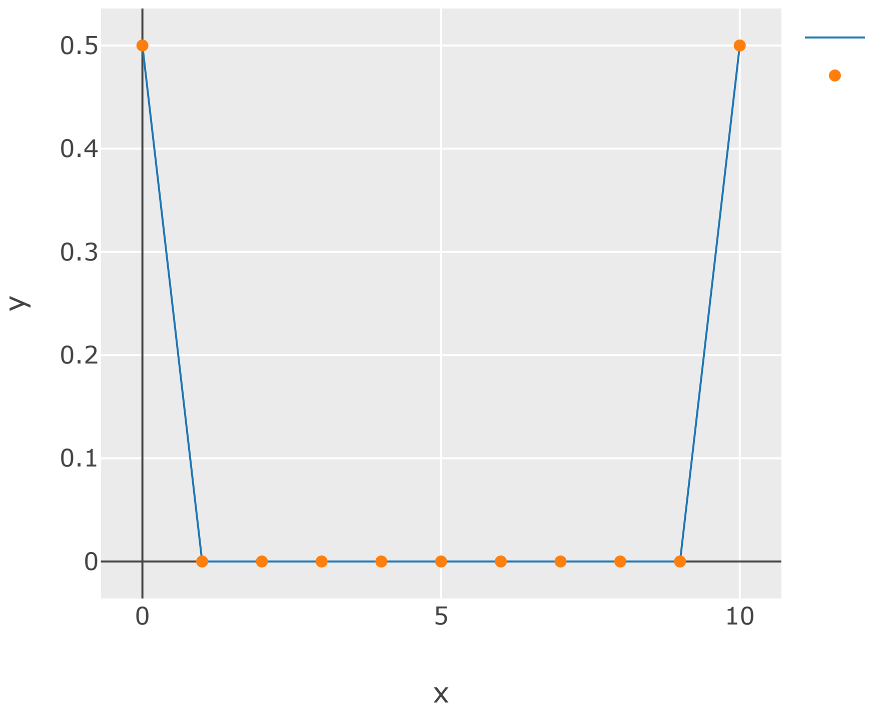

0.4
0.3
0.2
0.1
0
0
5
10
x


0.08
0.06
0.04
0.02
0
y
0
500
1000
1500
2000
x

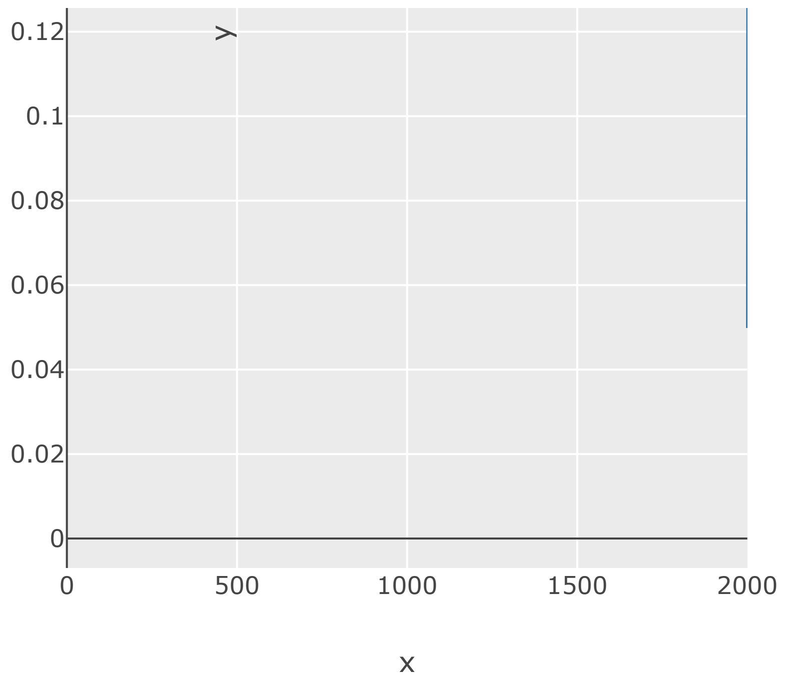
0.5

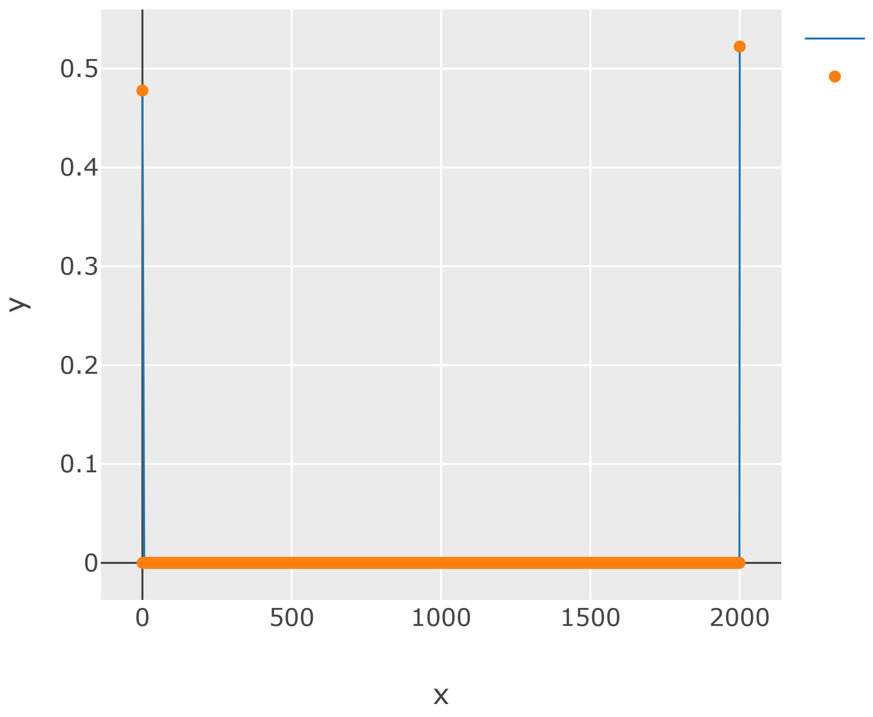


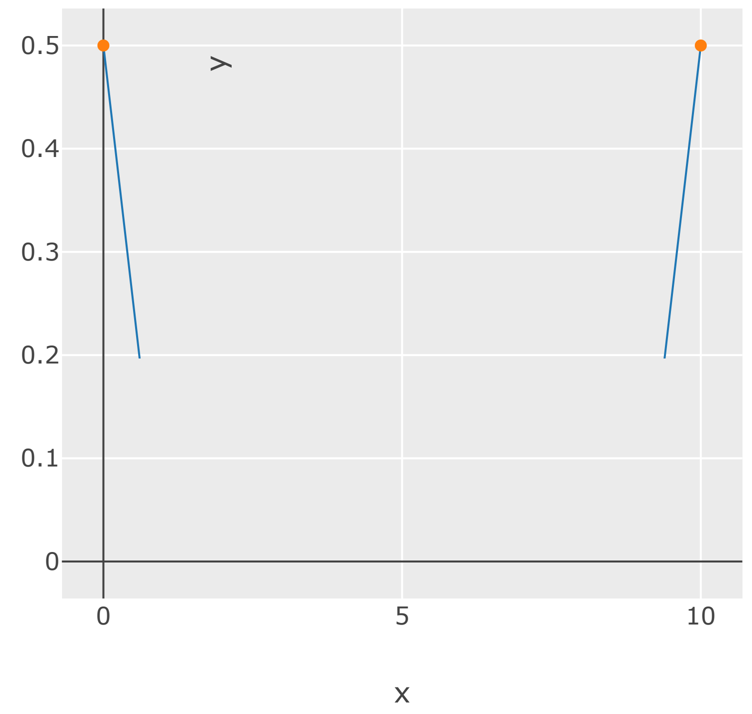
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
x

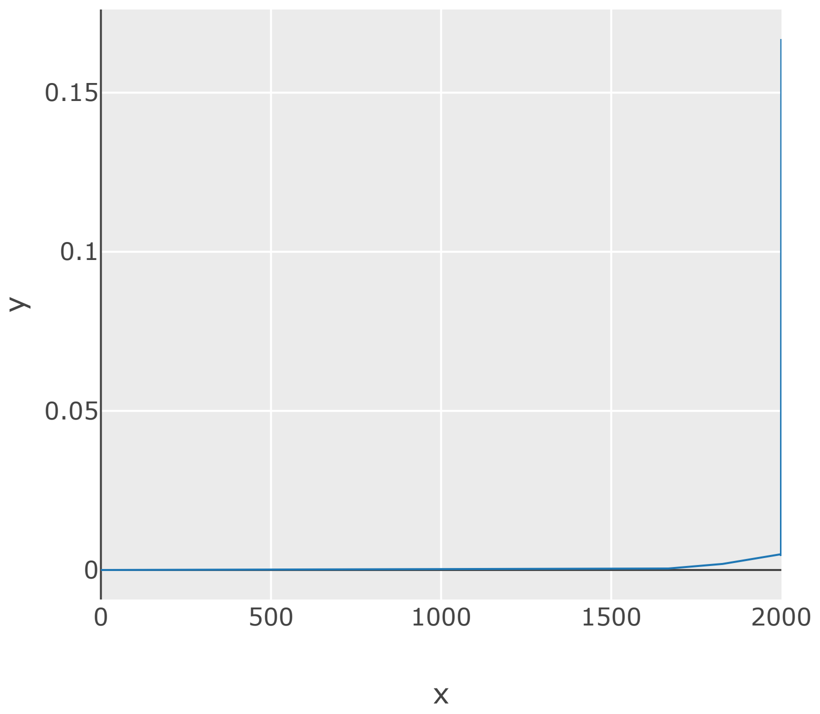


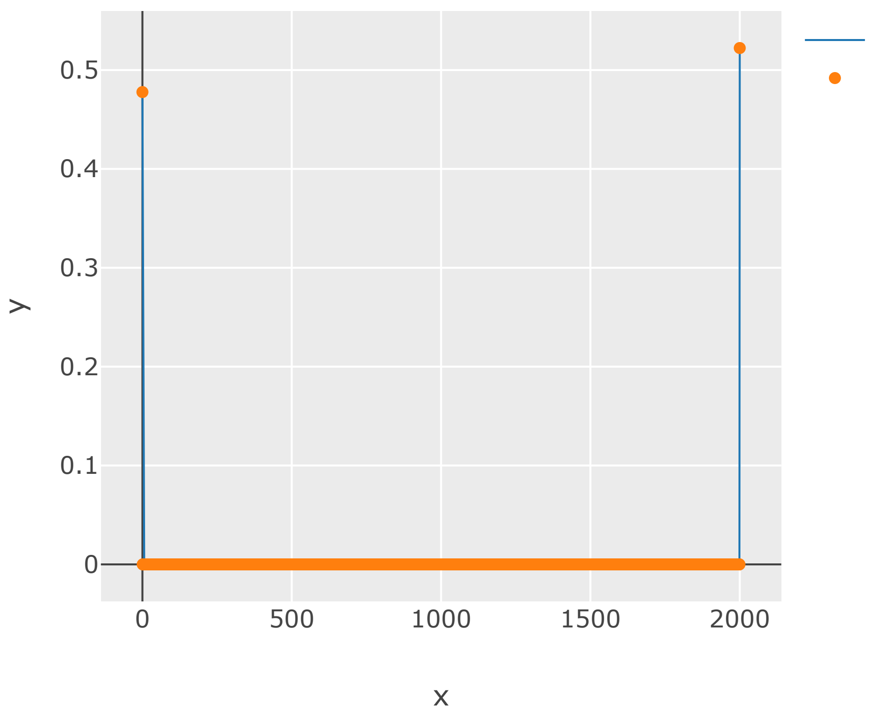
0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x


0.12
0.1
0.08
0.06
0.04
0.02
0
y
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
500
1000
1500
2000
x


0.5
0.4
0.3
0.2
0.1
0
y
0
5
10
x


0.15
0.1
0.05
0
y
0
500
1000
1500
2000
x



```
)
```

```
  (comment
    (fmat/shape predat-drift-mat)
    (fmat/shape reprod-drift-mat)
    (fmat/shape pred-reprod-mat)
    (map fmat/shape pred-reprod-tran-mats)
  )
```

```
source: src/clj/drifthappens/predreprod1.clj
```