Contributing to PyMC

A testimonial by Ricardo Vieira

Slides at
tinyurl.com/contributing-pymc

My intentions

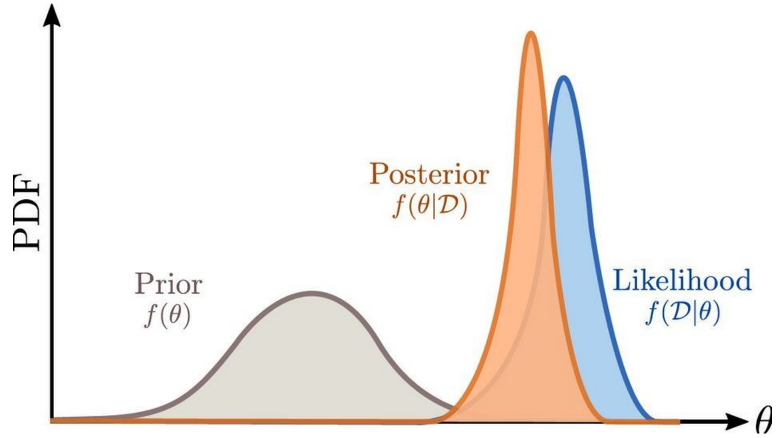
- Illustrate the Open Source collaboration experience
  - What it looks like when you are just starting
  - What it looks like for a regular contributor
- Share my biased views
  - It feels very random, you are unlikely to know where it leads
  - It is a great opportunity to learn
  - A lot of **fun** (hopefully)
- Invite you to give it a try!

PyMC

Probabilistic Programming in Python

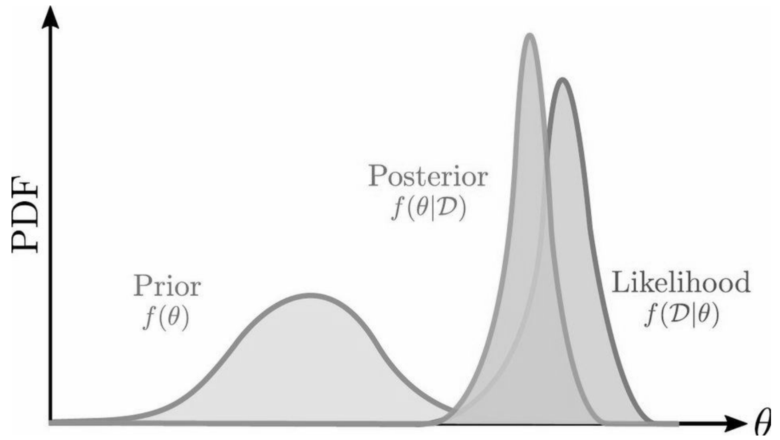
PyMC

Probabilistic Programming in Python



PyMC

Probabilistic Programming in Python

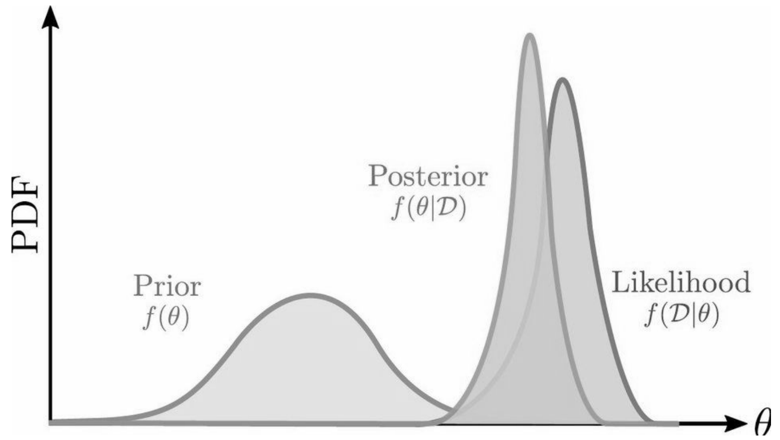


```python
import pymc as pm
import pandas as pd
```

```python
obs_data = pd.read_csv("my_observed_data.csv")

with pm.Model() as m:
    mean = pm.Normal("mean", 0, 1)
    noise = pm.HalfNormal("noise", 1)
    data = pm.Normal("data", mean, noise, observed=obs_data)

    posterior = pm.sample()
```

PyMC

Probabilistic Programming in Python



```python
import pymc as pm
import pandas as pd
```

```python
obs_data = pd.read_csv("my_observed_data.csv")

with pm.Model() as m:
    mean = pm.Normal("mean", 0, 1)
    noise = pm.HalfNormal("noise", 1)
    data = pm.Normal("data", mean, noise, observed=obs_data)

    posterior = pm.sample()
```

Learn more at: [docs.pymc.io/en/latest]
[Austin Rochford: Intro to Probabilistic Programming with PyMC]

	PyMC timeline	Personal timeline

	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school

	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)

	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)
2013	PyMC v3 initial release	Bachelors in Psychology, learns some old fashioned statistics

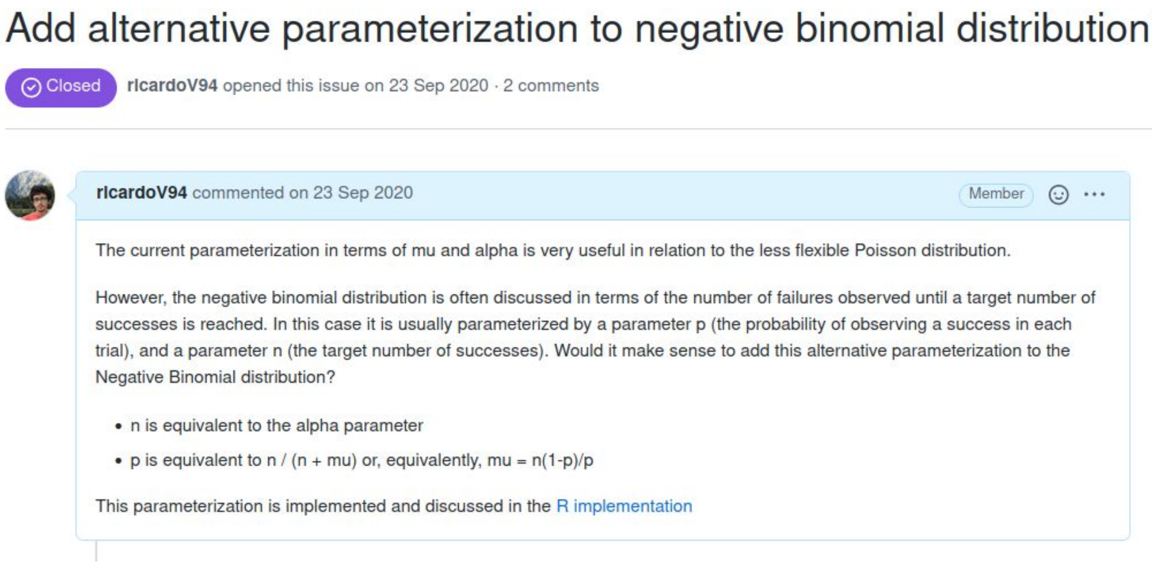
	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)
2013	PyMC v3 initial release	Bachelors in Psychology, learns some old fashioned statistics
2016	PyMC v3 paper published	Masters in Neuroscience, ports some Python2 code (err... adds parentheses to print statements)

	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)
2013	PyMC v3 initial release	Bachelors in Psychology, learns some old fashioned statistics
2016	PyMC v3 paper published	Masters in Neuroscience, ports some Python2 code (err... adds parentheses to print statements)
2018	TensorFlow backend alternative explored	Starts PhD in Cognitive Science, learns some edgy statistics

	PyMC timeline	**Personal timeline**
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)
2013	PyMC v3 initial release	Bachelors in Psychology, learns some old fashioned statistics
2016	PyMC v3 paper published	Masters in Neuroscience, ports some Python2 code (err... adds parentheses to print statements)
2018	TensorFlow backend alternative explored	Starts PhD in Cognitive Science, learns some edgy statistics
2020	PyMC developers decide to maintain Theano	First contributions to PyMC

	PyMC timeline	Personal timeline
2003	PyMC v1 development starts	Plays Pokemon in primary school
2010	PyMC v2 paper published	Dabbles in Javascript during high school (err... watches youtube tutorials)
2013	PyMC v3 initial release	Bachelors in Psychology, learns some old fashioned statistics
2016	PyMC v3 paper published	Masters in Neuroscience, ports some Python2 code (err... adds parentheses to print statements)
2018	TensorFlow backend alternative explored	Starts PhD in Cognitive Science, learns some edgy statistics
2020	PyMC developers decide to fork Theano	First contributions to PyMC
2021	PyMC v4 beta is released	Becomes core developer Participates in Google Summer of Code Quits PhD Gets financial support from Chan Zuckerberg Initiative (Via NumFocus)

An innocent feature request



Add alternative parameterization to negative binomial distribution

Closed rlcardoV94 opened this issue on 23 Sep 2020 · 2 comments

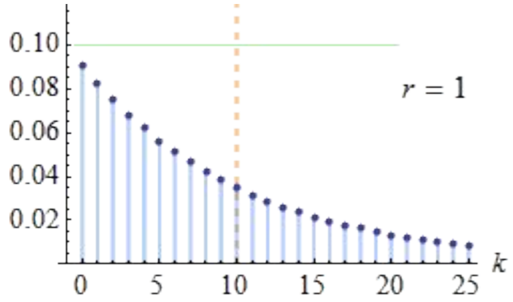
rlcardoV94 commented on 23 Sep 2020 Member

The current parameterization in terms of mu and alpha is very useful in relation to the less flexible Poisson distribution.

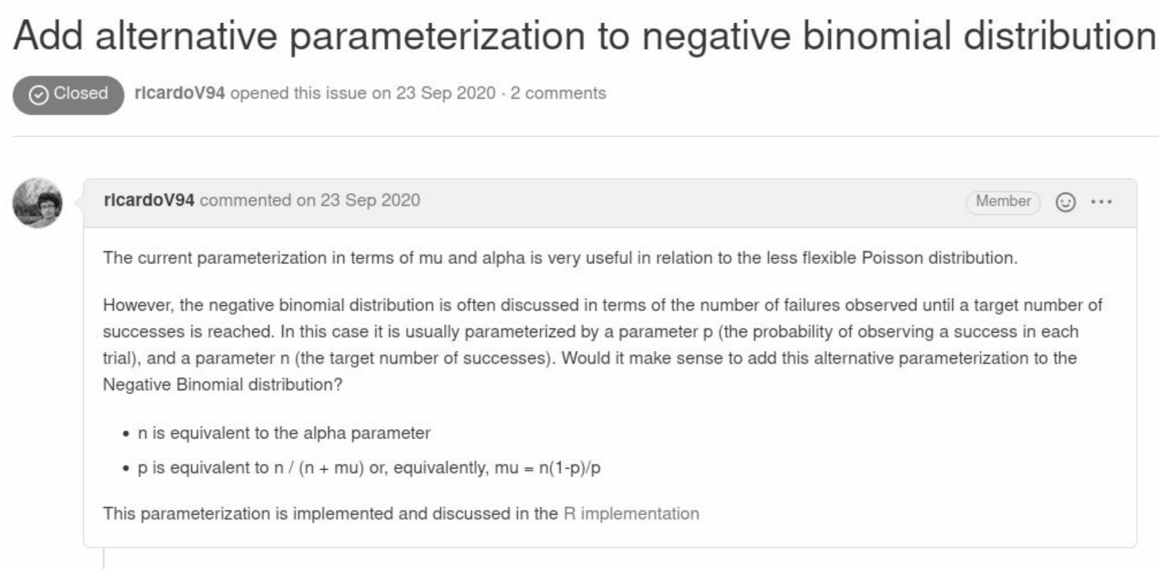
However, the negative binomial distribution is often discussed in terms of the number of failures observed until a target number of successes is reached. In this case it is usually parameterized by a parameter p (the probability of observing a success in each trial), and a parameter n (the target number of successes). Would it make sense to add this alternative parameterization to the Negative Binomial distribution?

- n is equivalent to the alpha parameter
- p is equivalent to n / (n + mu) or, equivalently, mu = n(1-p)/p

This parameterization is implemented and discussed in the R implementation



An innocent feature request



Add alternative parameterization to negative binomial distribution

Closed ricardoV94 opened this issue on 23 Sep 2020 · 2 comments

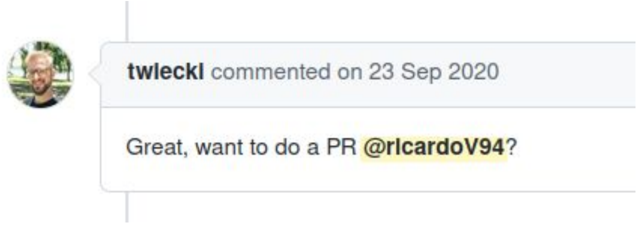
ricardoV94 commented on 23 Sep 2020 Member

The current parameterization in terms of mu and alpha is very useful in relation to the less flexible Poisson distribution.

However, the negative binomial distribution is often discussed in terms of the number of failures observed until a target number of successes is reached. In this case it is usually parameterized by a parameter p (the probability of observing a success in each trial), and a parameter n (the target number of successes). Would it make sense to add this alternative parameterization to the Negative Binomial distribution?

- n is equivalent to the alpha parameter
- p is equivalent to n / (n + mu) or, equivalently, mu = n(1-p)/p

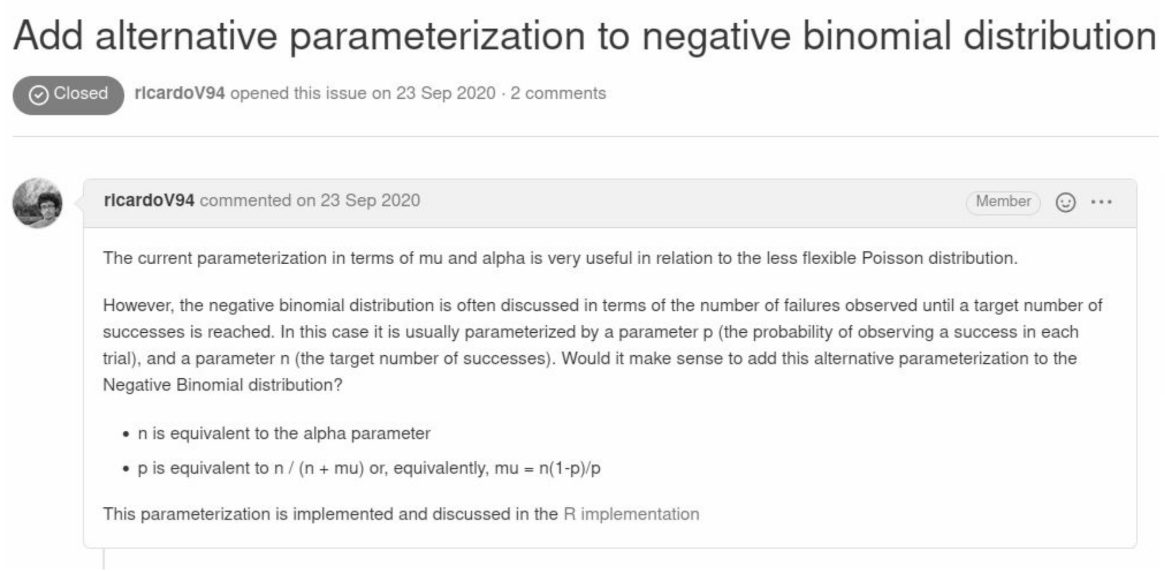
This parameterization is implemented and discussed in the R implementation



twiecki commented on 23 Sep 2020

Great, want to do a PR @ricardoV94?

An innocent feature request



Add alternative parameterization to negative binomial distribution

Closed ricardoV94 opened this issue on 23 Sep 2020 · 2 comments

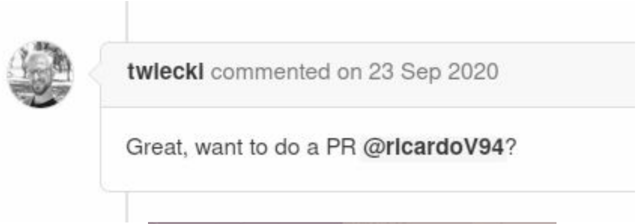
ricardoV94 commented on 23 Sep 2020 Member

The current parameterization in terms of mu and alpha is very useful in relation to the less flexible Poisson distribution.

However, the negative binomial distribution is often discussed in terms of the number of failures observed until a target number of successes is reached. In this case it is usually parameterized by a parameter p (the probability of observing a success in each trial), and a parameter n (the target number of successes). Would it make sense to add this alternative parameterization to the Negative Binomial distribution?

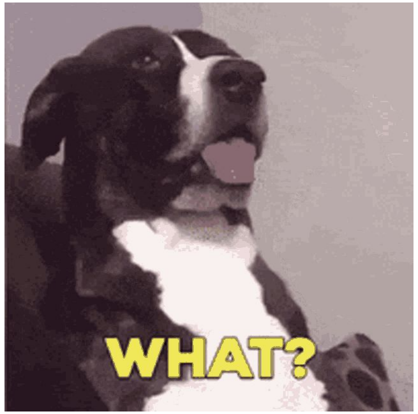
- n is equivalent to the alpha parameter
- p is equivalent to n / (n + mu) or, equivalently, mu = n(1-p)/p

This parameterization is implemented and discussed in the R implementation

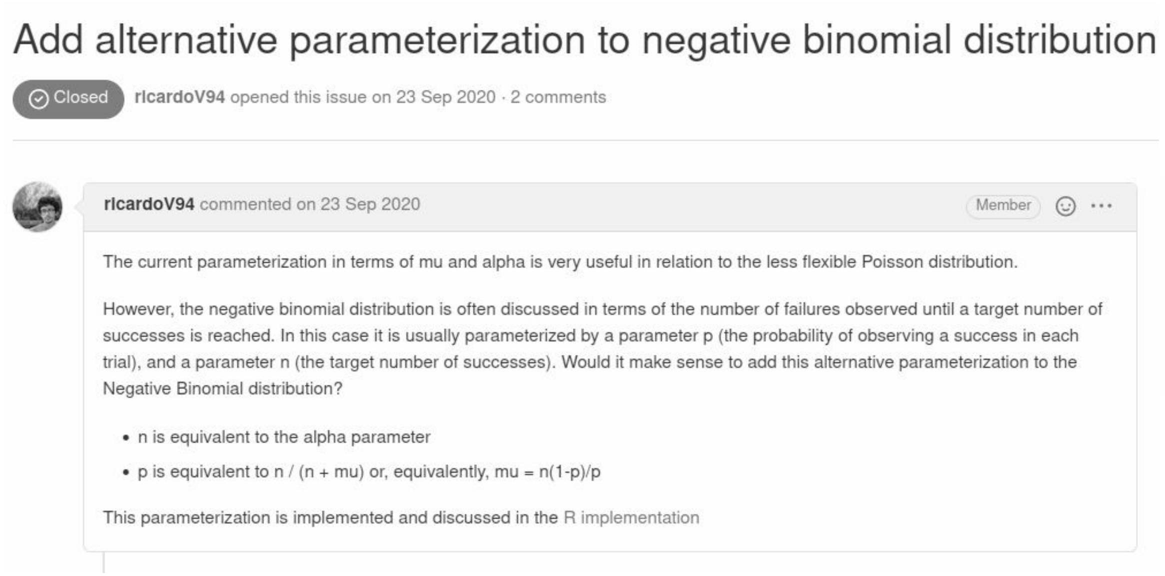


twiecki commented on 23 Sep 2020

Great, want to do a PR @ricardoV94?



An innocent feature request



Add alternative parameterization to negative binomial distribution

Closed ricardoV94 opened this issue on 23 Sep 2020 · 2 comments

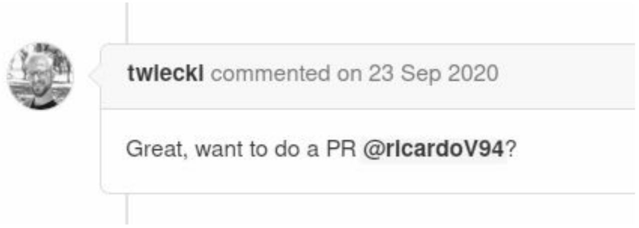
ricardoV94 commented on 23 Sep 2020 Member

The current parameterization in terms of mu and alpha is very useful in relation to the less flexible Poisson distribution.

However, the negative binomial distribution is often discussed in terms of the number of failures observed until a target number of successes is reached. In this case it is usually parameterized by a parameter p (the probability of observing a success in each trial), and a parameter n (the target number of successes). Would it make sense to add this alternative parameterization to the Negative Binomial distribution?

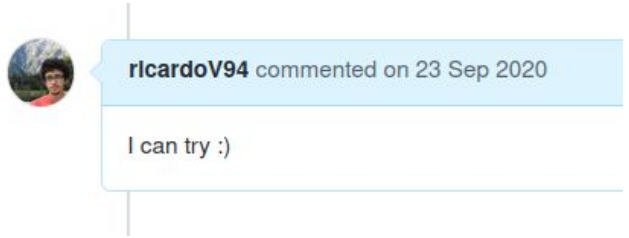
- n is equivalent to the alpha parameter
- p is equivalent to n / (n + mu) or, equivalently, mu = n(1-p)/p

This parameterization is implemented and discussed in the R implementation



twiecki commented on 23 Sep 2020

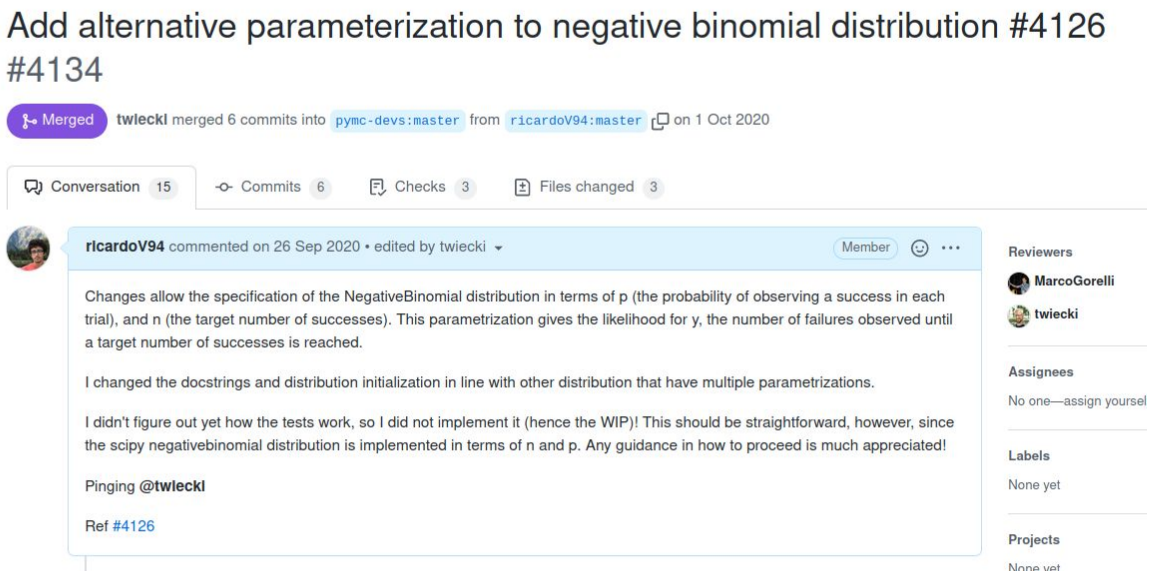
Great, want to do a PR @ricardoV94?



ricardoV94 commented on 23 Sep 2020

I can try :)

What does a (first) PR look like?



Add alternative parameterization to negative binomial distribution #4126 #4134

Merged twiecki merged 6 commits into `pymc-devs:master` from `ricardoV94:master` on 1 Oct 2020

ricardoV94 commented on 26 Sep 2020 • edited by twiecki

Changes allow the specification of the NegativeBinomial distribution in terms of p (the probability of observing a success in each trial), and n (the target number of successes). This parametrization gives the likelihood for y, the number of failures observed until a target number of successes is reached.

I changed the docstrings and distribution initialization in line with other distribution that have multiple parametrizations.

I didn't figure out yet how the tests work, so I did not implement it (hence the WIP)! This should be straightforward, however, since the scipy negativebinomial distribution is implemented in terms of n and p. Any guidance in how to proceed is much appreciated!

Pinging **@twiecki**

Ref #4126

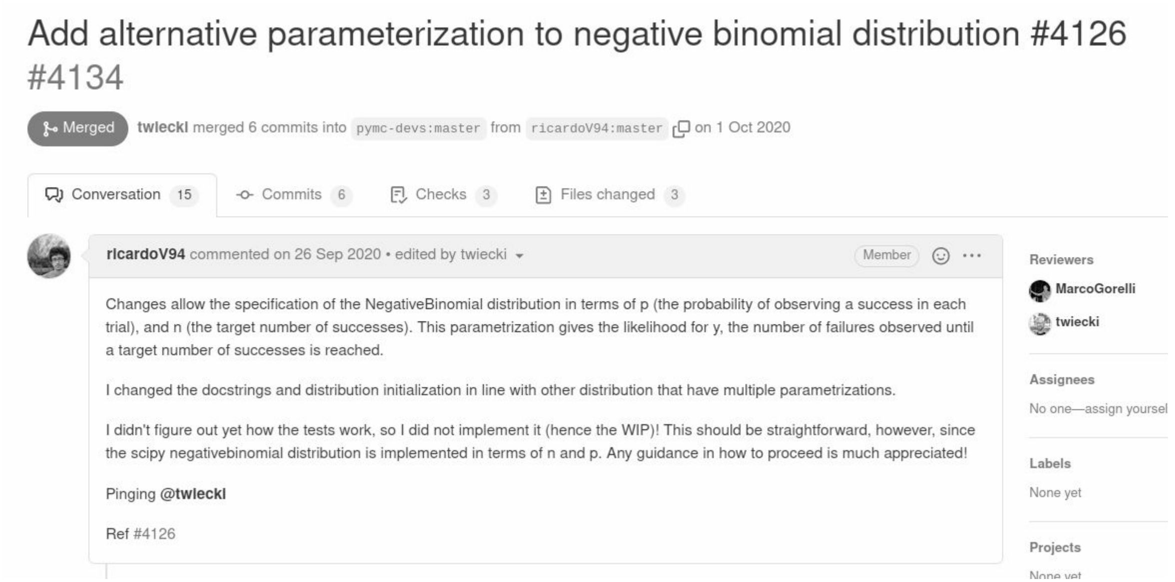
Reviewers
MarcoGorelli
twiecki

Assignees
No one—assign yoursel

Labels
None yet

Projects
None yet

What does a (first) PR look like?



Add alternative parameterization to negative binomial distribution #4126 #4134

twiecki merged 6 commits into `pymc-devs:master` from `ricardoV94:master` on 1 Oct 2020

ricardoV94 commented on 26 Sep 2020 • edited by twiecki

Changes allow the specification of the NegativeBinomial distribution in terms of p (the probability of observing a success in each trial), and n (the target number of successes). This parametrization gives the likelihood for y, the number of failures observed until a target number of successes is reached.

I changed the docstrings and distribution initialization in line with other distribution that have multiple parametrizations.

I didn't figure out yet how the tests work, so I did not implement it (hence the WIP)! This should be straightforward, however, since the scipy negativebinomial distribution is implemented in terms of n and p. Any guidance in how to proceed is much appreciated!

Pinging **@twiecki**

Ref #4126

Reviewers

MarcoGorelli

twiecki

Assignees

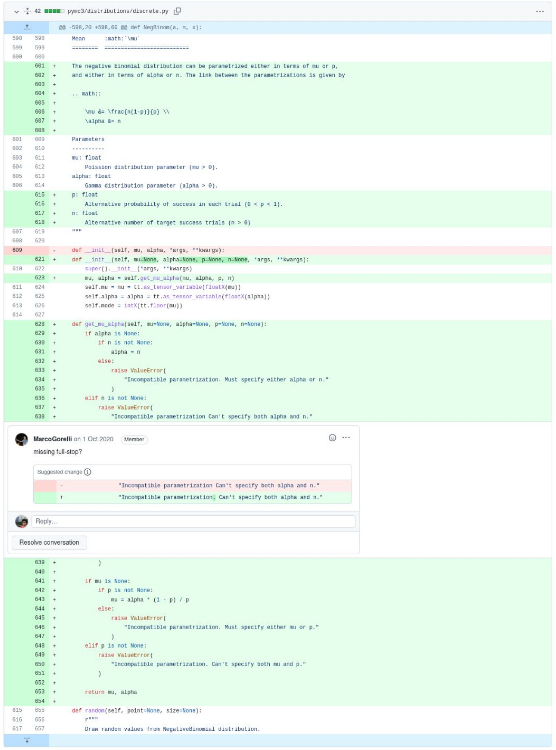
No one—assign yoursel

Labels

None yet

Projects

None yet



```
42 pymc3/distributions/discrete.py

@@ -598,20 +598,60 @@ def NegBinom(a, m, x):
     Mean      :math:`\mu`
     ========  ==========================

+    The negative binomial distribution can be parametrized either in terms of mu or p,
+    and either in terms of alpha or n. The link between the parametrizations is given by
+
+    .. math::
+
+        \mu &= \frac{n(1-p)}{p} \\
+        \alpha &= n
+
     Parameters
     ----------
     mu: float
         Poission distribution parameter (mu > 0).
     alpha: float
         Gamma distribution parameter (alpha > 0).
+    p: float
+        Alternative probability of success in each trial (0 < p < 1).
+    n: float
+        Alternative number of target success trials (n > 0)
     """

-    def __init__(self, mu, alpha, *args, **kwargs):
+    def __init__(self, mu=None, alpha=None, p=None, n=None, *args, **kwargs):
         super().__init__(*args, **kwargs)
+        mu, alpha = self.get_mu_alpha(mu, alpha, p, n)
         self.mu = mu = tt.as_tensor_variable(floatX(mu))
         self.alpha = alpha = tt.as_tensor_variable(floatX(alpha))
         self.mode = intX(tt.floor(mu))

+    def get_mu_alpha(self, mu=None, alpha=None, p=None, n=None):
+        if alpha is None:
+            if n is not None:
+                alpha = n
+            else:
+                raise ValueError(
+                    "Incompatible parametrization. Must specify either alpha or n."
+                )
+        elif n is not None:
+            raise ValueError(
+                "Incompatible parametrization Can't specify both alpha and n."
```

MarcoGorelli on 1 Oct 2020 Member

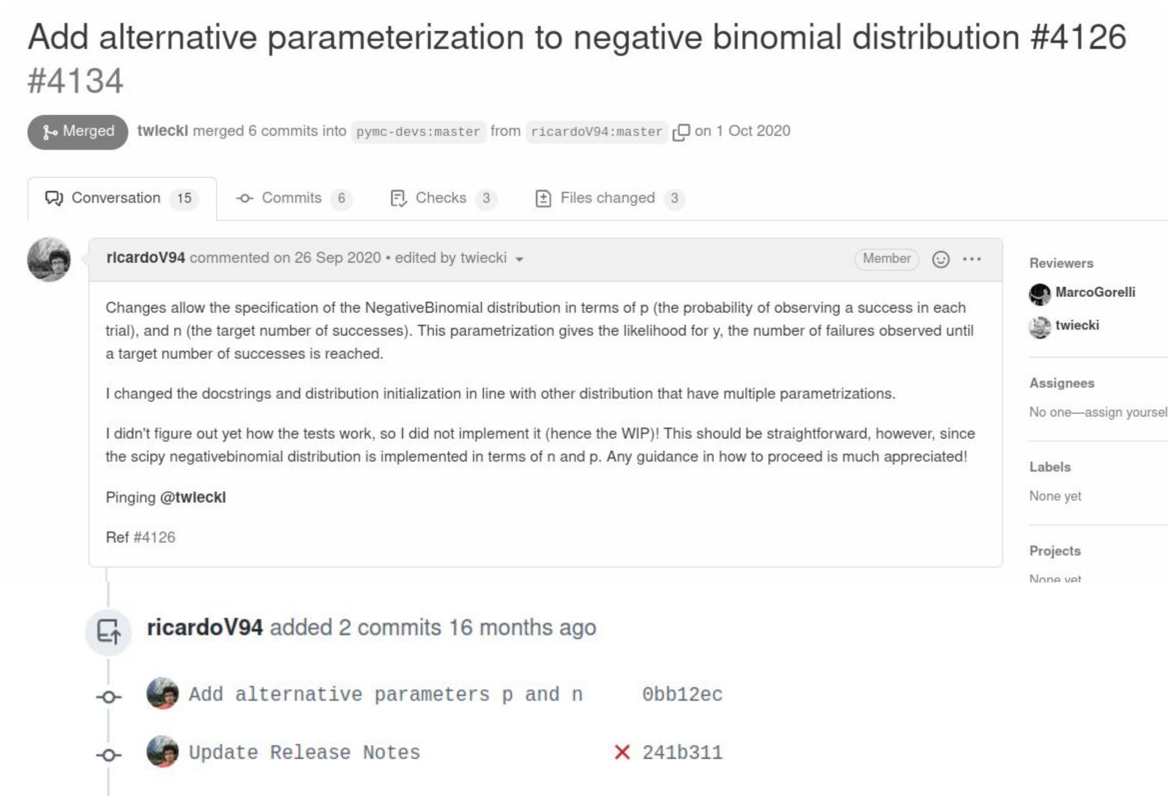
missing full-stop?

Suggested change

```
-                "Incompatible parametrization Can't specify both alpha and n."
+                "Incompatible parametrization. Can't specify both alpha and n."
```

```
+            )
+
+        if mu is None:
+            if p is not None:
+                mu = alpha * (1 - p) / p
+            else:
+                raise ValueError(
+                    "Incompatible parametrization. Must specify either mu or p."
+                )
+        elif p is not None:
+            raise ValueError(
+                "Incompatible parametrization. Can't specify both mu and p."
+            )
+
+        return mu, alpha
+
     def random(self, point=None, size=None):
         r"""
         Draw random values from NegativeBinomial distribution.
```

What does a (first) PR look like?



Add alternative parameterization to negative binomial distribution #4126 #4134

Merged **twiecki** merged 6 commits into `pymc-devs:master` from `ricardoV94:master` on 1 Oct 2020

Conversation 15 · Commits 6 · Checks 3 · Files changed 3

ricardoV94 commented on 26 Sep 2020 • edited by twiecki · Member

Changes allow the specification of the NegativeBinomial distribution in terms of p (the probability of observing a success in each trial), and n (the target number of successes). This parametrization gives the likelihood for y, the number of failures observed until a target number of successes is reached.

I changed the docstrings and distribution initialization in line with other distribution that have multiple parametrizations.

I didn't figure out yet how the tests work, so I did not implement it (hence the WIP)! This should be straightforward, however, since the scipy negativebinomial distribution is implemented in terms of n and p. Any guidance in how to proceed is much appreciated!

Pinging **@twiecki**

Ref #4126

Reviewers: **MarcoGorelli**, **twiecki**

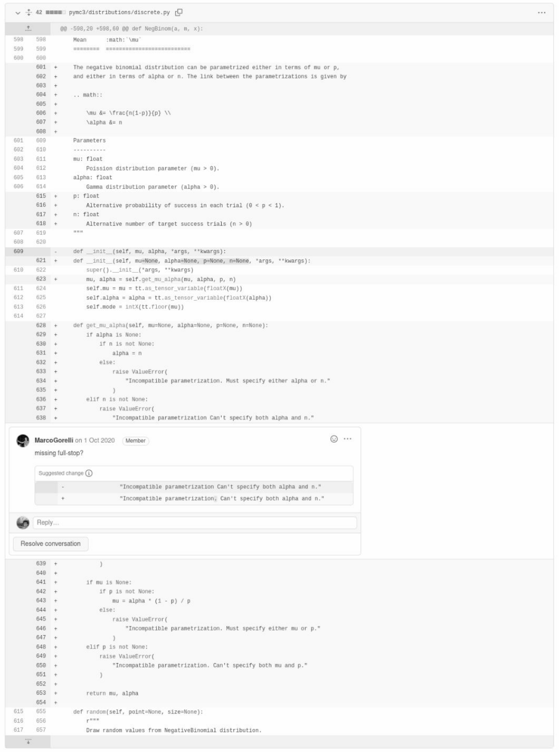
Assignees: No one—assign yoursel

Labels: None yet

Projects: None yet

ricardoV94 added 2 commits 16 months ago

```
Add alternative parameters p and n     0bb12ec
Update Release Notes                 ✗ 241b311
```



42 pymc3/distributions/discrete.py

```
@@ -598,20 +598,60 @@ def NegBinom(a, m, x):
      Mean      :math:`\mu`
      ========  ==========================

+     The negative binomial distribution can be parametrized either in terms of mu or p,
+     and either in terms of alpha or n. The link between the parametrizations is given by
+
+     .. math::
+
+         \mu &= \frac{n(1-p)}{p} \\
+         \alpha &= n
+
      Parameters
      ----------
      mu: float
          Poission distribution parameter (mu > 0).
      alpha: float
          Gamma distribution parameter (alpha > 0).
+     p: float
+         Alternative probability of success in each trial (0 < p < 1).
+     n: float
+         Alternative number of target success trials (n > 0)
      """

-     def __init__(self, mu, alpha, *args, **kwargs):
+     def __init__(self, mu=None, alpha=None, p=None, n=None, *args, **kwargs):
          super().__init__(*args, **kwargs)
+         mu, alpha = self.get_mu_alpha(mu, alpha, p, n)
          self.mu = mu = tt.as_tensor_variable(floatX(mu))
          self.alpha = alpha = tt.as_tensor_variable(floatX(alpha))
          self.mode = intX(tt.floor(mu))

+     def get_mu_alpha(self, mu=None, alpha=None, p=None, n=None):
+         if alpha is None:
+             if n is not None:
+                 alpha = n
+             else:
+                 raise ValueError(
+                     "Incompatible parametrization. Must specify either alpha or n."
+                 )
+         elif n is not None:
+             raise ValueError(
+                 "Incompatible parametrization Can't specify both alpha and n."
```

MarcoGorelli on 1 Oct 2020 · Member

missing full-stop?

Suggested change

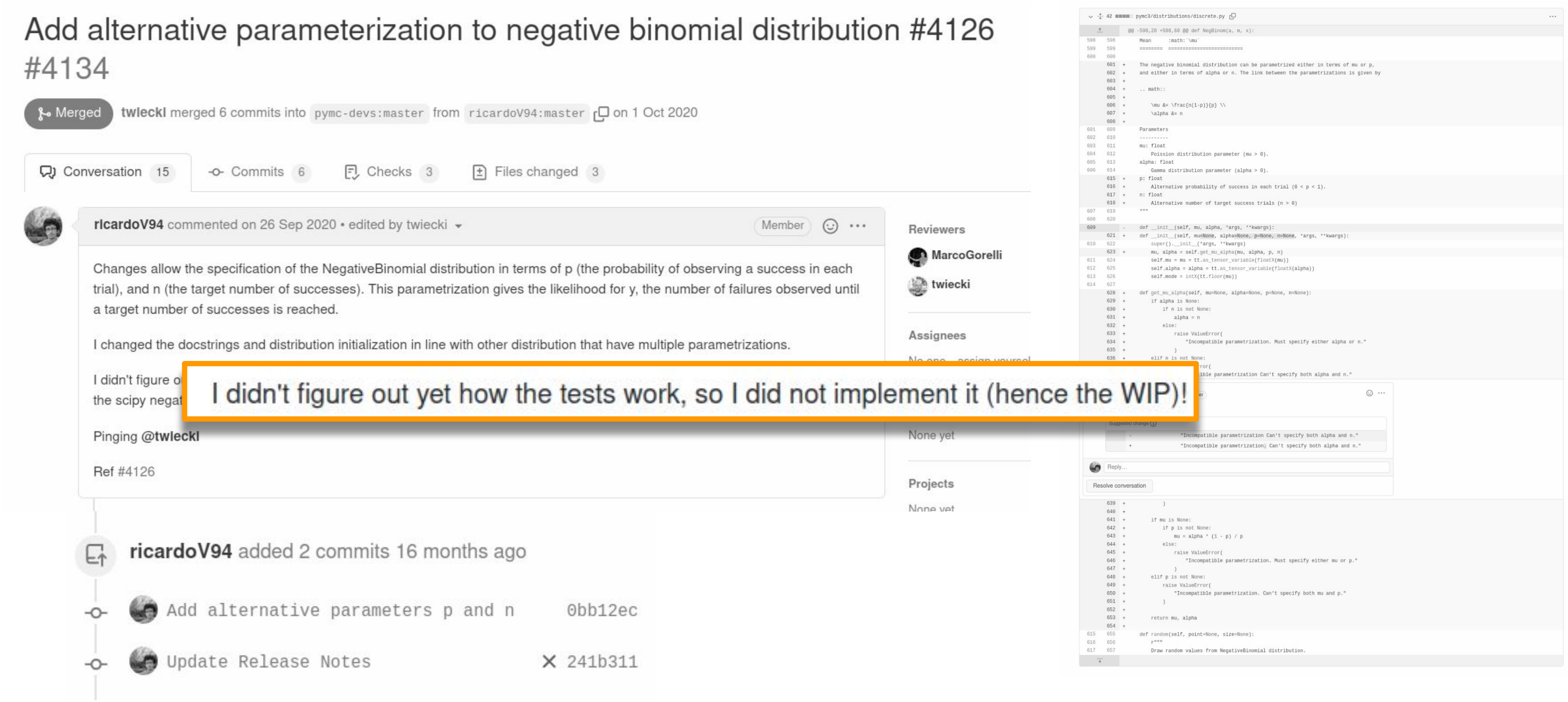
```
-                 "Incompatible parametrization Can't specify both alpha and n."
+                 "Incompatible parametrization. Can't specify both alpha and n."
```

Reply...

Resolve conversation

```
+             )
+
+         if mu is None:
+             if p is not None:
+                 mu = alpha * (1 - p) / p
+             else:
+                 raise ValueError(
+                     "Incompatible parametrization. Must specify either mu or p."
+                 )
+         elif p is not None:
+             raise ValueError(
+                 "Incompatible parametrization. Can't specify both mu and p."
+             )
+
+         return mu, alpha
+
      def random(self, point=None, size=None):
          r"""
          Draw random values from NegativeBinomial distribution.
```

What does a (first) PR look like?



Add alternative parameterization to negative binomial distribution #4126 #4134

Merged twiecki merged 6 commits into `pymc-devs:master` from `ricardoV94:master` on 1 Oct 2020

Conversation 15 · Commits 6 · Checks 3 · Files changed 3

ricardoV94 commented on 26 Sep 2020 • edited by twiecki · Member

Changes allow the specification of the NegativeBinomial distribution in terms of p (the probability of observing a success in each trial), and n (the target number of successes). This parametrization gives the likelihood for y, the number of failures observed until a target number of successes is reached.

I changed the docstrings and distribution initialization in line with other distribution that have multiple parametrizations.

I didn't figure o
the scipy negat

Pinging @twiecki

Ref #4126

Reviewers: MarcoGorelli, twiecki

Assignees: No one—assign yoursel

None yet

Projects: None yet

ricardoV94 added 2 commits 16 months ago

```
Add alternative parameters p and n     0bb12ec
Update Release Notes                 ✕ 241b311
```

I didn't figure out yet how the tests work, so I did not implement it (hence the WIP)!

42 pymc3/distributions/discrete.py

```
@@ -598,20 +598,60 @@ def NegBinom(a, m, x):
      Mean      :math:`\mu`
      ========  ==========================

+     The negative binomial distribution can be parametrized either in terms of mu or p,
+     and either in terms of alpha or n. The link between the parametrizations is given by
+
+     .. math::
+
+         \mu &= \frac{n(1-p)}{p} \\
+         \alpha &= n
+
      Parameters
      ----------
      mu: float
          Poission distribution parameter (mu > 0).
      alpha: float
          Gamma distribution parameter (alpha > 0).
+     p: float
+         Alternative probability of success in each trial (0 < p < 1).
+     n: float
+         Alternative number of target success trials (n > 0)
      """

-     def __init__(self, mu, alpha, *args, **kwargs):
+     def __init__(self, mu=None, alpha=None, p=None, n=None, *args, **kwargs):
          super().__init__(*args, **kwargs)
+         mu, alpha = self.get_mu_alpha(mu, alpha, p, n)
          self.mu = mu = tt.as_tensor_variable(floatX(mu))
          self.alpha = alpha = tt.as_tensor_variable(floatX(alpha))
          self.mode = intX(tt.floor(mu))

+     def get_mu_alpha(self, mu=None, alpha=None, p=None, n=None):
+         if alpha is None:
+             if n is not None:
+                 alpha = n
+             else:
+                 raise ValueError(
+                     "Incompatible parametrization. Must specify either alpha or n."
+                 )
+         elif n is not None:
                          ror(
                          ible parametrization Can't specify both alpha and n."
```

Suggested change

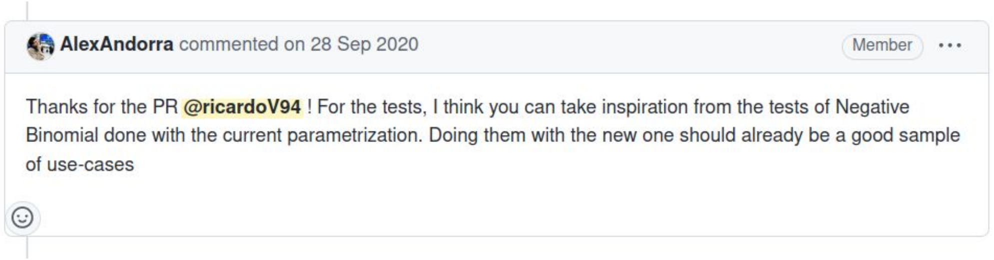
```
-                     "Incompatible parametrization Can't specify both alpha and n."
+                     "Incompatible parametrization. Can't specify both alpha and n."
```

Reply...

Resolve conversation

```
+                 )
+
+         if mu is None:
+             if p is not None:
+                 mu = alpha * (1 - p) / p
+             else:
+                 raise ValueError(
+                     "Incompatible parametrization. Must specify either mu or p."
+                 )
+         elif p is not None:
+             raise ValueError(
+                 "Incompatible parametrization. Can't specify both mu and p."
+             )
+
+         return mu, alpha
+
      def random(self, point=None, size=None):
          r"""
          Draw random values from NegativeBinomial distribution.
```

What does a (first) PR look like?

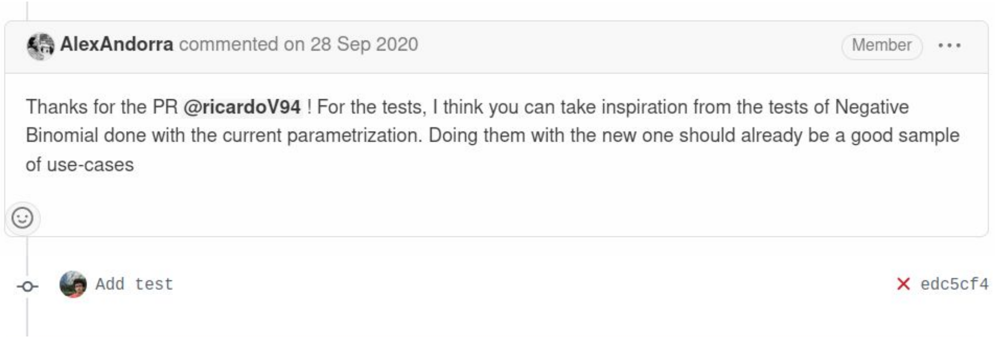


AlexAndorra commented on 28 Sep 2020

Member

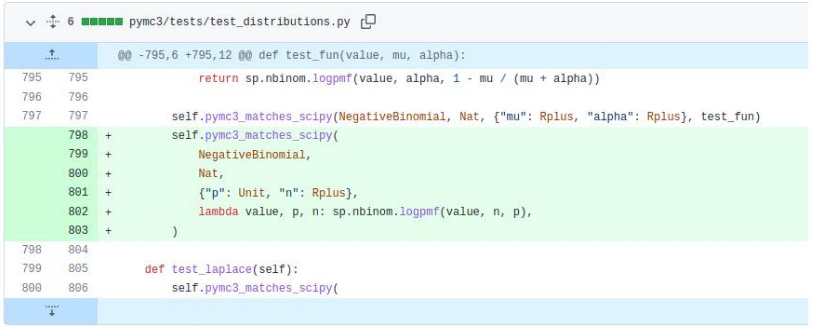
Thanks for the PR **@ricardoV94** ! For the tests, I think you can take inspiration from the tests of Negative Binomial done with the current parametrization. Doing them with the new one should already be a good sample of use-cases

What does a (first) PR look like?


AlexAndorra commented on 28 Sep 2020 Member

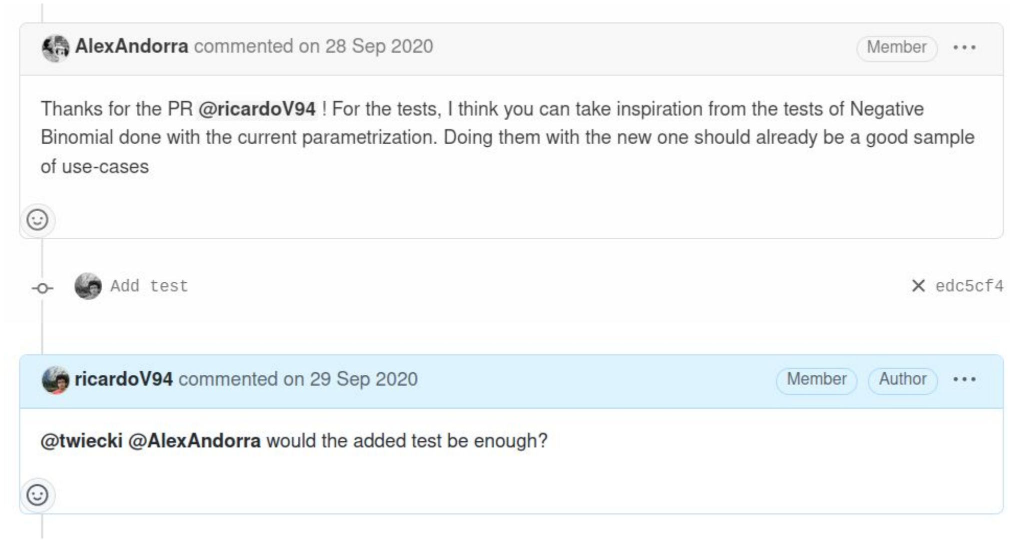
Thanks for the PR **@ricardoV94** ! For the tests, I think you can take inspiration from the tests of Negative Binomial done with the current parametrization. Doing them with the new one should already be a good sample of use-cases

Add test ✗ edc5cf4


6 pymc3/tests/test_distributions.py

```
@@ -795,6 +795,12 @@ def test_fun(value, mu, alpha):
                return sp.nbinom.logpmf(value, alpha, 1 - mu / (mu + alpha))

            self.pymc3_matches_scipy(NegativeBinomial, Nat, {"mu": Rplus, "alpha": Rplus}, test_fun)
+           self.pymc3_matches_scipy(
+               NegativeBinomial,
+               Nat,
+               {"p": Unit, "n": Rplus},
+               lambda value, p, n: sp.nbinom.logpmf(value, n, p),
+           )

        def test_laplace(self):
            self.pymc3_matches_scipy(
```

What does a (first) PR look like?



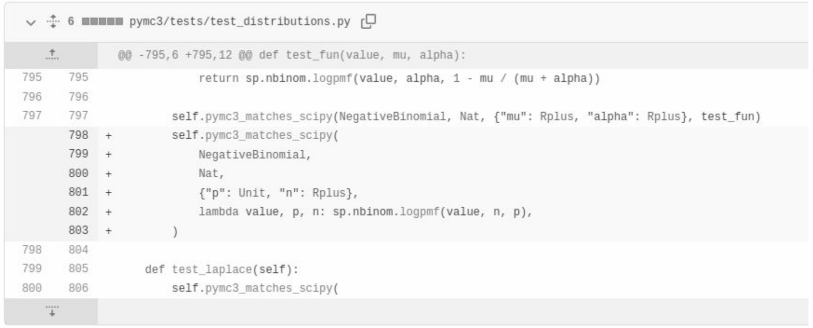
AlexAndorra commented on 28 Sep 2020 Member

Thanks for the PR **@ricardoV94** ! For the tests, I think you can take inspiration from the tests of Negative Binomial done with the current parametrization. Doing them with the new one should already be a good sample of use-cases

Add test edc5cf4

ricardoV94 commented on 29 Sep 2020 Member Author

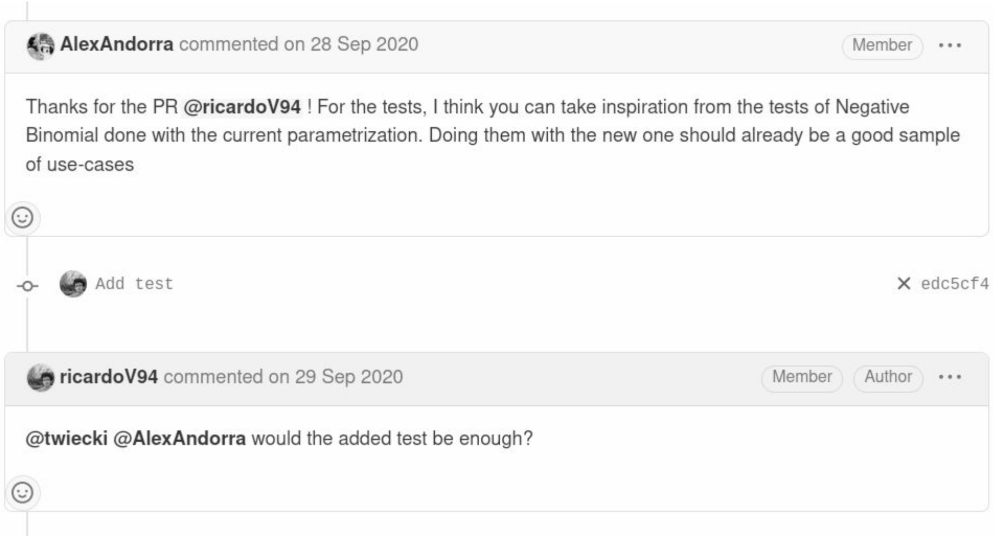
@twiecki @AlexAndorra would the added test be enough?



6 pymc3/tests/test_distributions.py

```
@@ -795,6 +795,12 @@ def test_fun(value, mu, alpha):
            return sp.nbinom.logpmf(value, alpha, 1 - mu / (mu + alpha))

        self.pymc3_matches_scipy(NegativeBinomial, Nat, {"mu": Rplus, "alpha": Rplus}, test_fun)
+       self.pymc3_matches_scipy(
+           NegativeBinomial,
+           Nat,
+           {"p": Unit, "n": Rplus},
+           lambda value, p, n: sp.nbinom.logpmf(value, n, p),
+       )

    def test_laplace(self):
        self.pymc3_matches_scipy(
```

What does a (first) PR look like?



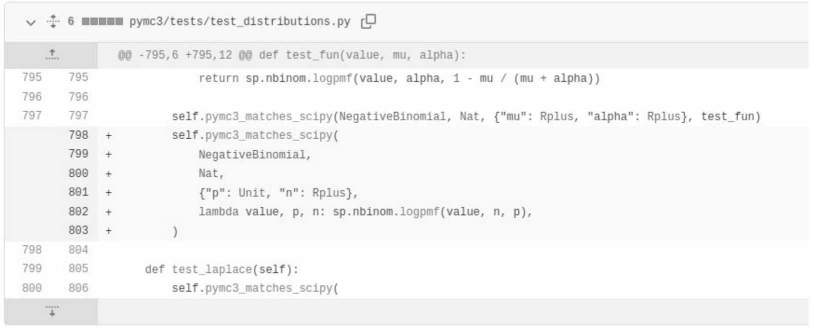
AlexAndorra commented on 28 Sep 2020 · Member

Thanks for the PR **@ricardoV94** ! For the tests, I think you can take inspiration from the tests of Negative Binomial done with the current parametrization. Doing them with the new one should already be a good sample of use-cases

`Add test` ✕ edc5cf4

ricardoV94 commented on 29 Sep 2020 · Member · Author

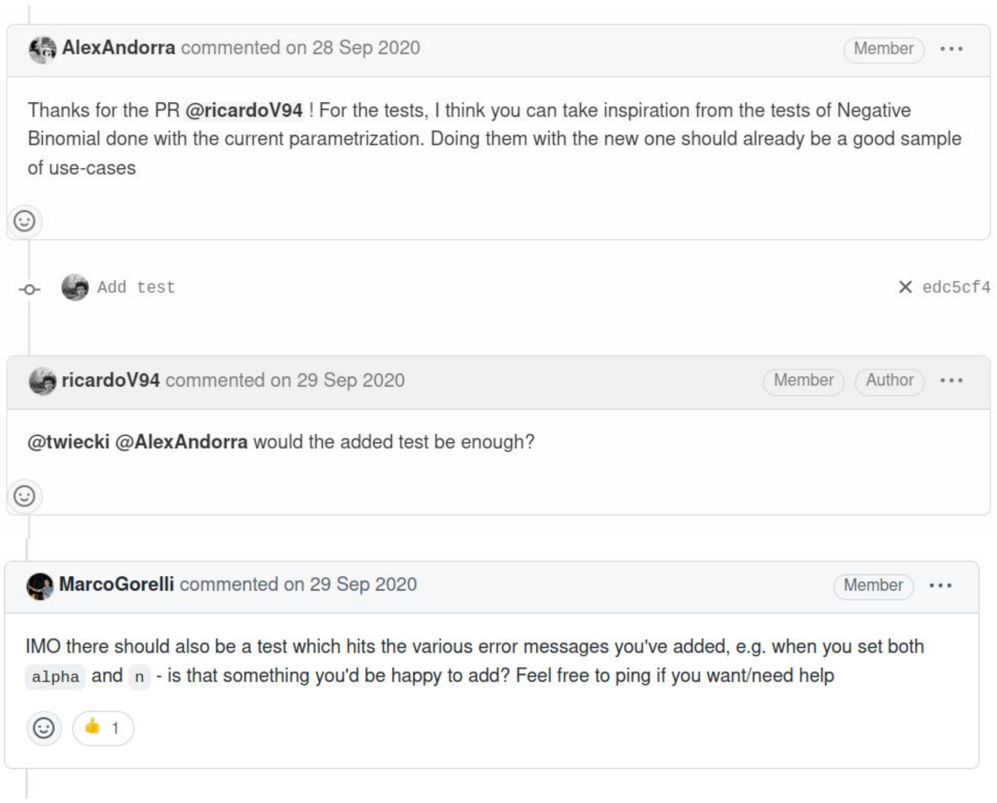
@twiecki @AlexAndorra would the added test be enough?



6 pymc3/tests/test_distributions.py

```
@@ -795,6 +795,12 @@ def test_fun(value, mu, alpha):
                 return sp.nbinom.logpmf(value, alpha, 1 - mu / (mu + alpha))

             self.pymc3_matches_scipy(NegativeBinomial, Nat, {"mu": Rplus, "alpha": Rplus}, test_fun)
+            self.pymc3_matches_scipy(
+                NegativeBinomial,
+                Nat,
+                {"p": Unit, "n": Rplus},
+                lambda value, p, n: sp.nbinom.logpmf(value, n, p),
+            )

         def test_laplace(self):
             self.pymc3_matches_scipy(
```

What does a (first) PR look like?



AlexAndorra commented on 28 Sep 2020 · Member

Thanks for the PR **@ricardoV94** ! For the tests, I think you can take inspiration from the tests of Negative Binomial done with the current parametrization. Doing them with the new one should already be a good sample of use-cases

`Add test` edc5cf4

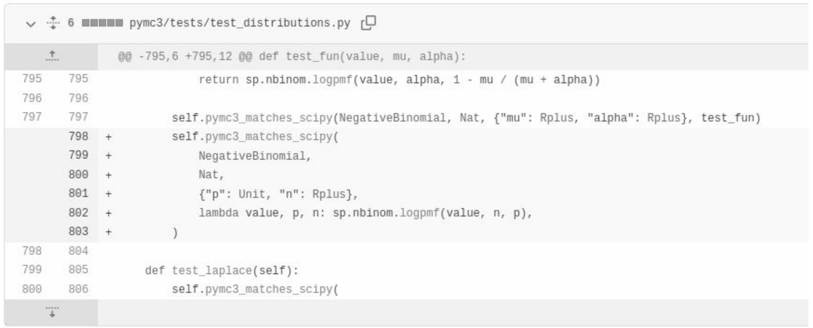
ricardoV94 commented on 29 Sep 2020 · Member · Author

@twiecki **@AlexAndorra** would the added test be enough?

MarcoGorelli commented on 29 Sep 2020 · Member

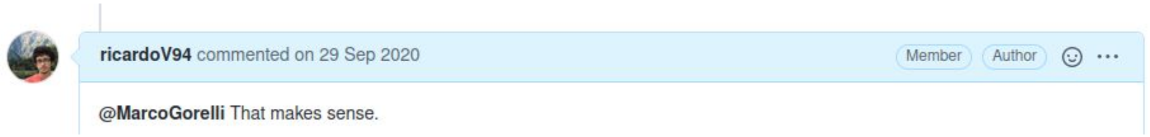
IMO there should also be a test which hits the various error messages you've added, e.g. when you set both `alpha` and `n` - is that something you'd be happy to add? Feel free to ping if you want/need help

👍 1



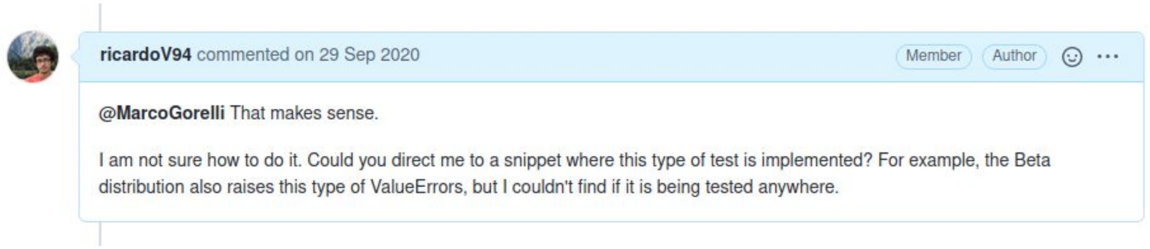
```
6 ■■■■■ pymc3/tests/test_distributions.py
@@ -795,6 +795,12 @@ def test_fun(value, mu, alpha):
                 return sp.nbinom.logpmf(value, alpha, 1 - mu / (mu + alpha))

             self.pymc3_matches_scipy(NegativeBinomial, Nat, {"mu": Rplus, "alpha": Rplus}, test_fun)
+            self.pymc3_matches_scipy(
+                NegativeBinomial,
+                Nat,
+                {"p": Unit, "n": Rplus},
+                lambda value, p, n: sp.nbinom.logpmf(value, n, p),
+            )

         def test_laplace(self):
             self.pymc3_matches_scipy(
```

What does a (first) PR look like?


ricardoV94 commented on 29 Sep 2020

@MarcoGorelli That makes sense.

What does a (first) PR look like?



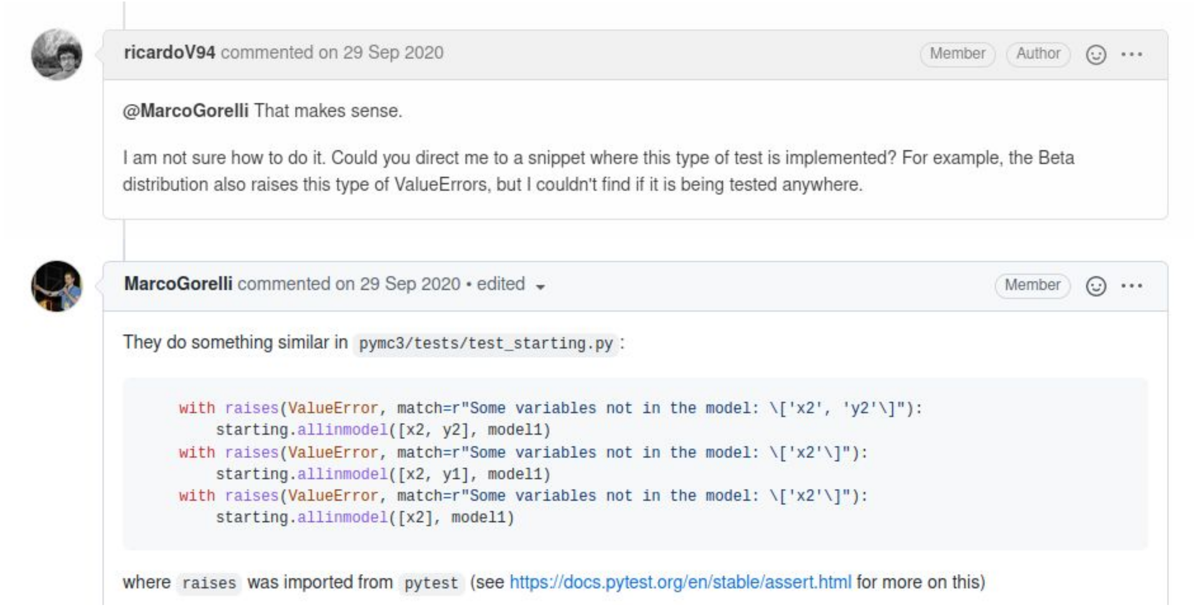
ricardoV94 commented on 29 Sep 2020

Member Author

@MarcoGorelli That makes sense.

I am not sure how to do it. Could you direct me to a snippet where this type of test is implemented? For example, the Beta distribution also raises this type of ValueErrors, but I couldn't find if it is being tested anywhere.

What does a (first) PR look like?



ricardoV94 commented on 29 Sep 2020 Member Author

@MarcoGorelli That makes sense.

I am not sure how to do it. Could you direct me to a snippet where this type of test is implemented? For example, the Beta distribution also raises this type of ValueErrors, but I couldn't find if it is being tested anywhere.

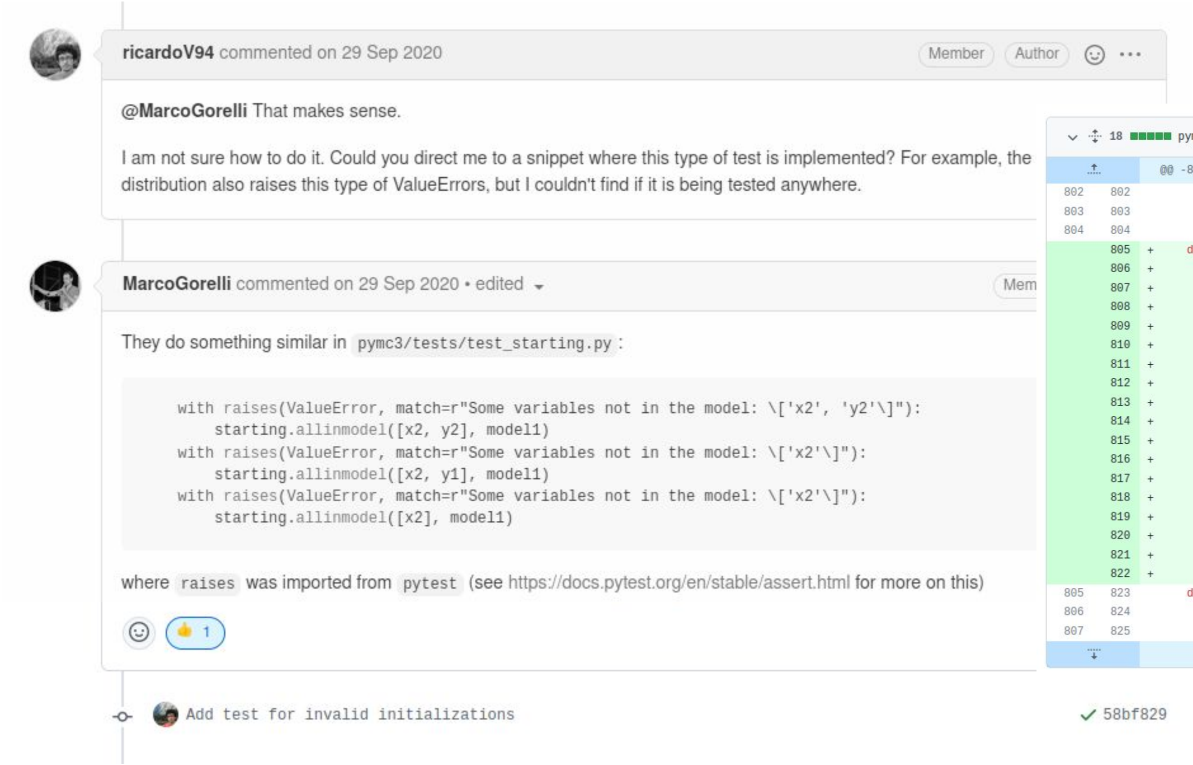
MarcoGorelli commented on 29 Sep 2020 • edited Member

They do something similar in `pymc3/tests/test_starting.py`:

```
    with raises(ValueError, match=r"Some variables not in the model: \['x2', 'y2'\]"):
        starting.allinmodel([x2, y2], model1)
    with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
        starting.allinmodel([x2, y1], model1)
    with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
        starting.allinmodel([x2], model1)
```

where `raises` was imported from `pytest` (see https://docs.pytest.org/en/stable/assert.html for more on this)

What does a (first) PR look like?



ricardoV94 commented on 29 Sep 2020 Member Author

@MarcoGorelli That makes sense.

I am not sure how to do it. Could you direct me to a snippet where this type of test is implemented? For example, the distribution also raises this type of ValueErrors, but I couldn't find if it is being tested anywhere.

MarcoGorelli commented on 29 Sep 2020 • edited Mem

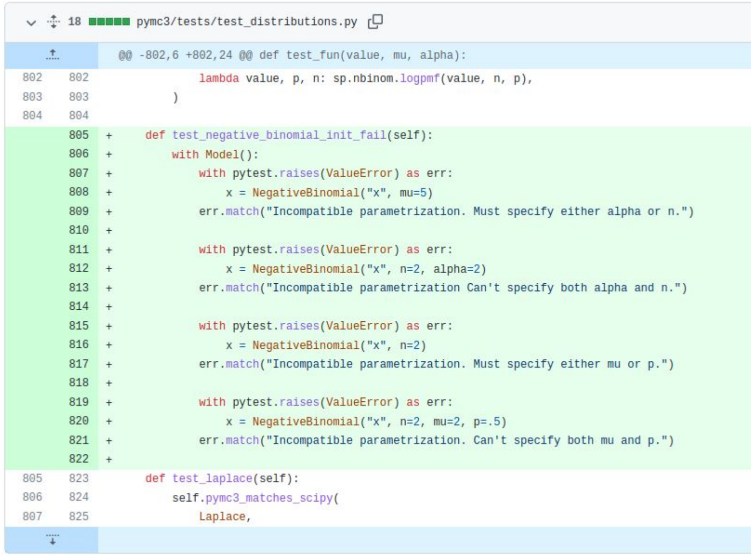
They do something similar in `pymc3/tests/test_starting.py` :

```
with raises(ValueError, match=r"Some variables not in the model: \['x2', 'y2'\]"):
    starting.allinmodel([x2, y2], model1)
with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
    starting.allinmodel([x2, y1], model1)
with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
    starting.allinmodel([x2], model1)
```

where `raises` was imported from `pytest` (see https://docs.pytest.org/en/stable/assert.html for more on this)

👍 1

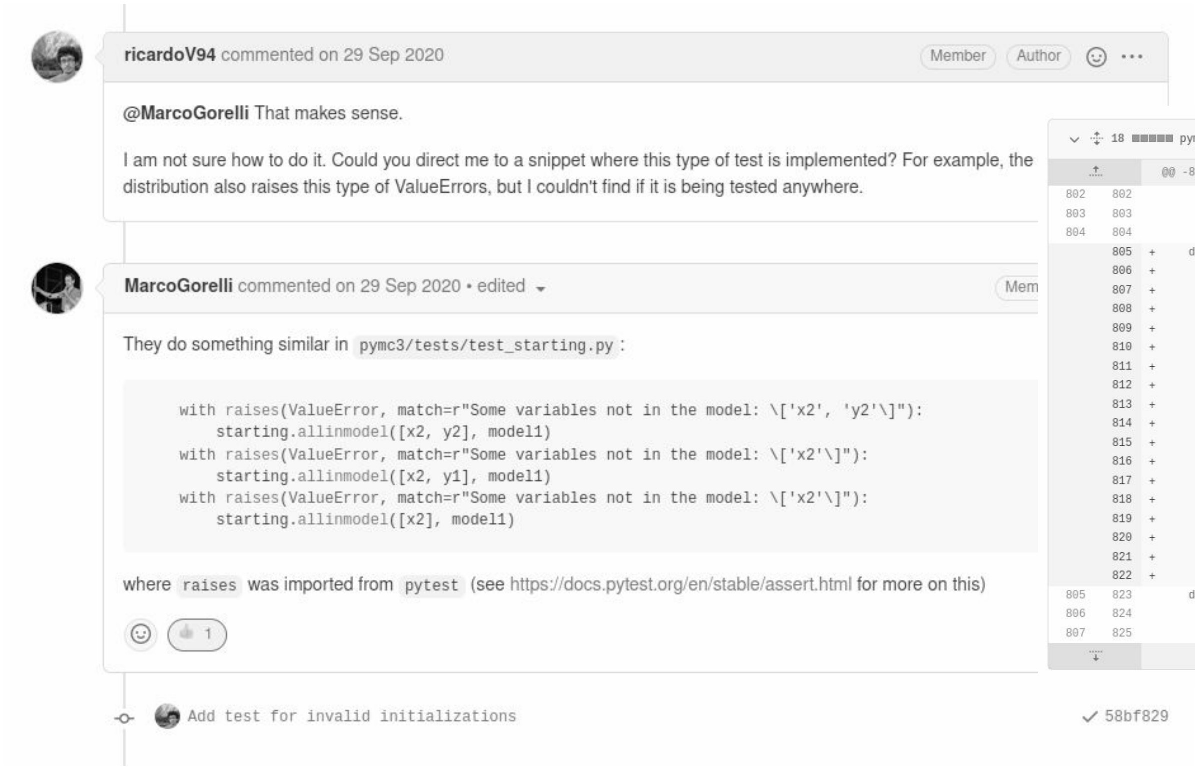
Add test for invalid initializations ✓ 58bf829



18 pymc3/tests/test_distributions.py

```
@@ -802,6 +802,24 @@ def test_fun(value, mu, alpha):
            lambda value, p, n: sp.nbinom.logpmf(value, n, p),
        )

+    def test_negative_binomial_init_fail(self):
+        with Model():
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", mu=5)
+            err.match("Incompatible parametrization. Must specify either alpha or n.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2, alpha=2)
+            err.match("Incompatible parametrization Can't specify both alpha and n.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2)
+            err.match("Incompatible parametrization. Must specify either mu or p.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2, mu=2, p=.5)
+            err.match("Incompatible parametrization. Can't specify both mu and p.")
+
    def test_laplace(self):
        self.pymc3_matches_scipy(
            Laplace,
```

What does a (first) PR look like?



ricardoV94 commented on 29 Sep 2020 · Member · Author

@MarcoGorelli That makes sense.

I am not sure how to do it. Could you direct me to a snippet where this type of test is implemented? For example, the distribution also raises this type of ValueErrors, but I couldn't find if it is being tested anywhere.

MarcoGorelli commented on 29 Sep 2020 • edited · Mem

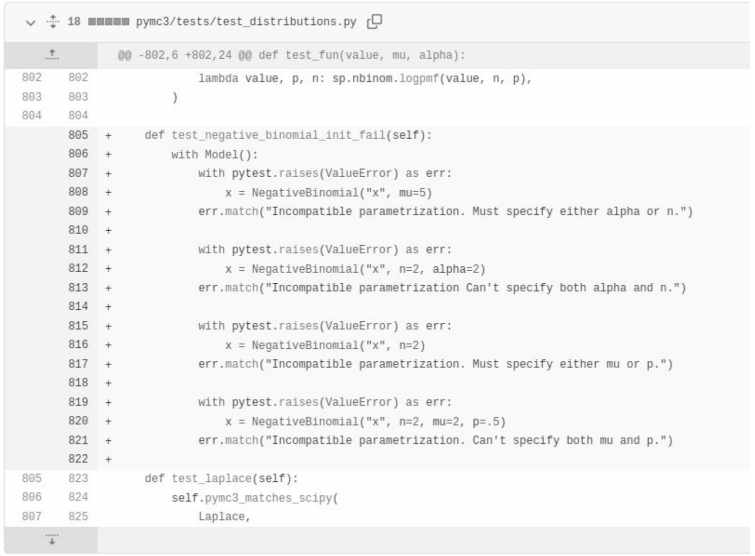
They do something similar in `pymc3/tests/test_starting.py` :

```
    with raises(ValueError, match=r"Some variables not in the model: \['x2', 'y2'\]"):
        starting.allinmodel([x2, y2], model1)
    with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
        starting.allinmodel([x2, y1], model1)
    with raises(ValueError, match=r"Some variables not in the model: \['x2'\]"):
        starting.allinmodel([x2], model1)
```

where `raises` was imported from `pytest` (see https://docs.pytest.org/en/stable/assert.html for more on this)

👍 1

Add test for invalid initializations ✓ 58bf829



18 pymc3/tests/test_distributions.py

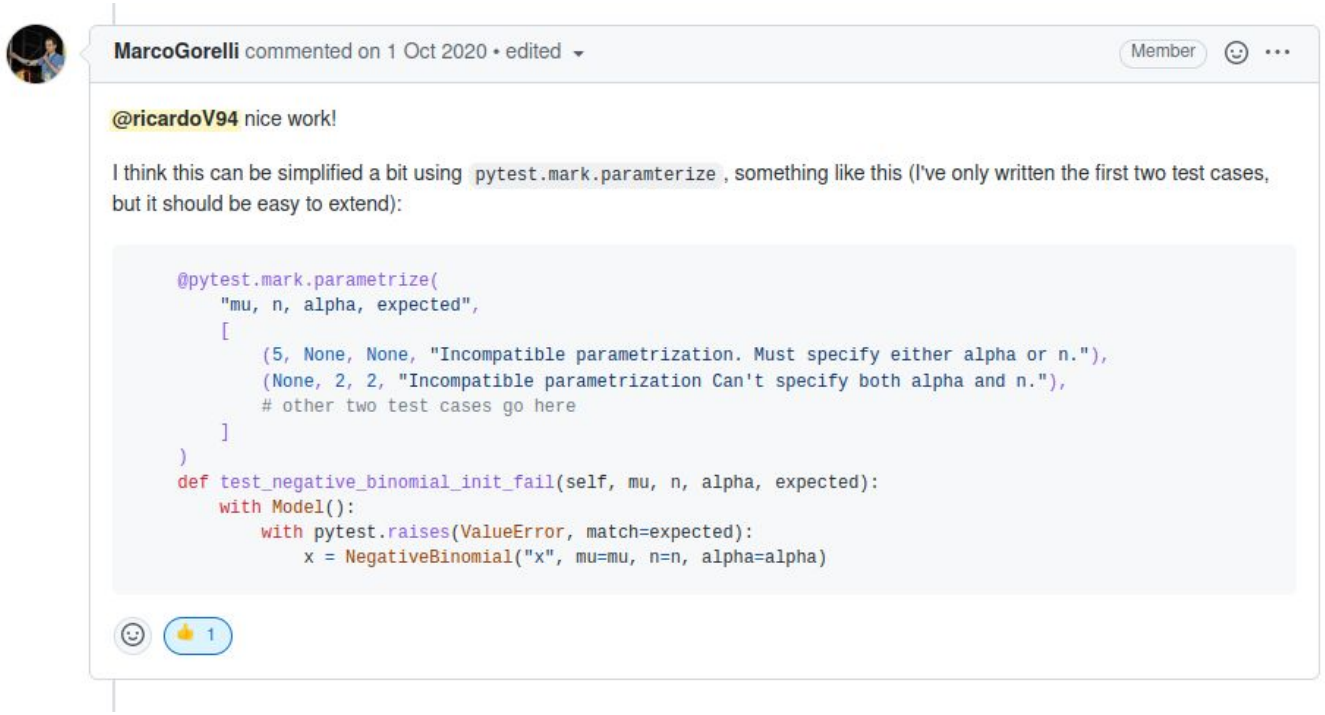
```
@@ -802,6 +802,24 @@ def test_fun(value, mu, alpha):
             lambda value, p, n: sp.nbinom.logpmf(value, n, p),
         )

+    def test_negative_binomial_init_fail(self):
+        with Model():
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", mu=5)
+            err.match("Incompatible parametrization. Must specify either alpha or n.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2, alpha=2)
+            err.match("Incompatible parametrization Can't specify both alpha and n.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2)
+            err.match("Incompatible parametrization. Must specify either mu or p.")
+
+            with pytest.raises(ValueError) as err:
+                x = NegativeBinomial("x", n=2, mu=2, p=.5)
+            err.match("Incompatible parametrization. Can't specify both mu and p.")
+
     def test_laplace(self):
         self.pymc3_matches_scipy(
             Laplace,
```

Check out Marco Gorelli: Contributing to pandas

What does a (first) PR look like?

What does a (first) PR look like?

What does a (first) PR look like?



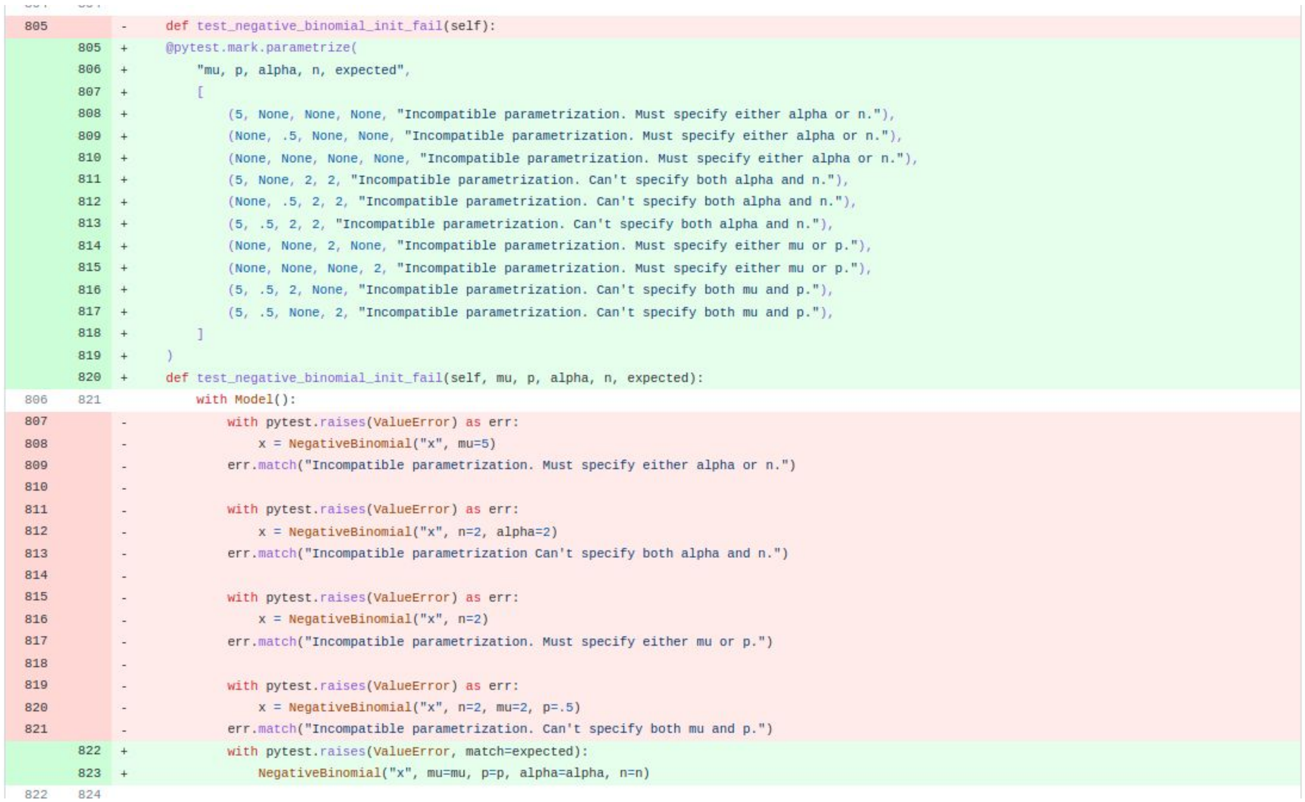
MarcoGorelli commented on 1 Oct 2020 • edited

@ricardoV94 nice work!

I think this can be simplified a bit using `pytest.mark.paramterize`, something like this (I've only written the first two test cases, but it should be easy to extend):

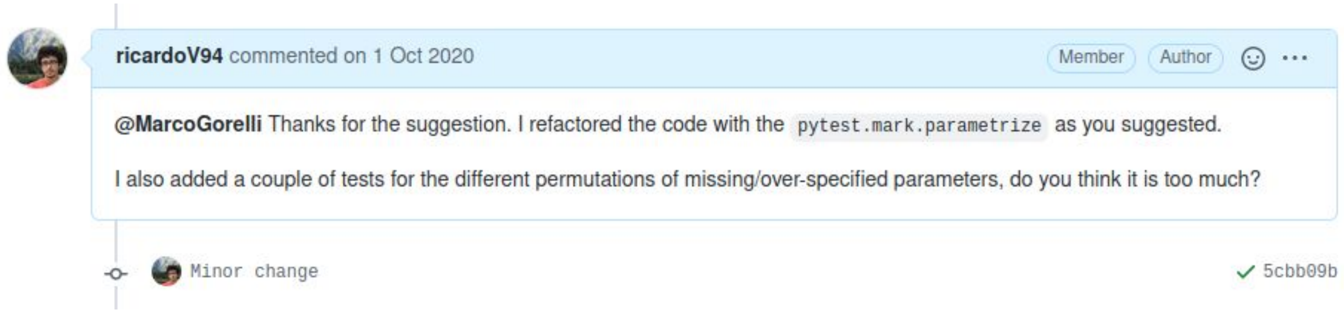
```
    @pytest.mark.parametrize(
        "mu, n, alpha, expected",
        [
            (5, None, None, "Incompatible parametrization. Must specify either alpha or n."),
            (None, 2, 2, "Incompatible parametrization Can't specify both alpha and n."),
            # other two test cases go here
        ]
    )
    def test_negative_binomial_init_fail(self, mu, n, alpha, expected):
        with Model():
            with pytest.raises(ValueError, match=expected):
                x = NegativeBinomial("x", mu=mu, n=n, alpha=alpha)
```

What does a (first) PR look like?



```diff
-    def test_negative_binomial_init_fail(self):
+    @pytest.mark.parametrize(
+        "mu, p, alpha, n, expected",
+        [
+            (5, None, None, None, "Incompatible parametrization. Must specify either alpha or n."),
+            (None, .5, None, None, "Incompatible parametrization. Must specify either alpha or n."),
+            (None, None, None, None, "Incompatible parametrization. Must specify either alpha or n."),
+            (5, None, 2, 2, "Incompatible parametrization. Can't specify both alpha and n."),
+            (None, .5, 2, 2, "Incompatible parametrization. Can't specify both alpha and n."),
+            (5, .5, 2, 2, "Incompatible parametrization. Can't specify both alpha and n."),
+            (None, None, 2, None, "Incompatible parametrization. Must specify either mu or p."),
+            (None, None, None, 2, "Incompatible parametrization. Must specify either mu or p."),
+            (5, .5, 2, None, "Incompatible parametrization. Can't specify both mu and p."),
+            (5, .5, None, 2, "Incompatible parametrization. Can't specify both mu and p."),
+        ]
+    )
+    def test_negative_binomial_init_fail(self, mu, p, alpha, n, expected):
         with Model():
-            with pytest.raises(ValueError) as err:
-                x = NegativeBinomial("x", mu=5)
-            err.match("Incompatible parametrization. Must specify either alpha or n.")
-
-            with pytest.raises(ValueError) as err:
-                x = NegativeBinomial("x", n=2, alpha=2)
-            err.match("Incompatible parametrization Can't specify both alpha and n.")
-
-            with pytest.raises(ValueError) as err:
-                x = NegativeBinomial("x", n=2)
-            err.match("Incompatible parametrization. Must specify either mu or p.")
-
-            with pytest.raises(ValueError) as err:
-                x = NegativeBinomial("x", n=2, mu=2, p=.5)
-            err.match("Incompatible parametrization. Can't specify both mu and p.")
+            with pytest.raises(ValueError, match=expected):
+                NegativeBinomial("x", mu=mu, p=p, alpha=alpha, n=n)
```

What does a (first) PR look like?



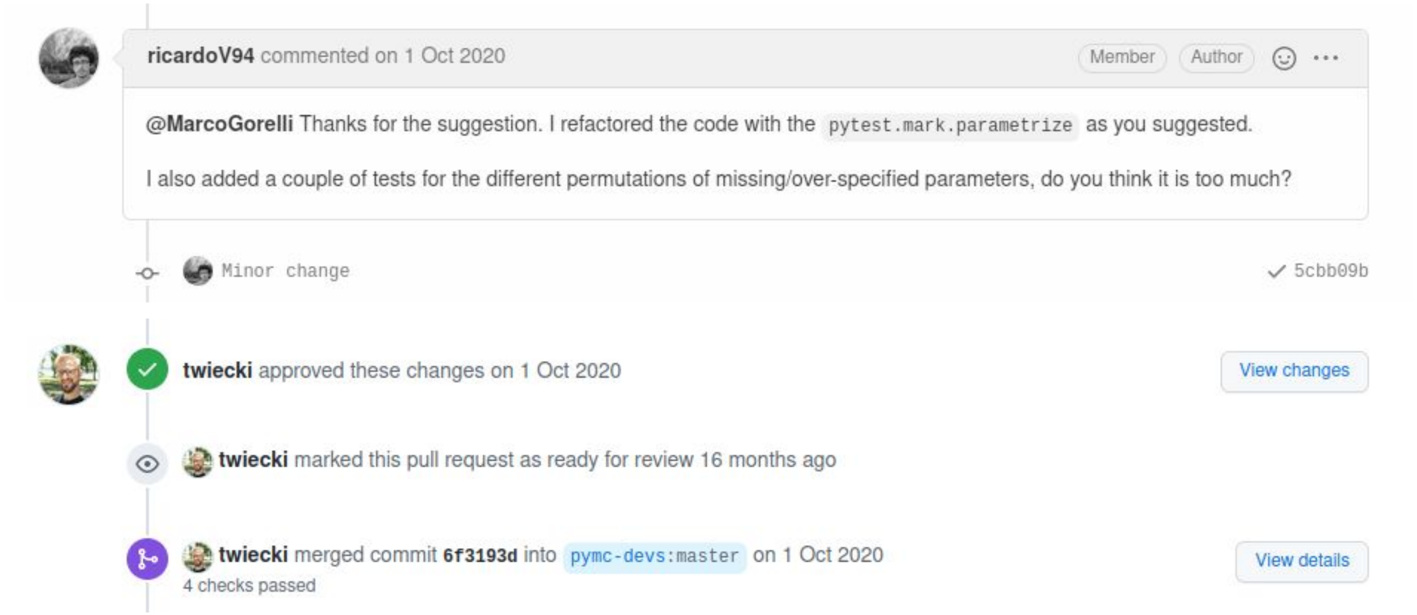
ricardoV94 commented on 1 Oct 2020

@MarcoGorelli Thanks for the suggestion. I refactored the code with the `pytest.mark.parametrize` as you suggested.

I also added a couple of tests for the different permutations of missing/over-specified parameters, do you think it is too much?

`Minor change` ✓ `5cbb09b`

What does a (first) PR look like?



ricardoV94 commented on 1 Oct 2020 · Member · Author

@MarcoGorelli Thanks for the suggestion. I refactored the code with the `pytest.mark.parametrize` as you suggested.

I also added a couple of tests for the different permutations of missing/over-specified parameters, do you think it is too much?

`Minor change` ✓ `5cbb09b`

twiecki approved these changes on 1 Oct 2020 · View changes

twiecki marked this pull request as ready for review 16 months ago

twiecki merged commit **6f3193d** into `pymc-devs:master` on 1 Oct 2020 · View details

4 checks passed

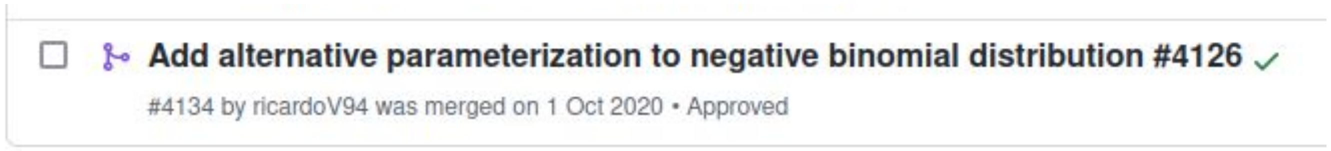
What was it like?

- More work than expected
- Learned many new concepts:
    - unittesting
    - parametrizing
    - monkey-patching
    - code style checks
- There is a whole community to support **and challenge** you
- It was fun?

SO MUCH FUN

I came back for more

26 Sep - 01 Oct



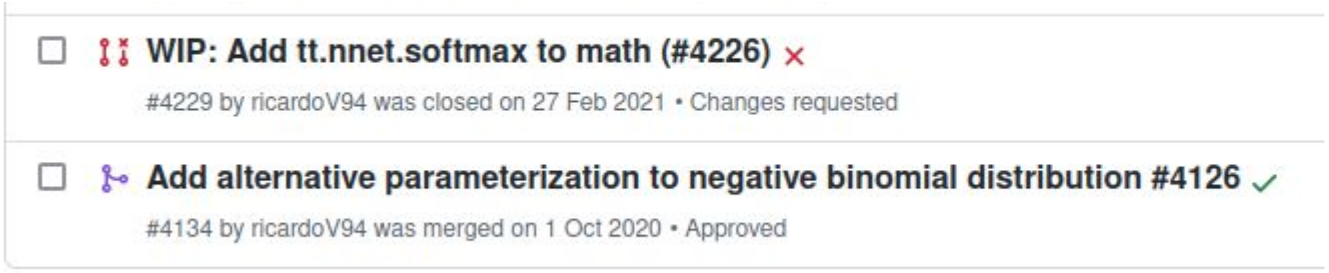
Add alternative parameterization to negative binomial distribution #4126 ✓

#4134 by ricardoV94 was merged on 1 Oct 2020 • Approved

I came back for more

16 Nov - 27 Feb

26 Sep - 01 Oct



WIP: Add tt.nnet.softmax to math (#4226) ✗

#4229 by ricardoV94 was closed on 27 Feb 2021 • Changes requested

Add alternative parameterization to negative binomial distribution #4126 ✓

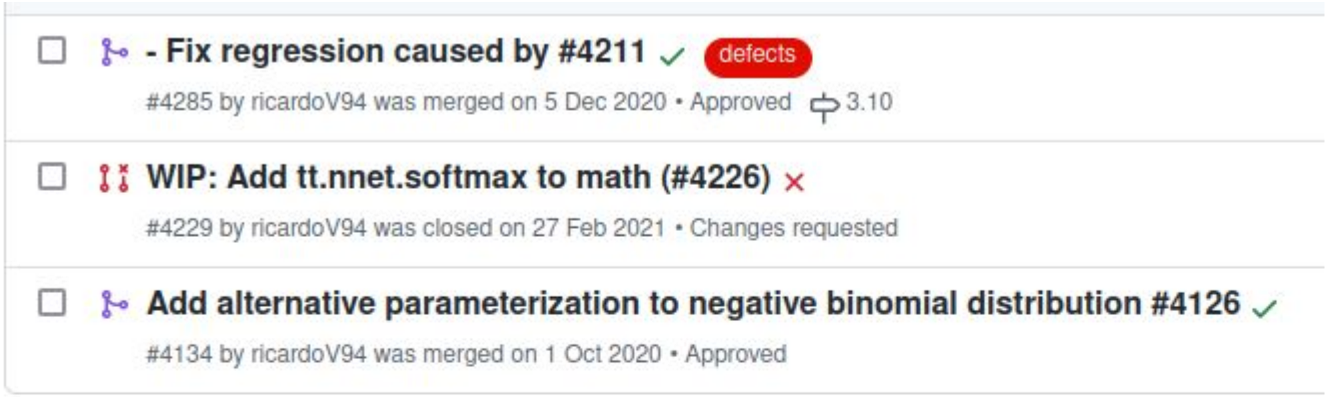
#4134 by ricardoV94 was merged on 1 Oct 2020 • Approved

I came back for more

2 Dec - 5 Dec

16 Nov - 27 Feb

26 Sep - 01 Oct



- Fix regression caused by #4211 ✓ defects

#4285 by ricardoV94 was merged on 5 Dec 2020 • Approved 3.10

WIP: Add tt.nnet.softmax to math (#4226) ✗

#4229 by ricardoV94 was closed on 27 Feb 2021 • Changes requested

Add alternative parameterization to negative binomial distribution #4126 ✓

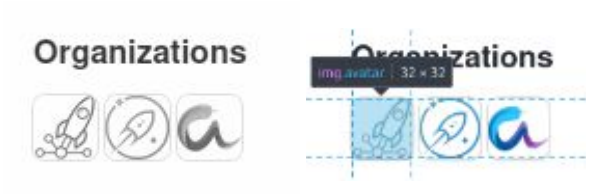
#4134 by ricardoV94 was merged on 1 Oct 2020 • Approved

One year later they gave me the keys

One year later they gave me the keys

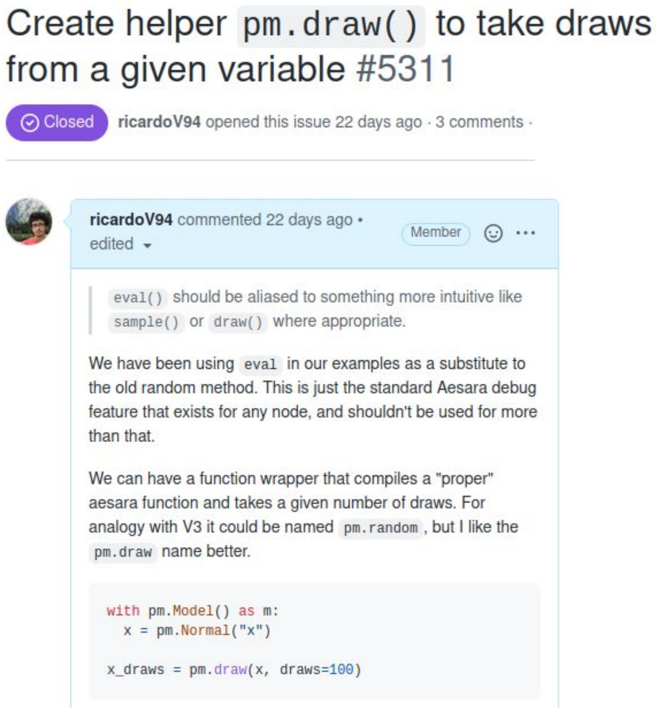
One year later they gave me the keys

One year later they gave me the keys



Reviewing a first time contributor PR

Reviewing a first time contributor PR



Create helper `pm.draw()` to take draws from a given variable #5311

Closed ricardoV94 opened this issue 22 days ago · 3 comments ·

ricardoV94 commented 22 days ago • edited

Member

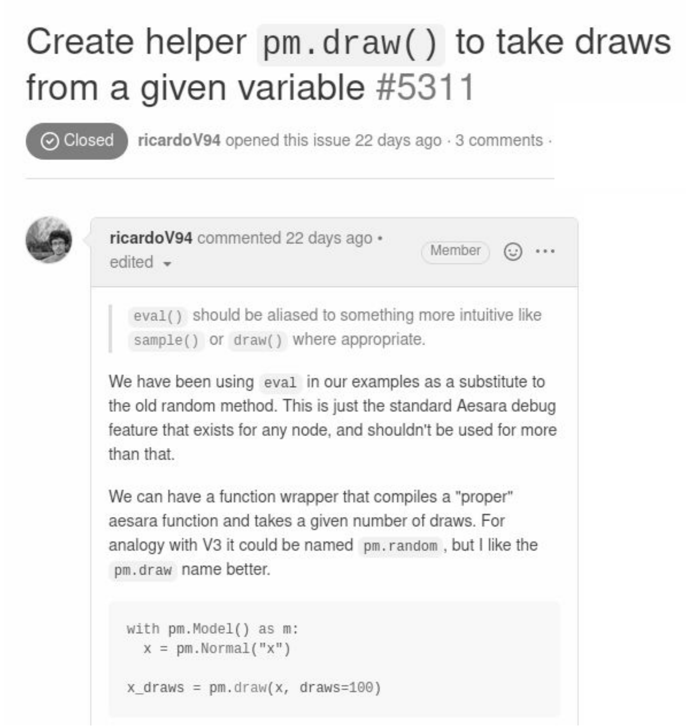
> `eval()` should be aliased to something more intuitive like `sample()` or `draw()` where appropriate.

We have been using `eval` in our examples as a substitute to the old random method. This is just the standard Aesara debug feature that exists for any node, and shouldn't be used for more than that.

We can have a function wrapper that compiles a "proper" aesara function and takes a given number of draws. For analogy with V3 it could be named `pm.random`, but I like the `pm.draw` name better.

```
with pm.Model() as m:
  x = pm.Normal("x")

x_draws = pm.draw(x, draws=100)
```

Reviewing a first time contributor PR



Create helper `pm.draw()` to take draws from a given variable #5311

Closed ricardoV94 opened this issue 22 days ago · 3 comments ·

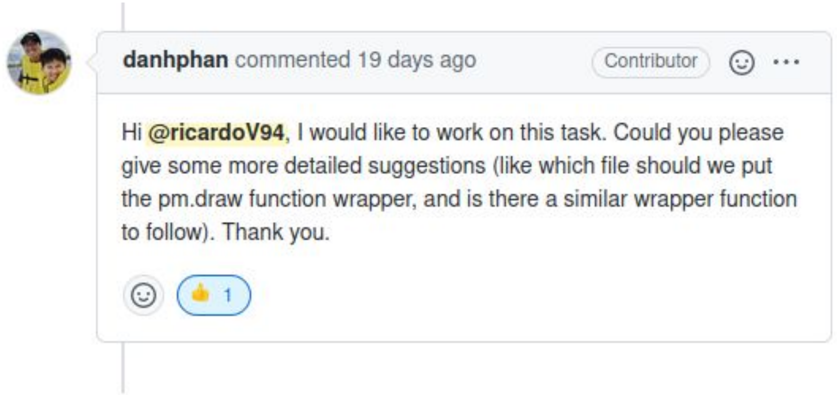
ricardoV94 commented 22 days ago • edited Member

> `eval()` should be aliased to something more intuitive like `sample()` or `draw()` where appropriate.

We have been using `eval` in our examples as a substitute to the old random method. This is just the standard Aesara debug feature that exists for any node, and shouldn't be used for more than that.

We can have a function wrapper that compiles a "proper" aesara function and takes a given number of draws. For analogy with V3 it could be named `pm.random`, but I like the `pm.draw` name better.

```
with pm.Model() as m:
  x = pm.Normal("x")

x_draws = pm.draw(x, draws=100)
```

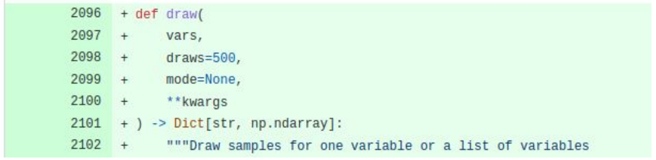


danhphan commented 19 days ago Contributor

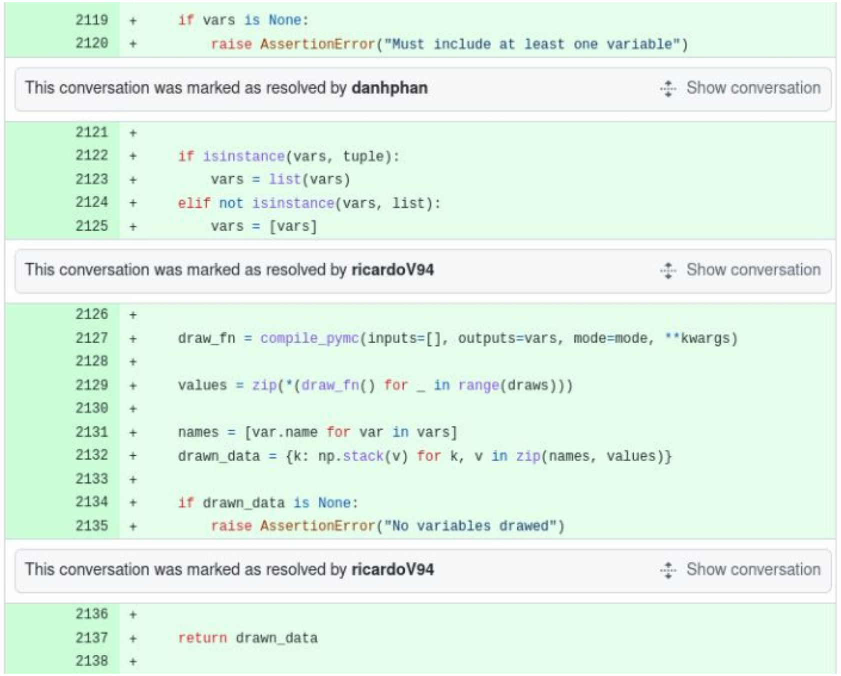
Hi @ricardoV94, I would like to work on this task. Could you please give some more detailed suggestions (like which file should we put the pm.draw function wrapper, and is there a similar wrapper function to follow). Thank you.

👍 1

Reviewing a first time contributor PR



```
+ def draw(
+     vars,
+     draws=500,
+     mode=None,
+     **kwargs
+ ) -> Dict[str, np.ndarray]:
+     """Draw samples for one variable or a list of variables
```



```
+     if vars is None:
+         raise AssertionError("Must include at least one variable")
```

This conversation was marked as resolved by **danhphan** Show conversation

```
+
+     if isinstance(vars, tuple):
+         vars = list(vars)
+     elif not isinstance(vars, list):
+         vars = [vars]
```

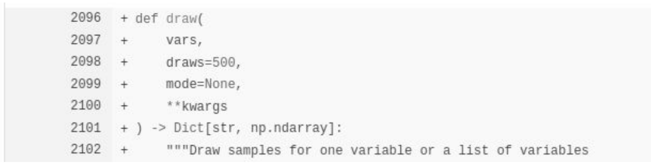
This conversation was marked as resolved by **ricardoV94** Show conversation

```
+
+     draw_fn = compile_pymc(inputs=[], outputs=vars, mode=mode, **kwargs)
+
+     values = zip(*(draw_fn() for _ in range(draws)))
+
+     names = [var.name for var in vars]
+     drawn_data = {k: np.stack(v) for k, v in zip(names, values)}
+
+     if drawn_data is None:
+         raise AssertionError("No variables drawed")
```

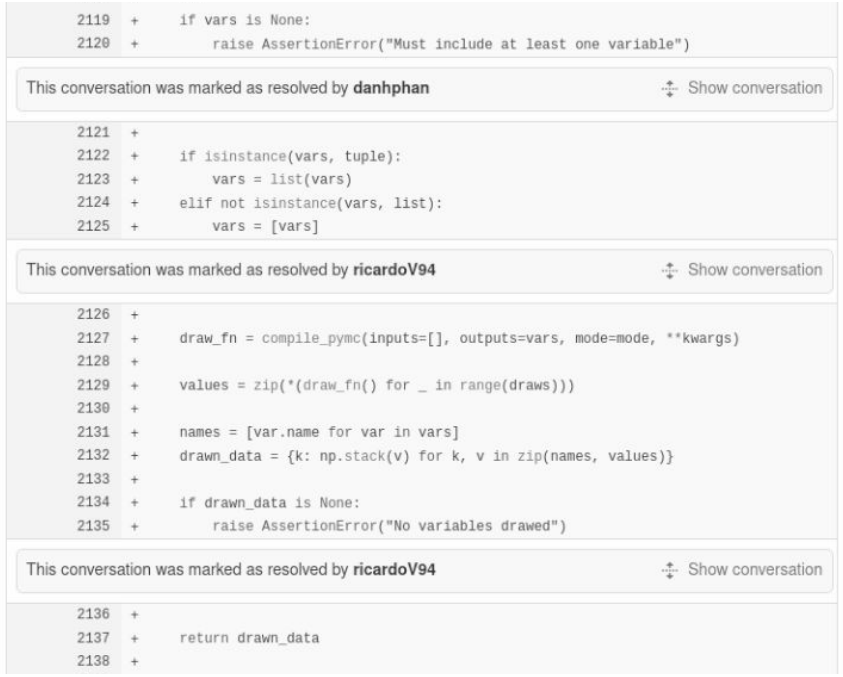
This conversation was marked as resolved by **ricardoV94** Show conversation

```
+
+     return drawn_data
+
```

Reviewing a first time contributor PR



```
+ def draw(
+     vars,
+     draws=500,
+     mode=None,
+     **kwargs
+ ) -> Dict[str, np.ndarray]:
+     """Draw samples for one variable or a list of variables
```



```
+     if vars is None:
+         raise AssertionError("Must include at least one variable")
```

This conversation was marked as resolved by **danhphan** Show conversation

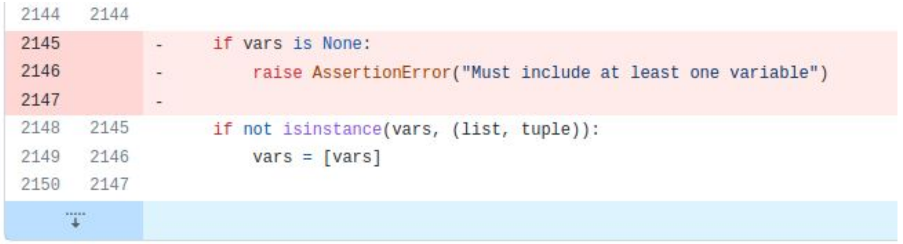
```
+
+     if isinstance(vars, tuple):
+         vars = list(vars)
+     elif not isinstance(vars, list):
+         vars = [vars]
```

This conversation was marked as resolved by **ricardoV94** Show conversation

```
+
+     draw_fn = compile_pymc(inputs=[], outputs=vars, mode=mode, **kwargs)
+
+     values = zip(*(draw_fn() for _ in range(draws)))
+
+     names = [var.name for var in vars]
+     drawn_data = {k: np.stack(v) for k, v in zip(names, values)}
+
+     if drawn_data is None:
+         raise AssertionError("No variables drawed")
```

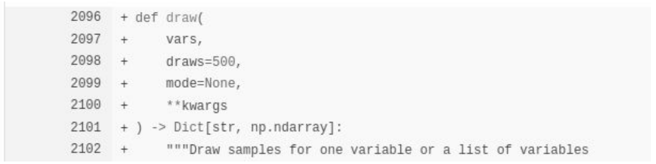
This conversation was marked as resolved by **ricardoV94** Show conversation

```
+
+     return drawn_data
+
```

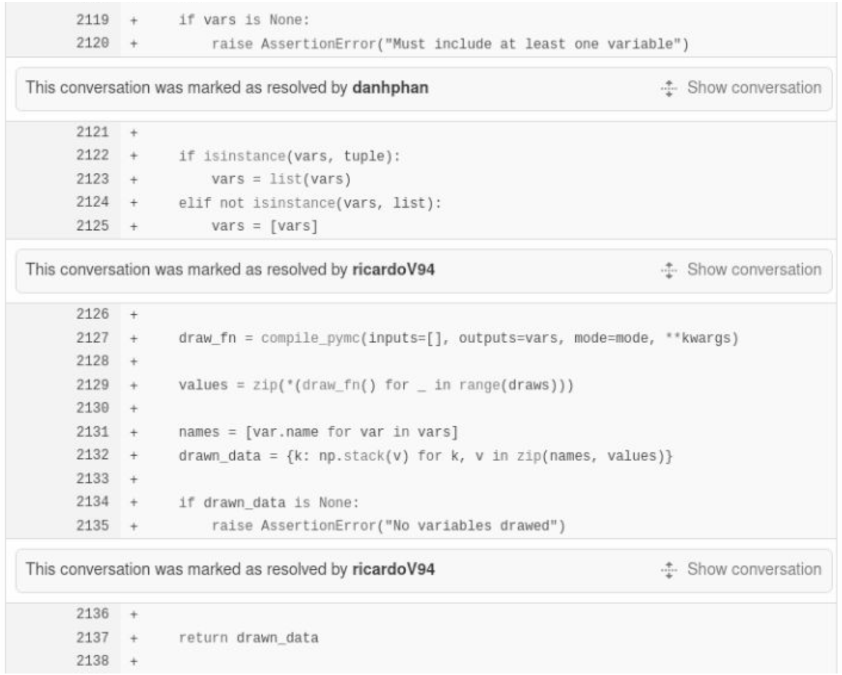


```

-     if vars is None:
-         raise AssertionError("Must include at least one variable")
-
      if not isinstance(vars, (list, tuple)):
          vars = [vars]

```

Reviewing a first time contributor PR



```
+ def draw(
+     vars,
+     draws=500,
+     mode=None,
+     **kwargs
+ ) -> Dict[str, np.ndarray]:
+     """Draw samples for one variable or a list of variables
```



```
+     if vars is None:
+         raise AssertionError("Must include at least one variable")
```

This conversation was marked as resolved by **danhphan** Show conversation

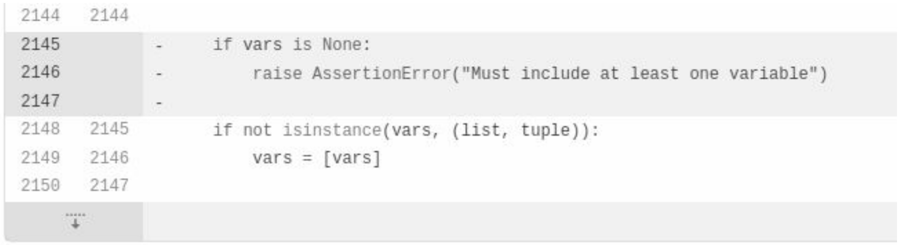
```
+
+     if isinstance(vars, tuple):
+         vars = list(vars)
+     elif not isinstance(vars, list):
+         vars = [vars]
```

This conversation was marked as resolved by **ricardoV94** Show conversation

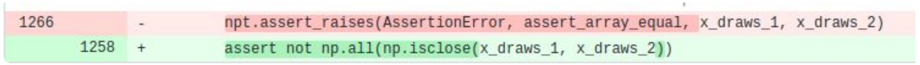
```
+
+     draw_fn = compile_pymc(inputs=[], outputs=vars, mode=mode, **kwargs)
+
+     values = zip(*(draw_fn() for _ in range(draws)))
+
+     names = [var.name for var in vars]
+     drawn_data = {k: np.stack(v) for k, v in zip(names, values)}
+
+     if drawn_data is None:
+         raise AssertionError("No variables drawed")
```

This conversation was marked as resolved by **ricardoV94** Show conversation

```
+
+     return drawn_data
+
```

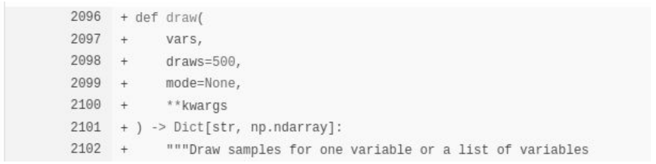


```
-     if vars is None:
-         raise AssertionError("Must include at least one variable")
-
      if not isinstance(vars, (list, tuple)):
          vars = [vars]

```

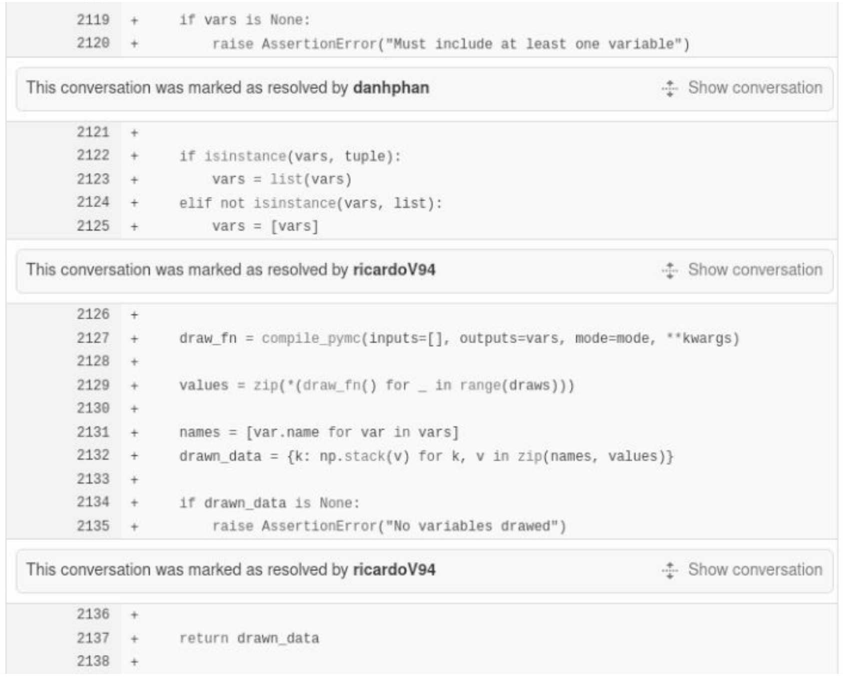


```
-         npt.assert_raises(AssertionError, assert_array_equal, x_draws_1, x_draws_2)
+         assert not np.all(np.isclose(x_draws_1, x_draws_2))
```

Reviewing a first time contributor PR



```
+ def draw(
+     vars,
+     draws=500,
+     mode=None,
+     **kwargs
+ ) -> Dict[str, np.ndarray]:
+     """Draw samples for one variable or a list of variables
```



```
+     if vars is None:
+         raise AssertionError("Must include at least one variable")
```

This conversation was marked as resolved by **danhphan** Show conversation

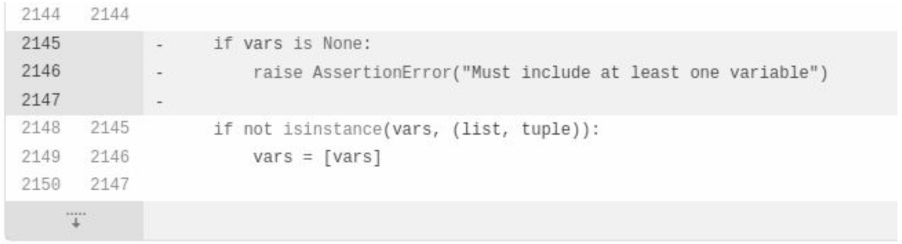
```
+
+     if isinstance(vars, tuple):
+         vars = list(vars)
+     elif not isinstance(vars, list):
+         vars = [vars]
```

This conversation was marked as resolved by **ricardoV94** Show conversation

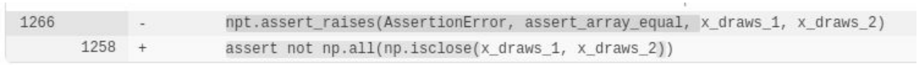
```
+
+     draw_fn = compile_pymc(inputs=[], outputs=vars, mode=mode, **kwargs)
+
+     values = zip(*(draw_fn() for _ in range(draws)))
+
+     names = [var.name for var in vars]
+     drawn_data = {k: np.stack(v) for k, v in zip(names, values)}
+
+     if drawn_data is None:
+         raise AssertionError("No variables drawed")
```

This conversation was marked as resolved by **ricardoV94** Show conversation

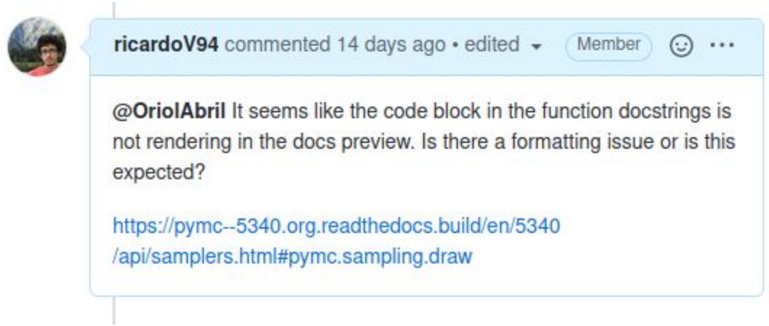
```
+
+     return drawn_data
+
```



```
-     if vars is None:
-         raise AssertionError("Must include at least one variable")
-
      if not isinstance(vars, (list, tuple)):
          vars = [vars]

```



```
-         npt.assert_raises(AssertionError, assert_array_equal, x_draws_1, x_draws_2)
+         assert not np.all(np.isclose(x_draws_1, x_draws_2))
```

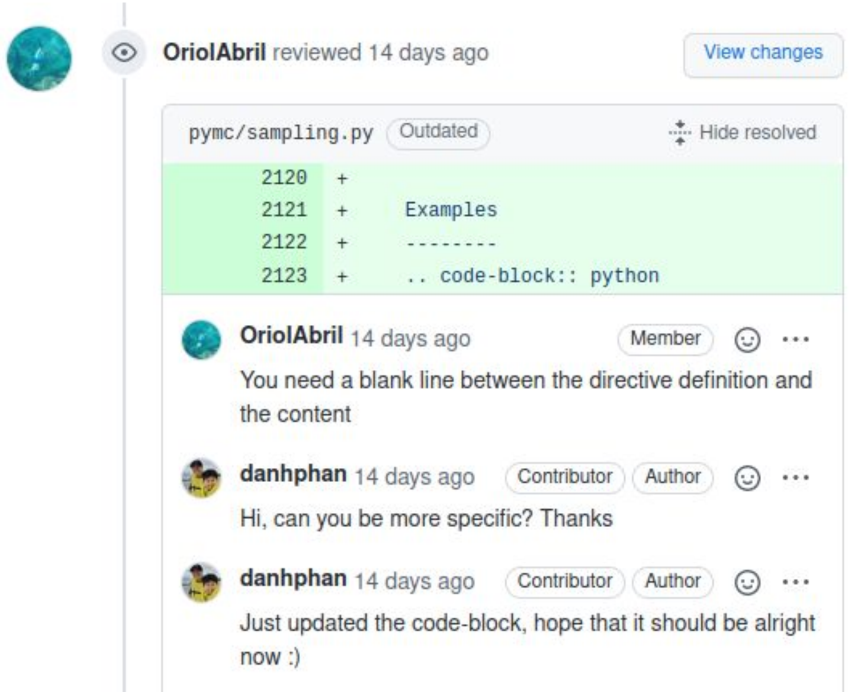


ricardoV94 commented 14 days ago • edited Member

@OriolAbril It seems like the code block in the function docstrings is not rendering in the docs preview. Is there a formatting issue or is this expected?

https://pymc--5340.org.readthedocs.build/en/5340/api/samplers.html#pymc.sampling.draw

Reviewing a first time contributor PR



OriolAbril reviewed 14 days ago

View changes

pymc/sampling.py Outdated Hide resolved

```
+
+    Examples
+    --------
+    .. code-block:: python
```

OriolAbril 14 days ago Member

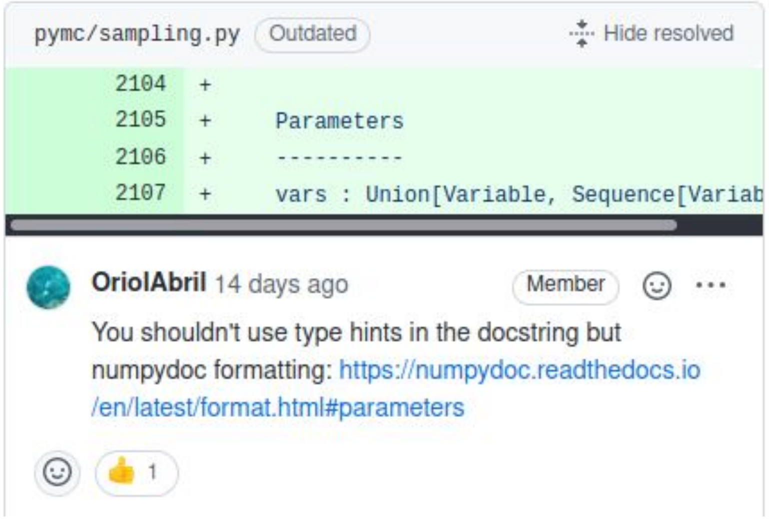
You need a blank line between the directive definition and the content

danhphan 14 days ago Contributor Author

Hi, can you be more specific? Thanks

danhphan 14 days ago Contributor Author

Just updated the code-block, hope that it should be alright now :)



pymc/sampling.py Outdated Hide resolved

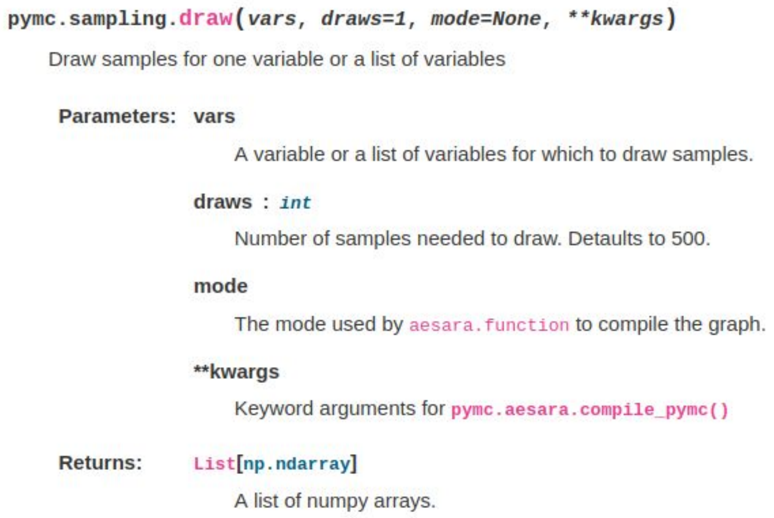
```
+
+    Parameters
+    ----------
+    vars : Union[Variable, Sequence[Variab
```

OriolAbril 14 days ago Member

You shouldn't use type hints in the docstring but numpydoc formatting: https://numpydoc.readthedocs.io/en/latest/format.html#parameters

👍 1

Reviewing a first time contributor PR



```
pymc.sampling.draw(vars, draws=1, mode=None, **kwargs)
```

Draw samples for one variable or a list of variables

Parameters: **vars**

A variable or a list of variables for which to draw samples.

draws : *int*

Number of samples needed to draw. Detaults to 500.

mode

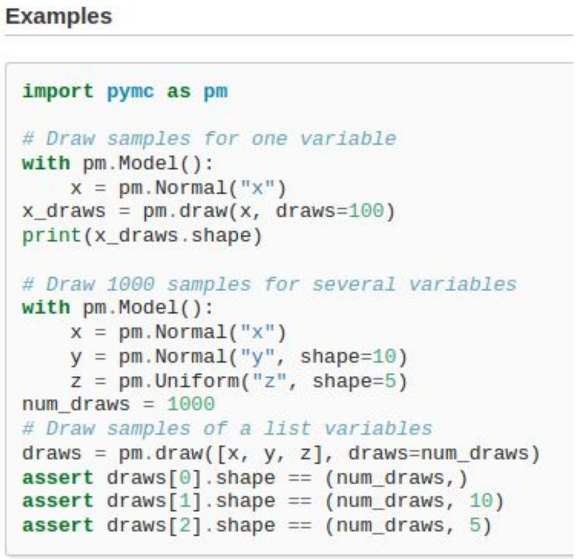
The mode used by `aesara.function` to compile the graph.

****kwargs**

Keyword arguments for `pymc.aesara.compile_pymc()`

Returns: `List[np.ndarray]`

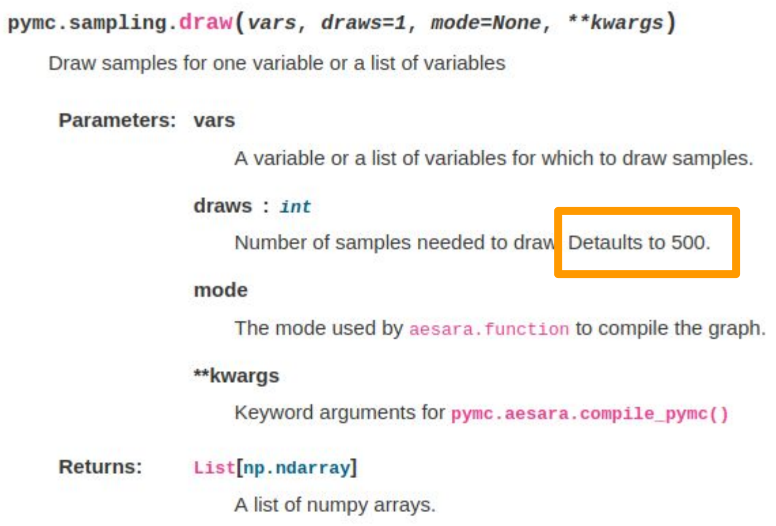
A list of numpy arrays.



Examples

```
import pymc as pm

# Draw samples for one variable
with pm.Model():
    x = pm.Normal("x")
x_draws = pm.draw(x, draws=100)
print(x_draws.shape)

# Draw 1000 samples for several variables
with pm.Model():
    x = pm.Normal("x")
    y = pm.Normal("y", shape=10)
    z = pm.Uniform("z", shape=5)
num_draws = 1000
# Draw samples of a list variables
draws = pm.draw([x, y, z], draws=num_draws)
assert draws[0].shape == (num_draws,)
assert draws[1].shape == (num_draws, 10)
assert draws[2].shape == (num_draws, 5)
```

Reviewing a first time contributor PR



```
pymc.sampling.draw(vars, draws=1, mode=None, **kwargs)
```

Draw samples for one variable or a list of variables

Parameters: **vars**

A variable or a list of variables for which to draw samples.

draws : *int*

Number of samples needed to draw. Detaults to 500.

mode

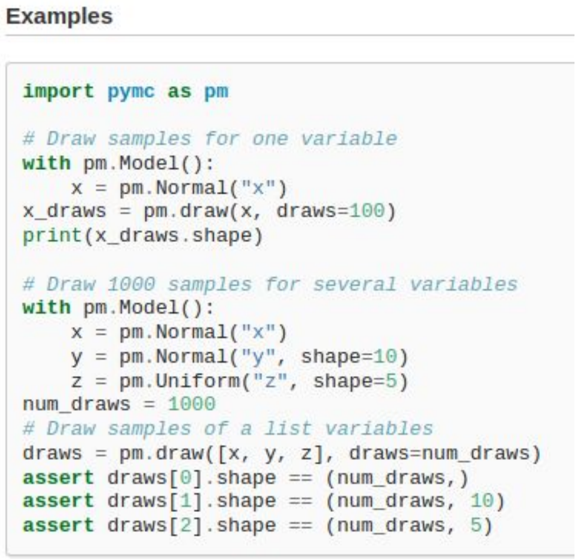
The mode used by `aesara.function` to compile the graph.

****kwargs**

Keyword arguments for `pymc.aesara.compile_pymc()`

Returns: `List[np.ndarray]`

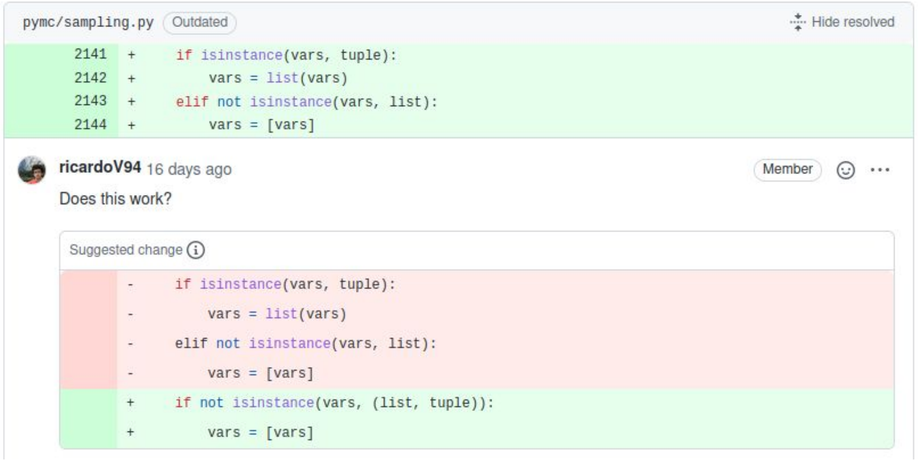
A list of numpy arrays.



Examples

```python
import pymc as pm

# Draw samples for one variable
with pm.Model():
    x = pm.Normal("x")
x_draws = pm.draw(x, draws=100)
print(x_draws.shape)

# Draw 1000 samples for several variables
with pm.Model():
    x = pm.Normal("x")
    y = pm.Normal("y", shape=10)
    z = pm.Uniform("z", shape=5)
num_draws = 1000
# Draw samples of a list variables
draws = pm.draw([x, y, z], draws=num_draws)
assert draws[0].shape == (num_draws,)
assert draws[1].shape == (num_draws, 10)
assert draws[2].shape == (num_draws, 5)
```

Reviewing a first time contributor PR



pymc/sampling.py Outdated

Hide resolved

```
+     if isinstance(vars, tuple):
+         vars = list(vars)
+     elif not isinstance(vars, list):
+         vars = [vars]
```

ricardoV94 16 days ago

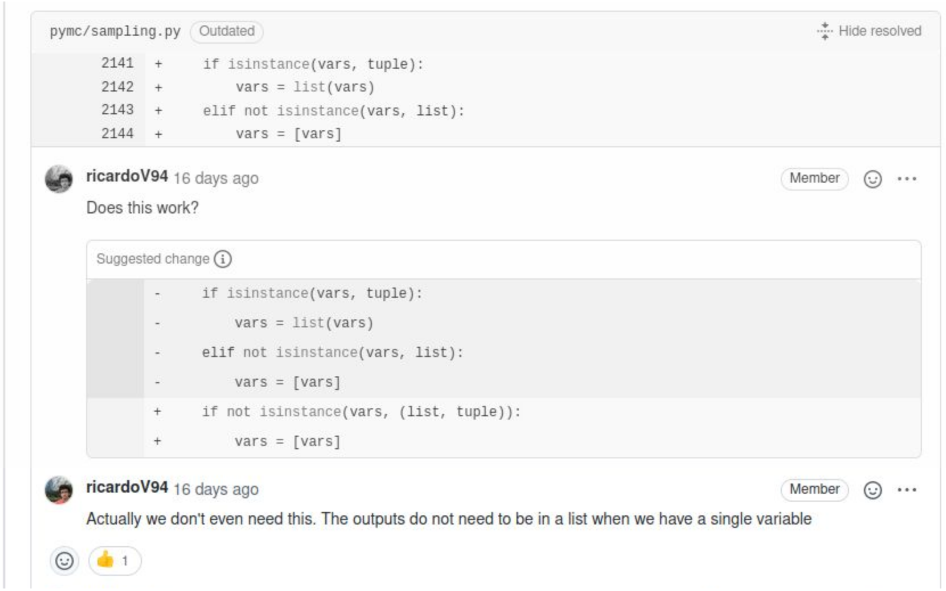
Member

Does this work?

Suggested change

```
-     if isinstance(vars, tuple):
-         vars = list(vars)
-     elif not isinstance(vars, list):
-         vars = [vars]
+     if not isinstance(vars, (list, tuple)):
+         vars = [vars]
```

Reviewing a first time contributor PR



pymc/sampling.py Outdated

Hide resolved

```
+    if isinstance(vars, tuple):
+        vars = list(vars)
+    elif not isinstance(vars, list):
+        vars = [vars]
```

ricardoV94 16 days ago

Member

Does this work?

Suggested change

```
-    if isinstance(vars, tuple):
-        vars = list(vars)
-    elif not isinstance(vars, list):
-        vars = [vars]
+    if not isinstance(vars, (list, tuple)):
+        vars = [vars]
```

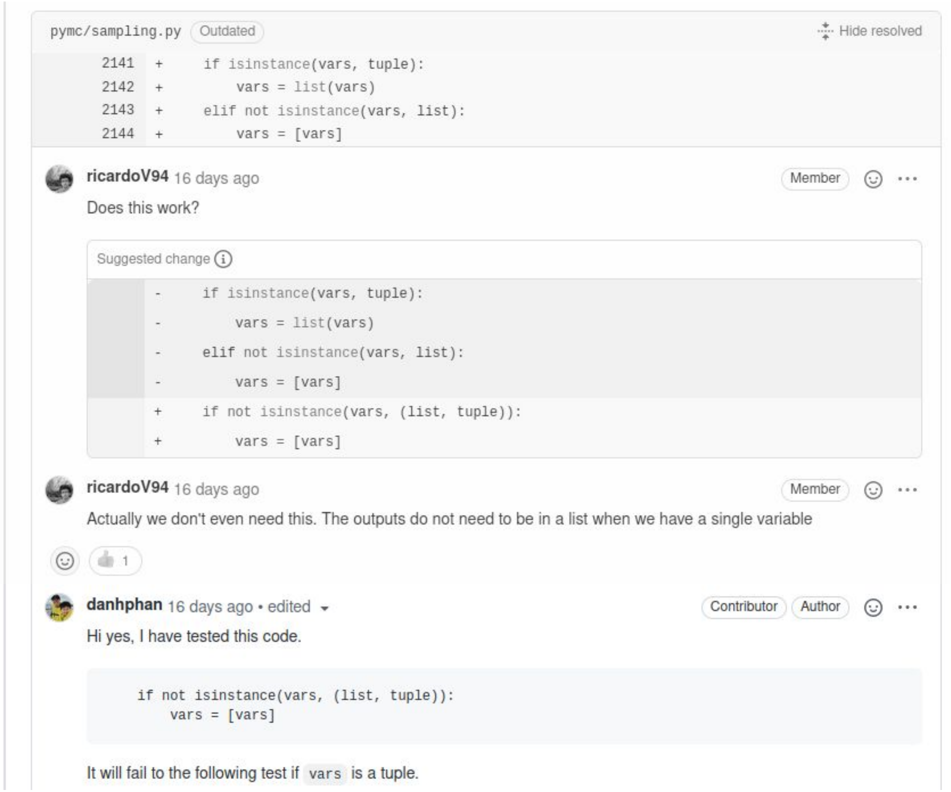
ricardoV94 16 days ago

Member

Actually we don't even need this. The outputs do not need to be in a list when we have a single variable

👍 1

Reviewing a first time contributor PR



pymc/sampling.py Outdated

Hide resolved

```
+    if isinstance(vars, tuple):
+        vars = list(vars)
+    elif not isinstance(vars, list):
+        vars = [vars]
```

ricardoV94 16 days ago Member

Does this work?

Suggested change

```
-    if isinstance(vars, tuple):
-        vars = list(vars)
-    elif not isinstance(vars, list):
-        vars = [vars]
+    if not isinstance(vars, (list, tuple)):
+        vars = [vars]
```

ricardoV94 16 days ago Member

Actually we don't even need this. The outputs do not need to be in a list when we have a single variable

1

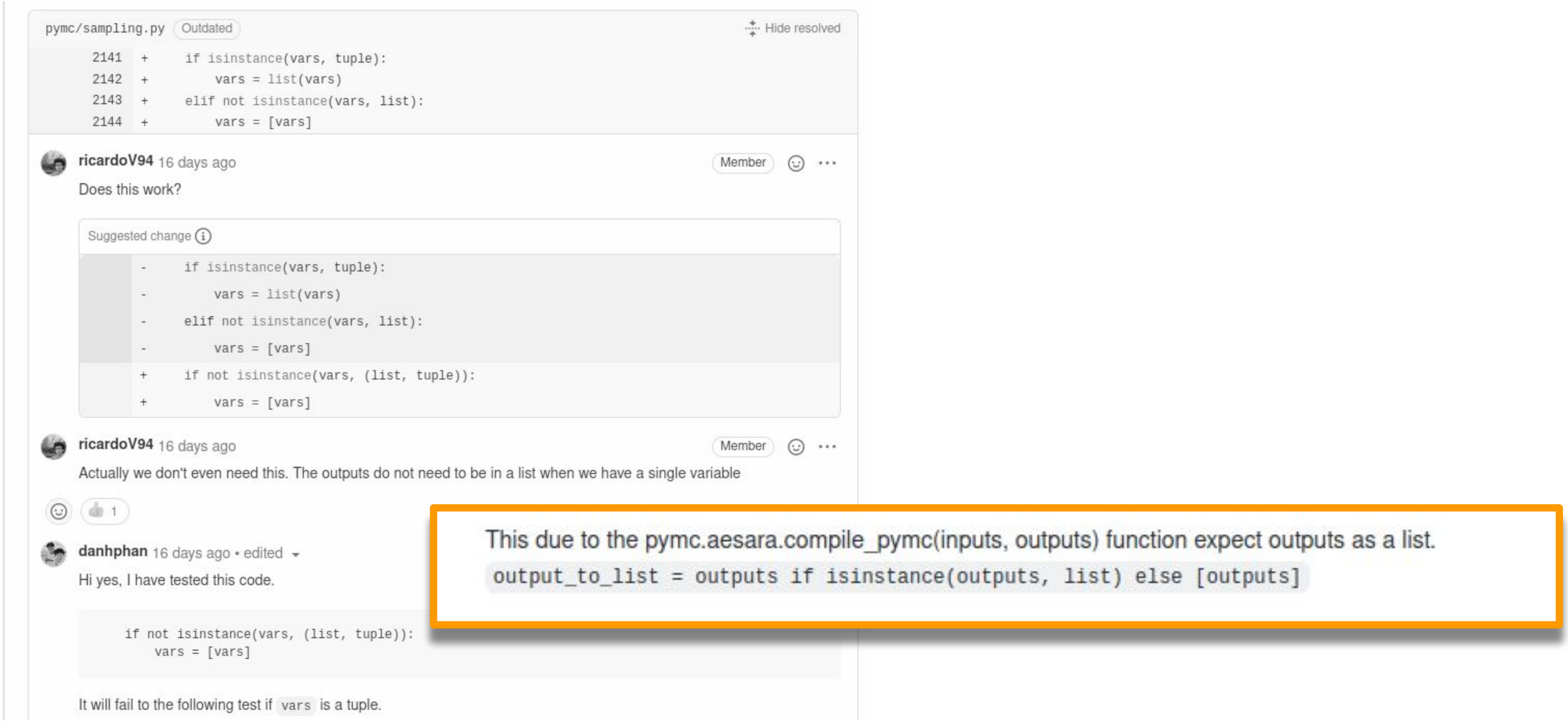
danhphan 16 days ago • edited Contributor Author

Hi yes, I have tested this code.

```
    if not isinstance(vars, (list, tuple)):
        vars = [vars]
```

It will fail to the following test if `vars` is a tuple.

Reviewing a first time contributor PR



pymc/sampling.py Outdated Hide resolved

```
+    if isinstance(vars, tuple):
+        vars = list(vars)
+    elif not isinstance(vars, list):
+        vars = [vars]
```

ricardoV94 16 days ago Member

Does this work?

Suggested change

```
-    if isinstance(vars, tuple):
-        vars = list(vars)
-    elif not isinstance(vars, list):
-        vars = [vars]
+    if not isinstance(vars, (list, tuple)):
+        vars = [vars]
```

ricardoV94 16 days ago Member

Actually we don't even need this. The outputs do not need to be in a list when we have a single variable

👍 1

danhphan 16 days ago • edited

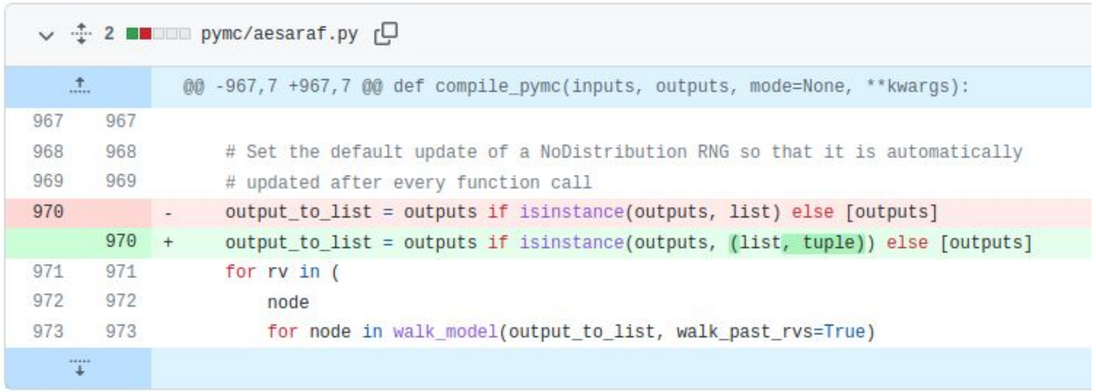
Hi yes, I have tested this code.

```
    if not isinstance(vars, (list, tuple)):
        vars = [vars]
```

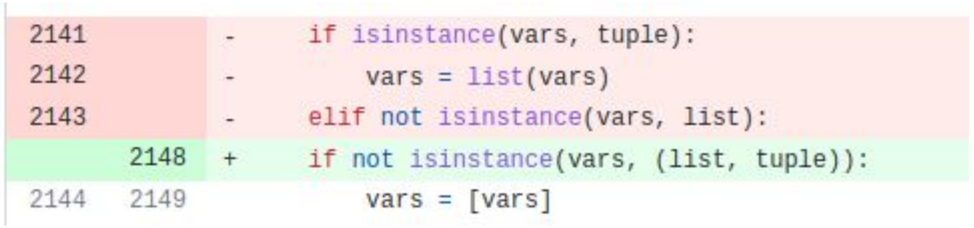
It will fail to the following test if `vars` is a tuple.

This due to the pymc.aesara.compile_pymc(inputs, outputs) function expect outputs as a list.
`output_to_list = outputs if isinstance(outputs, list) else [outputs]`

Reviewing a first time contributor PR

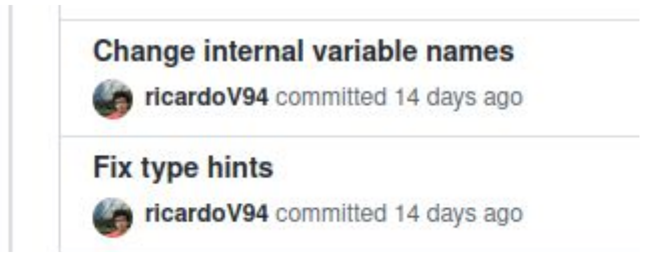


```
2 pymc/aesaraf.py

@@ -967,7 +967,7 @@ def compile_pymc(inputs, outputs, mode=None, **kwargs):
967  967
968  968         # Set the default update of a NoDistribution RNG so that it is automatically
969  969         # updated after every function call
970       -      output_to_list = outputs if isinstance(outputs, list) else [outputs]
     970  +      output_to_list = outputs if isinstance(outputs, (list, tuple)) else [outputs]
971  971         for rv in (
972  972             node
973  973             for node in walk_model(output_to_list, walk_past_rvs=True)
```



```
2141       -      if isinstance(vars, tuple):
2142       -          vars = list(vars)
2143       -      elif not isinstance(vars, list):
      2148 +      if not isinstance(vars, (list, tuple)):
2144  2149            vars = [vars]
```

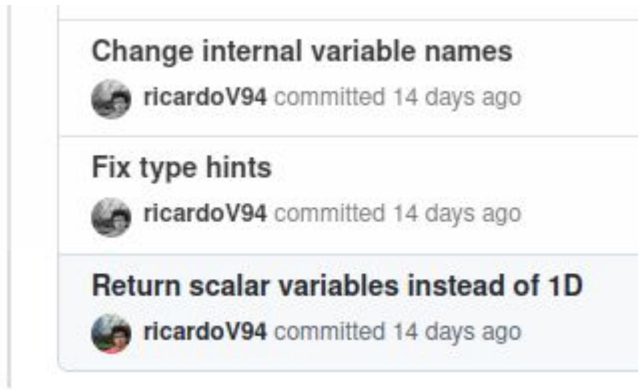
Reviewing a first time contributor PR


Change internal variable names

ricardoV94 committed 14 days ago

Fix type hints

ricardoV94 committed 14 days ago



Change internal variable names

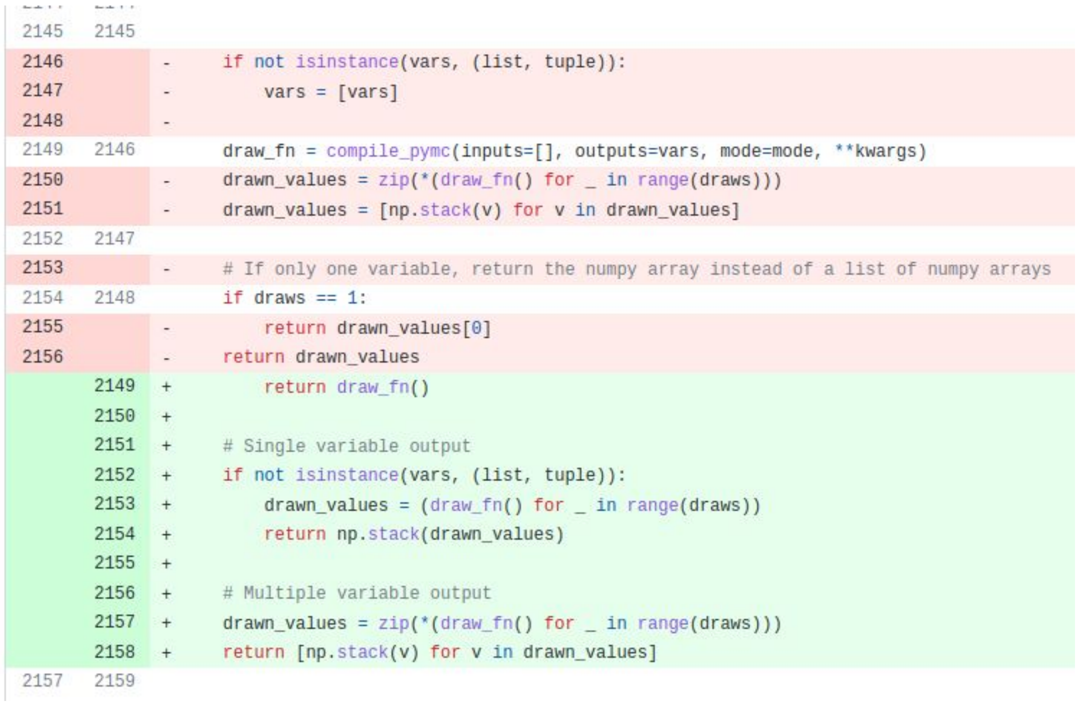
ricardoV94 committed 14 days ago

Fix type hints

ricardoV94 committed 14 days ago

Return scalar variables instead of 1D

ricardoV94 committed 14 days ago



```diff

-    if not isinstance(vars, (list, tuple)):
-        vars = [vars]
-
     draw_fn = compile_pymc(inputs=[], outputs=vars, mode=mode, **kwargs)
-    drawn_values = zip(*(draw_fn() for _ in range(draws)))
-    drawn_values = [np.stack(v) for v in drawn_values]

-    # If only one variable, return the numpy array instead of a list of numpy arrays
     if draws == 1:
-        return drawn_values[0]
-    return drawn_values
+        return draw_fn()
+
+    # Single variable output
+    if not isinstance(vars, (list, tuple)):
+        drawn_values = (draw_fn() for _ in range(draws))
+        return np.stack(drawn_values)
+
+    # Multiple variable output
+    drawn_values = zip(*(draw_fn() for _ in range(draws)))
+    return [np.stack(v) for v in drawn_values]

```

What is it like?

What is it like?

What is it like?

Still figuring it out

So you want to be an OSS contributor?

- Find a project you care about
- Engage with the community
- Be open to challenges
- Be ready to
  - learn
  - be wrong
  - make your case
- Take responsibility seriously
  - Your code will be used by many others
- Be patient
- Be polite

Thank you!