

Directional statistics with Stan

Nils Bertschinger

22 August 2018

Directional statistics

Directional statistics is an interesting subfield of statistics dealing with directional observations such as angles and higher-dimensional rotations. Angular data arise in many fields being it dates, e.g. hour of day, month of year, or angles for animal heading directions.

Mathematically, angles can be represented on the interval $[0, 2\pi)$ (or $[0, 360)$ when measured in degrees). Yet, some care has to be taken, just consider how you would compute the mean angle between 2 and 358 degrees.

Directional distributions

The density $p(\theta)$ of a circular distribution obeys the following properties for any $\theta \in \theta$

$$\begin{aligned} p(\theta) &\geq 0 \\ p(\theta) &= p(\theta + 2\pi) , \\ \int_{\theta}^{\theta+2\pi} p(x)dx &= 1 \end{aligned}$$

i.e. it is periodic and normalized over any interval of length 2π .

Common examples of circular distributions include the

- von Mises distribution: The density of the von Mises distribution is given as

$$p(\theta|\mu, \kappa) = \frac{e^{\kappa \cos(\theta - \mu)}}{2\pi I_0(\kappa)}$$

with center $\mu \in \mathbb{R}$ and concentration parameter $\kappa \in \mathbb{R}^+$. The normalization constant includes I_0 , i.e. the modified Bessel function of order 0.

- wrapped Cauchy distribution: The density of the wrapped Cauchy distribution is defined as

$$p(\theta|\theta_0, \gamma) = \sum_{n=-\infty}^{\infty} \frac{\gamma}{\pi(\gamma^2 + (\theta + 2n\pi - \theta_0)^2)} = \frac{1}{2\pi} \frac{\sinh \gamma}{\cosh \gamma - \cos(\theta - \theta_0)}$$

with center $\theta_0 \in \mathbb{R}$ and scale parameter $\gamma \in \mathbb{R}^+$. It is obtained by periodically wrapping a Cauchy distribution, hence its name.

Sampling the von Mises distribution

Suppose we want to sample from the von Mises distribution. It is readily available in Stan and thus, we would be tempted to write

```
data {
  real mu;
  real<lower=0> kappa;
}
parameters {
  real theta;
```

```

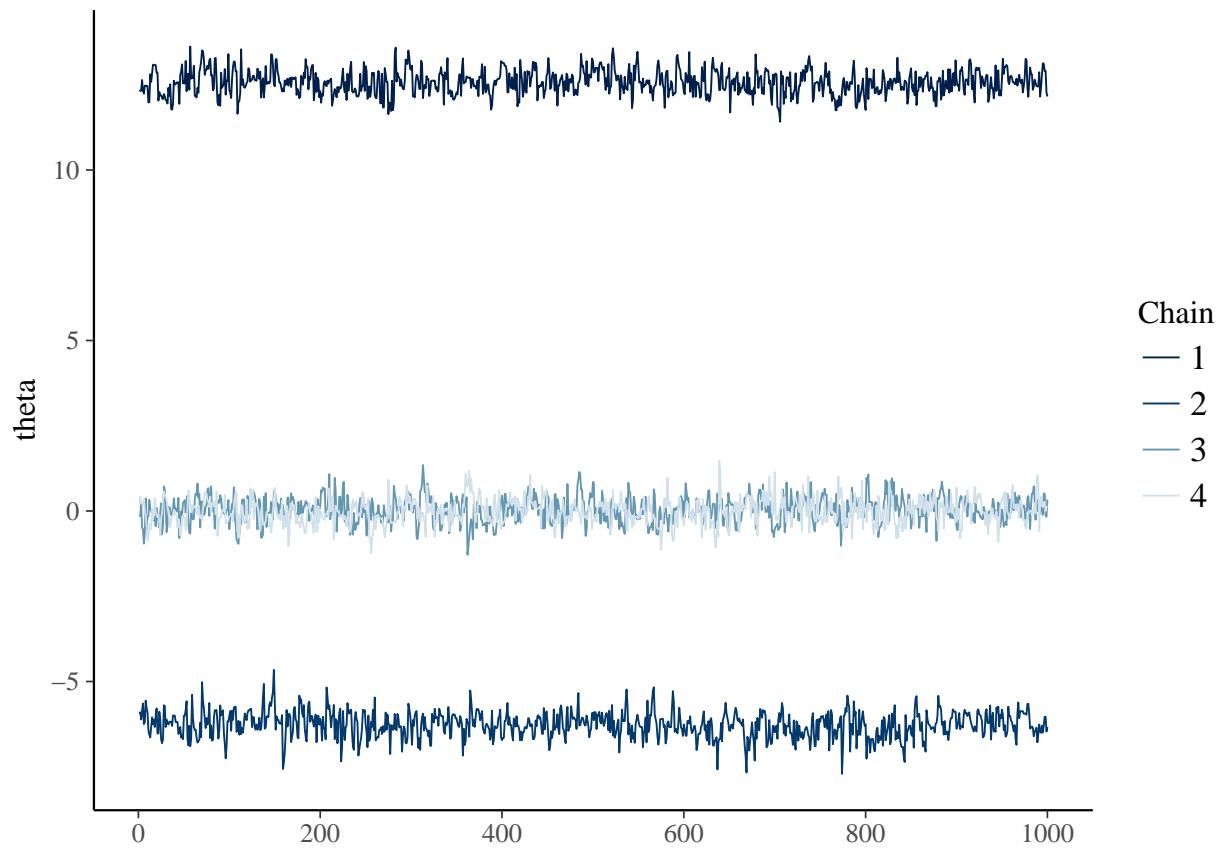
}

model {
  theta ~ von_mises(mu, kappa);
}

sample_1 <- sampling(model_1,
                      data = list(mu = 0, kappa = 8),
                      seed = 42, init_r = 5)

mcmc_trace(as.array(sample_1,
                     pars = "theta"))

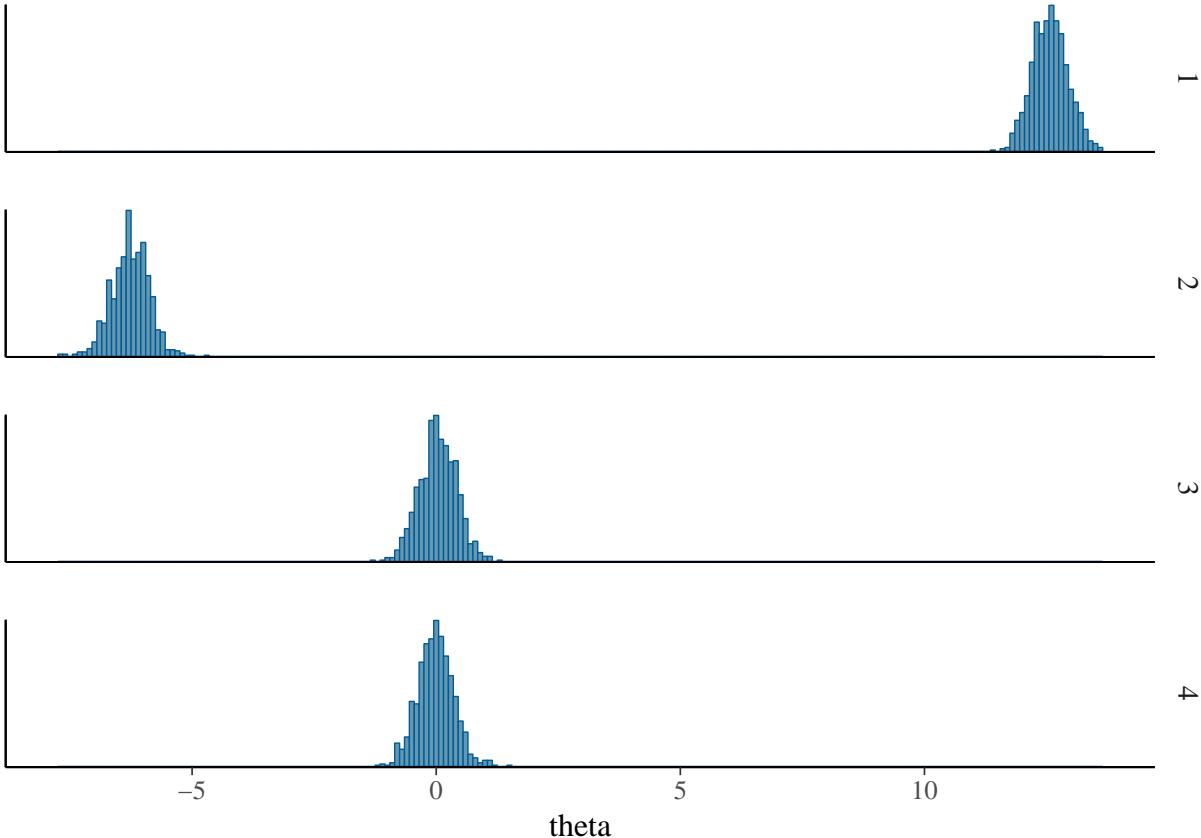
```



```

mcmc_hist_by_chain(as.array(sample_1,
                            pars = "theta"),
                            binwidth = 0.1)

```



This works, but leads to a highly multi-modal posterior due to the periodic density. We should restrict θ to some interval of length 2π , e.g. $(-\pi, \pi]$ as is also formally required in order to obtain a properly normalized density ...

```

data {
  real mu;
  real< lower = 0 > kappa;
}
parameters {
  real< lower = - pi(), upper = pi() > theta;
}
model {
  theta ~ von_mises(mu, kappa);
}

sample_2a <- sampling(model_2,
                      data = list(mu = 0, kappa = 8),
                      seed = 42)

print(sample_2a)

## Inference for Stan model: 3bf08ea08e553a5668a8686ed5de1efe.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##      mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff Rhat

```

```

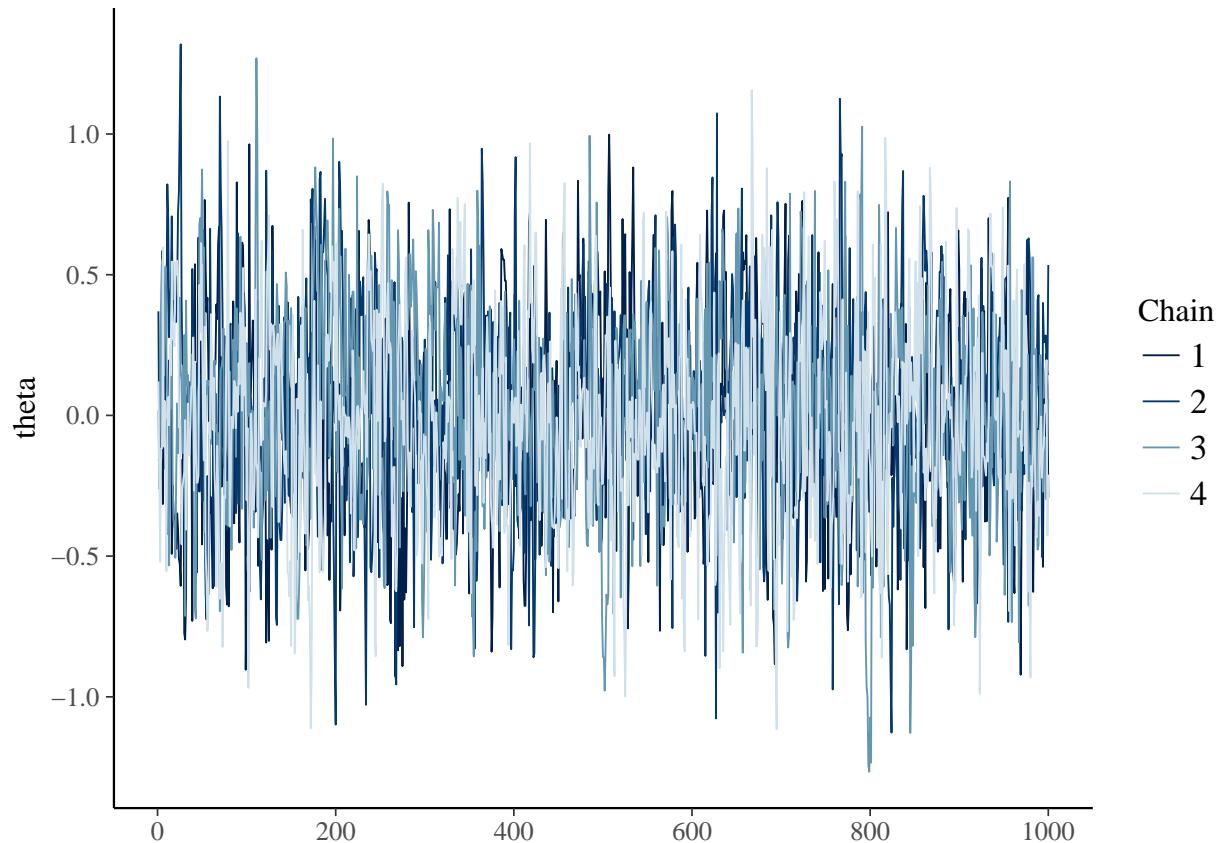
## theta 0.01    0.01 0.36 -0.72 -0.23 0.01 0.25   0.71   1106    1
## lp__  7.93    0.02 0.72  5.85  7.76 8.22 8.40   8.45   1479    1
##
## Samples were drawn using NUTS(diag_e) at Sat Aug 25 13:13:12 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).

```

```

mcmc_trace(as.array(sample_2a,
                     pars = "theta"))

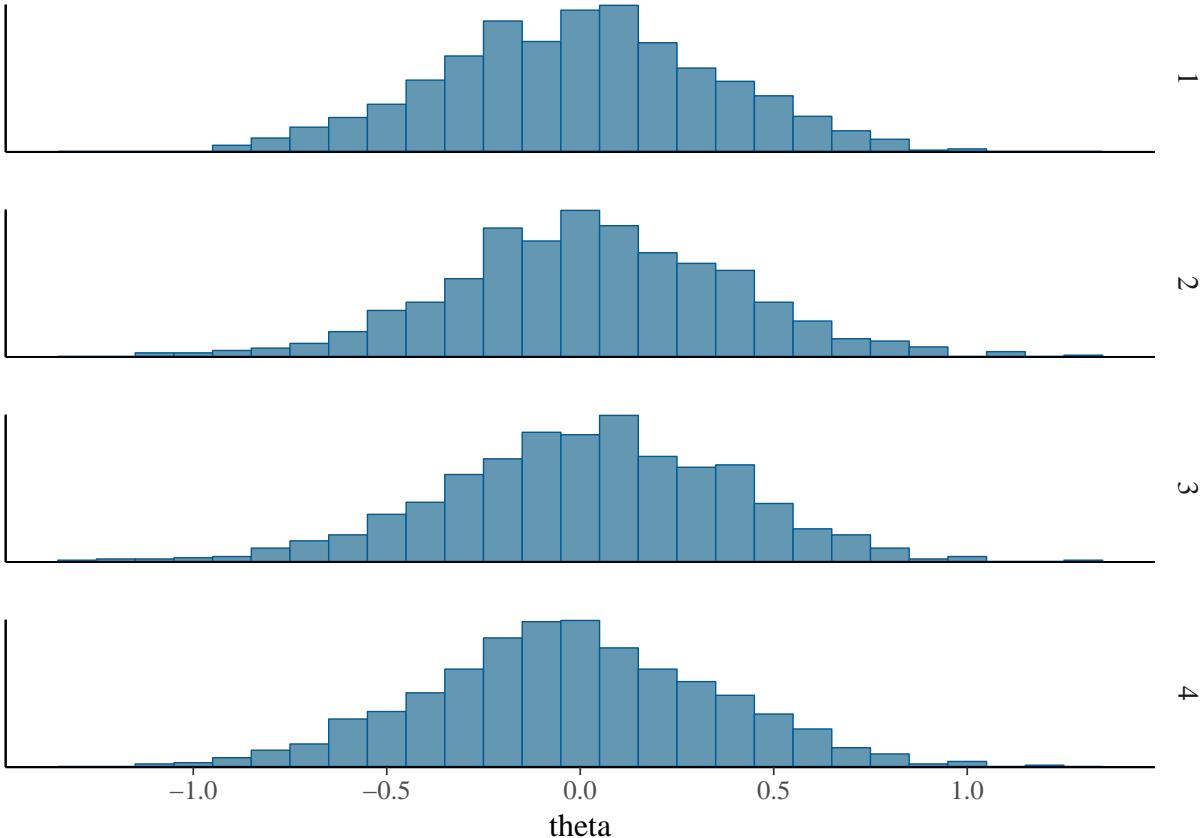
```



```

mcmc_hist_by_chain(as.array(sample_2a,
                            pars = "theta"),
                    binwidth = 0.1)

```



Unfortunately, the posterior can still become multi-modal if the density is centered close to the boundary of our (arbitrarily chosen) interval.

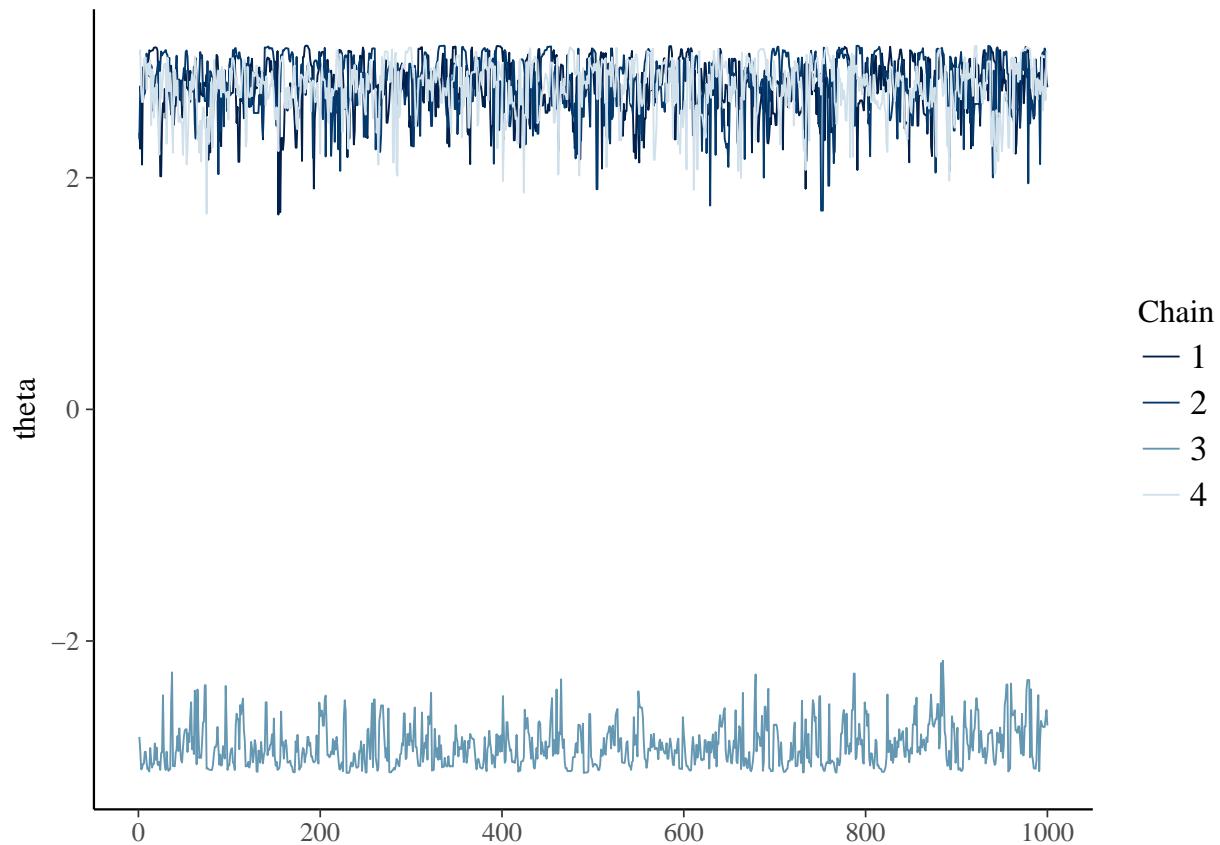
```
sample_2b <- sampling(model_2,
                      data = list(mu = 0.95*pi, kappa = 8),
                      seed = 42)

print(sample_2b)

## Inference for Stan model: 3bf08ea08e553a5668a8686ed5de1efe.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##      mean se_mean   sd  2.5%  25%  50%  75% 97.5% n_eff Rhat
## theta 1.36    1.74 2.48 -3.10  0.72  2.72  2.93  3.12     2 11.34
## lp__  5.98    0.28 0.86  3.78  5.62  6.19  6.68  6.79    10  1.11
##
## Samples were drawn using NUTS(diag_e) at Sat Aug 25 13:13:18 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

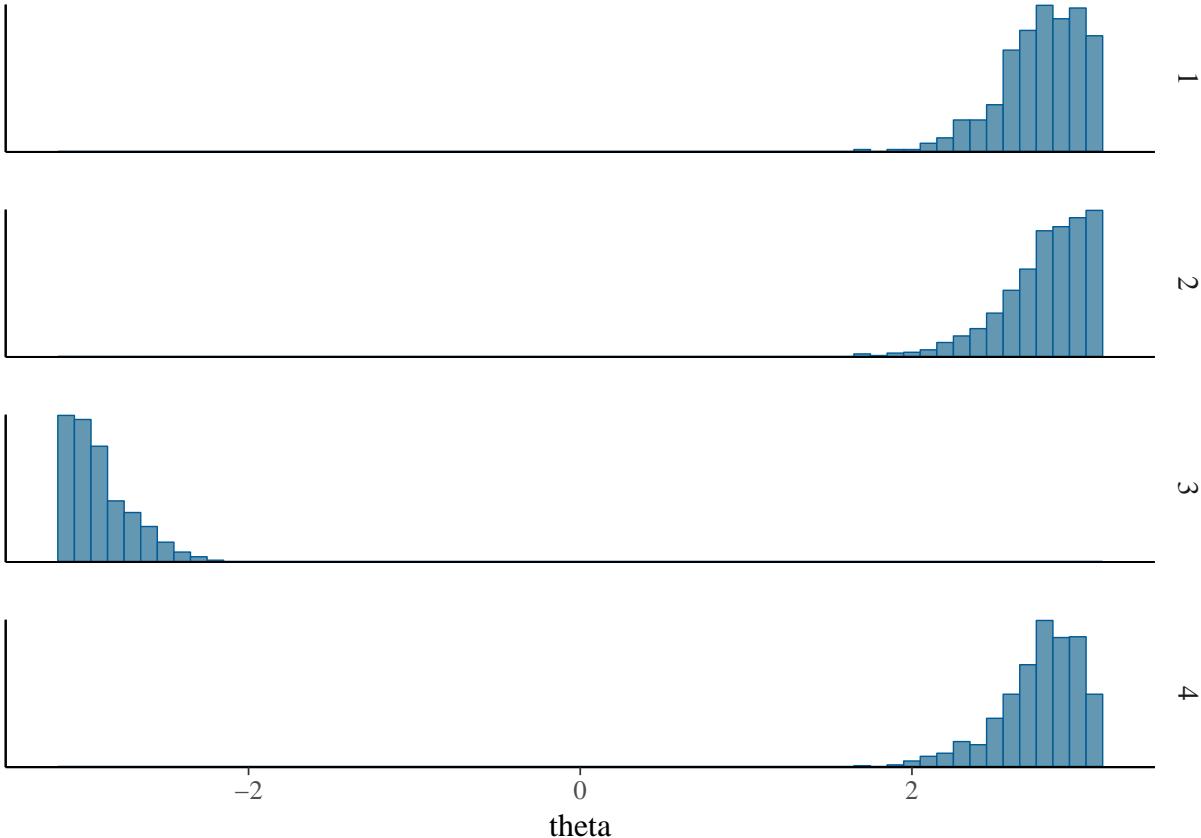
Here, the problem is that the sampler is unable to move across the interval boundaries. Thus, the actual circular space is not faithfully represented, a problem which would also arise if we take posterior averages (or do you think the above mean for θ is correct?).

```
mcmc_trace(as.array(sample_2b,  
                     pars = "theta"))
```



The multi-modality will not just mess up convergence diagnostics, but also bias the samples when some chain is stuck for a long time in a mode with small probability mass, i.e. the probability mass of the different modes will not be correct.

```
mcmc_hist_by_chain(as.array(sample_2b,  
                            pars = "theta"),  
                            binwidth = 0.1)
```



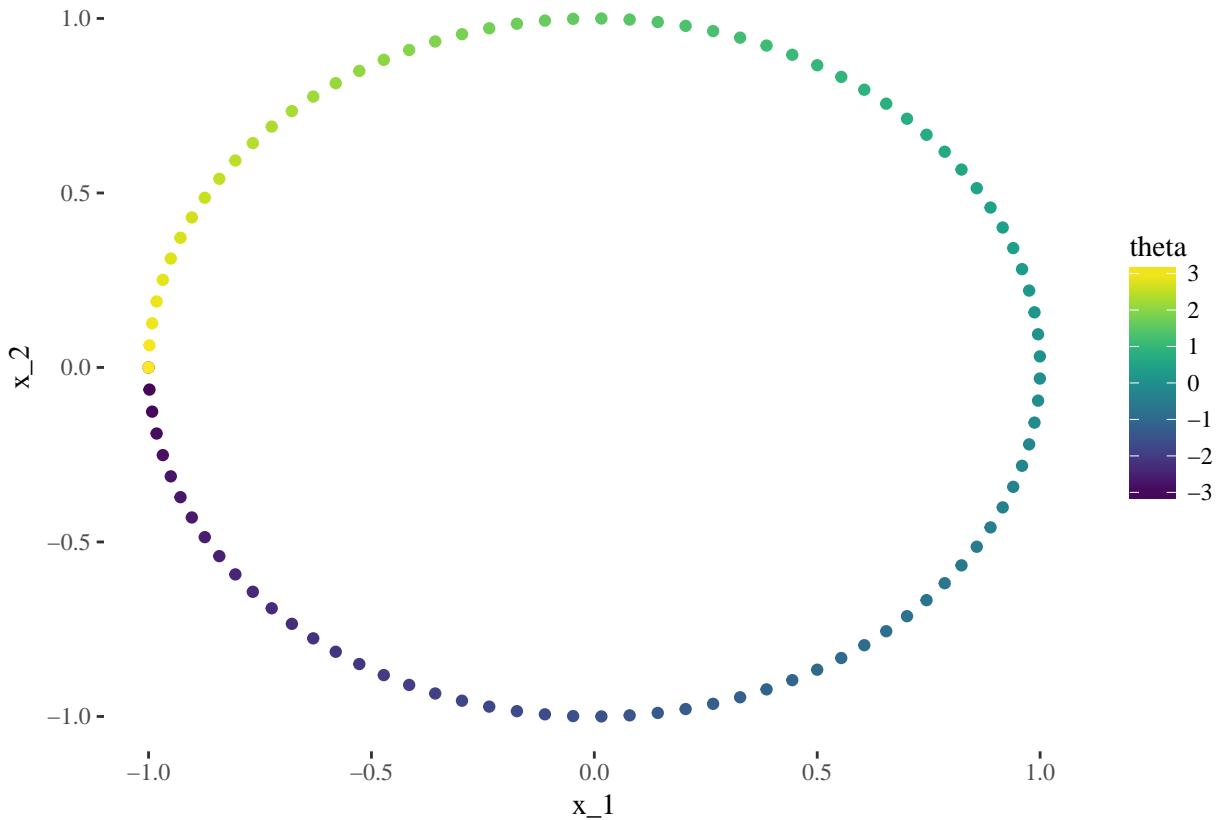
Unit-circle representation

A circular space is the simplest example of a non-Euclidean manifold. To faithfully represent its geometry we cannot use an interval on the real line (as already seen above). Instead, we can map an angle θ to a point on the unit circle via

$$\begin{aligned} x_1 &= \cos(\theta) \\ x_2 &= \sin(\theta) \end{aligned}$$

Note: This can also be expressed in terms of complex numbers where $\theta \mapsto \cos(\theta) + i \sin(\theta) = e^{i\theta}$, i.e. x_1 and x_2 corresponding to the real and imaginary part respectively.

```
tibble(theta = seq(-pi, pi, length.out = 100)) %>%
  mutate(x_1 = cos(theta), x_2 = sin(theta)) %>%
  ggplot(aes(x_1, x_2,
             color = theta)) +
  geom_point() +
  scale_color_viridis() +
  theme_tufte()
```



This trick is also suggested in the Stan manual: Reparameterize the model in terms of a unit vector and transform it to an angle.

```

data {
  real mu;
  real< lower = 0 > kappa;
}
parameters {
  unit_vector[2] x;
}
transformed parameters {
  real< lower = - pi(), upper = pi() > theta = atan2(x[2], x[1]);
}
model {
  theta ~ von_mises(mu, kappa);
}

sample_3 <- sampling(model_3,
  data = list(mu = 0.95*pi, kappa = 8),
  seed = 42)

```

```

## Warning: There were 3 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

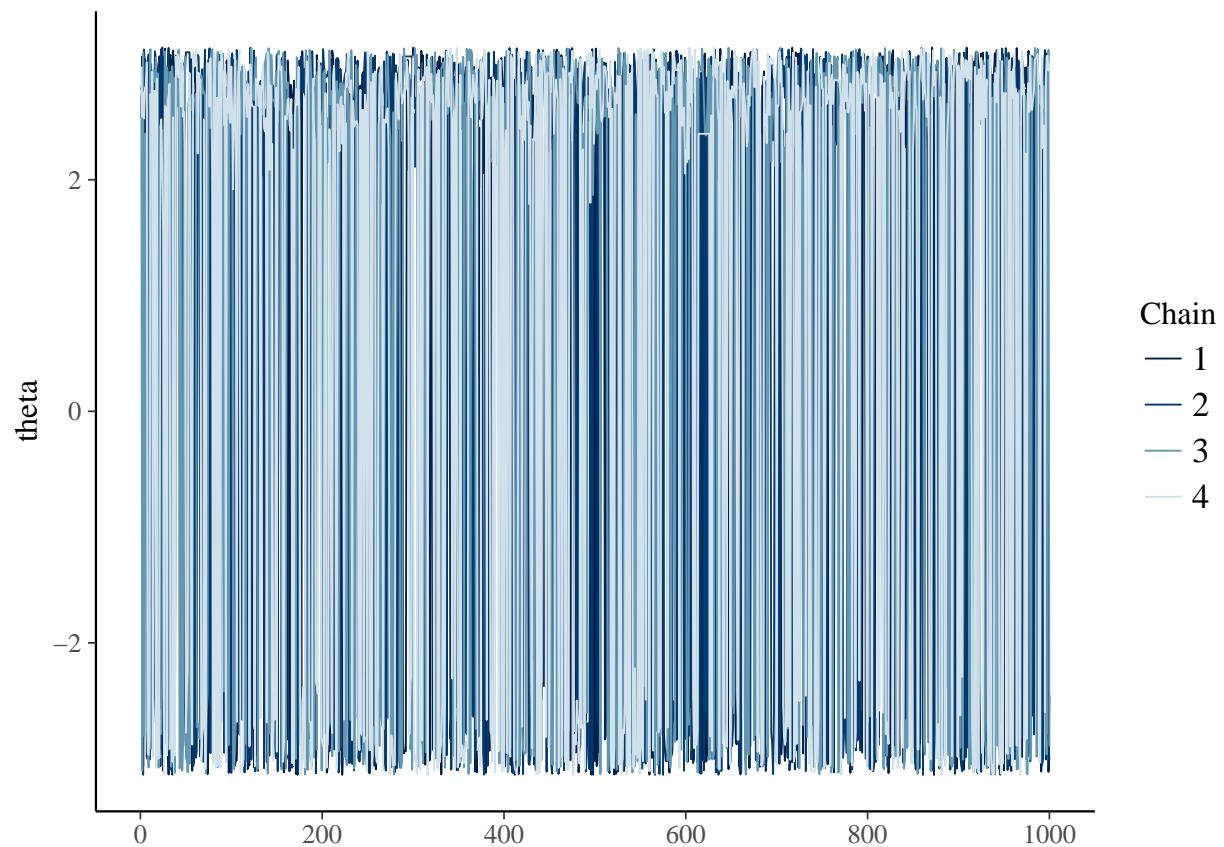
```

```

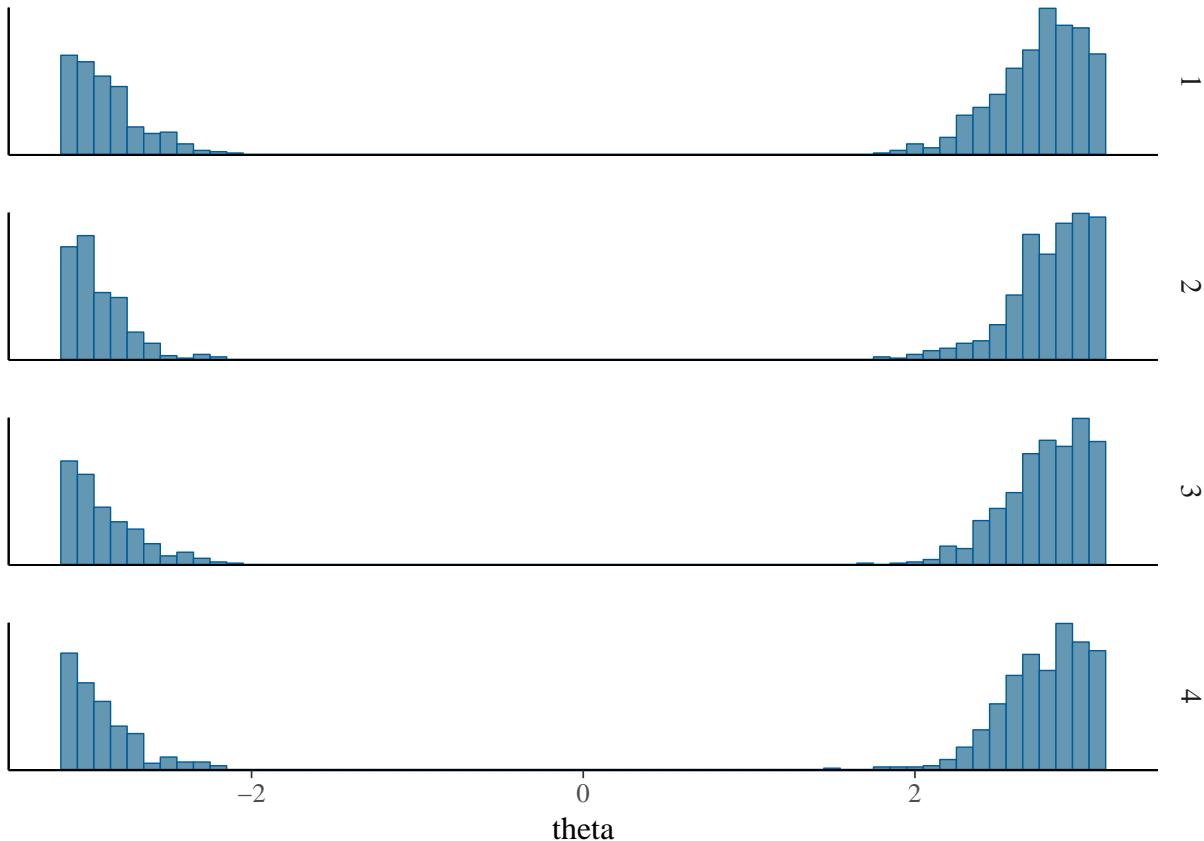
## Warning: Examine the pairs() plot to diagnose sampling problems

```

```
mcmc_trace(as.array(sample_3,
                     pars = "theta"))
```



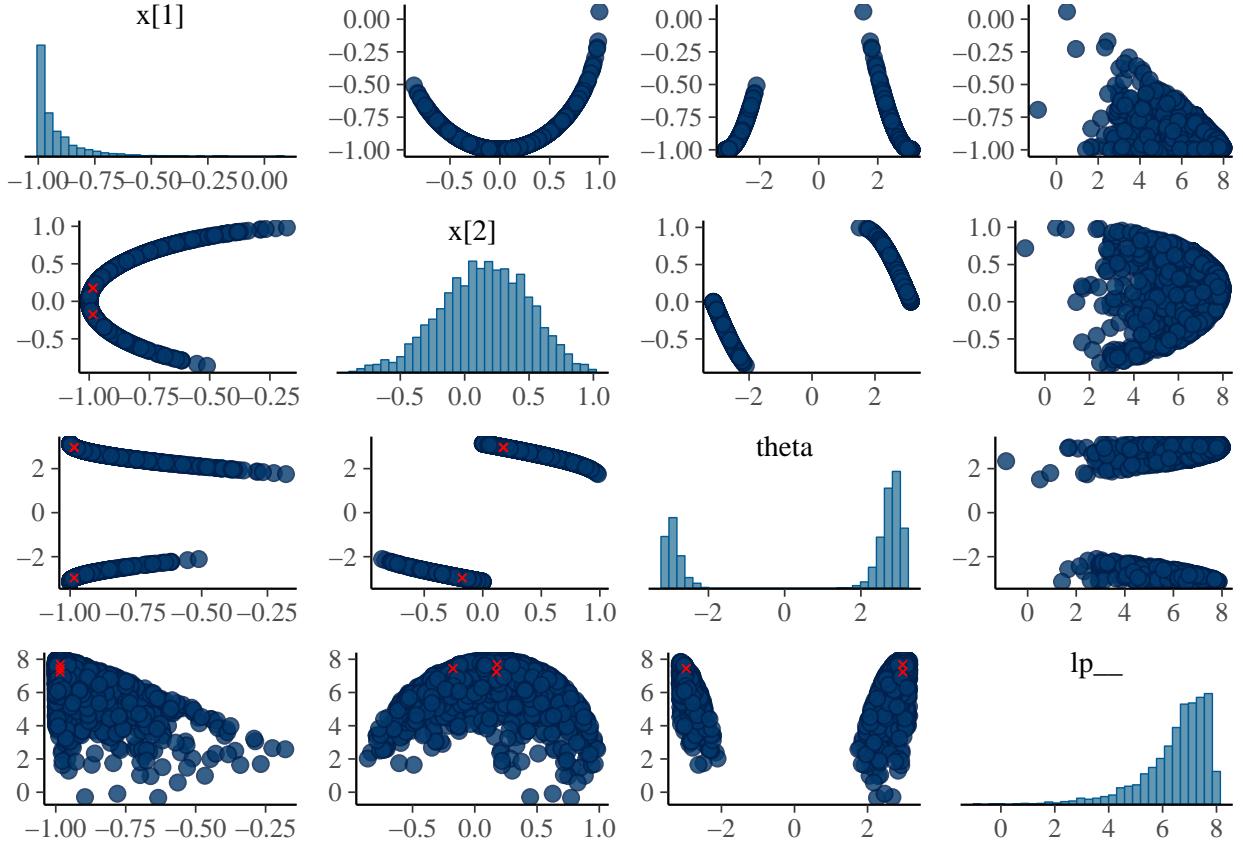
```
mcmc_hist_by_chain(as.array(sample_3,
                             pars = "theta"),
                             binwidth = 0.1)
```



Seems to work, but is it actually correct? Don't we need a Jacobian correction when putting a density on a transformed parameter?

Furthermore, we should inspect these divergent trajectories ...

```
mcmc_pairs(as.array(sample_3), np = nuts_params(sample_3))
```



It appears that problems arise if one of the x coordinates is close to zero, but this is not overly clear at this point ...

Polar coordinates

In order to better understand these issues let's formulate the above transformation more generally. Consider a two-dimensional vector $\mathbf{x} = (x_1, x_2)^T$. Now transform to polar coordinates given by

$$\begin{aligned} r &= \sqrt{x_1^2 + x_2^2} \\ \theta &= \text{atan2}(x_2, x_1) \end{aligned}$$

The angle θ is then again a circular parameter. Furthermore, the transformation is 1-to-1 and invertible except at the point $\mathbf{x} = \mathbf{0}$ where the angle is undefined. The inverse transformation is given by

$$x_1 = r \cos \theta, x_2 = r \sin \theta$$

with Jacobian

$$\begin{pmatrix} \frac{\partial x_1}{\partial r} & \frac{\partial x_1}{\partial \theta} \\ \frac{\partial x_2}{\partial r} & \frac{\partial x_2}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \cos \theta & -r \sin \theta \\ \sin \theta & r \cos \theta \end{pmatrix}$$

The absolute determinant of the Jacobian is therefore given as

$$|r \cos^2 \theta + r \sin^2 \theta| = r(\cos^2 \theta + \sin^2 \theta) = r$$

and therefore

$$p(r, \theta) = p(x_1, x_2)r \quad \text{or} \quad p(x_1, x_2) = \frac{1}{r}p(r, \theta)$$

Thus, we can use an unconstrained two-dimensional vector, transform it to polar coordinates, correct by the Jacobian of the transformation and define the desired circular distribution on the angle θ . Furthermore, the density on the angle is unaffected by whichever density we define on the vector length. As we know that the transformation is undefined at the origin, we can use a suitable density to keep r away from zero (and to further aid sampling in Stan give it a unit scale).

```

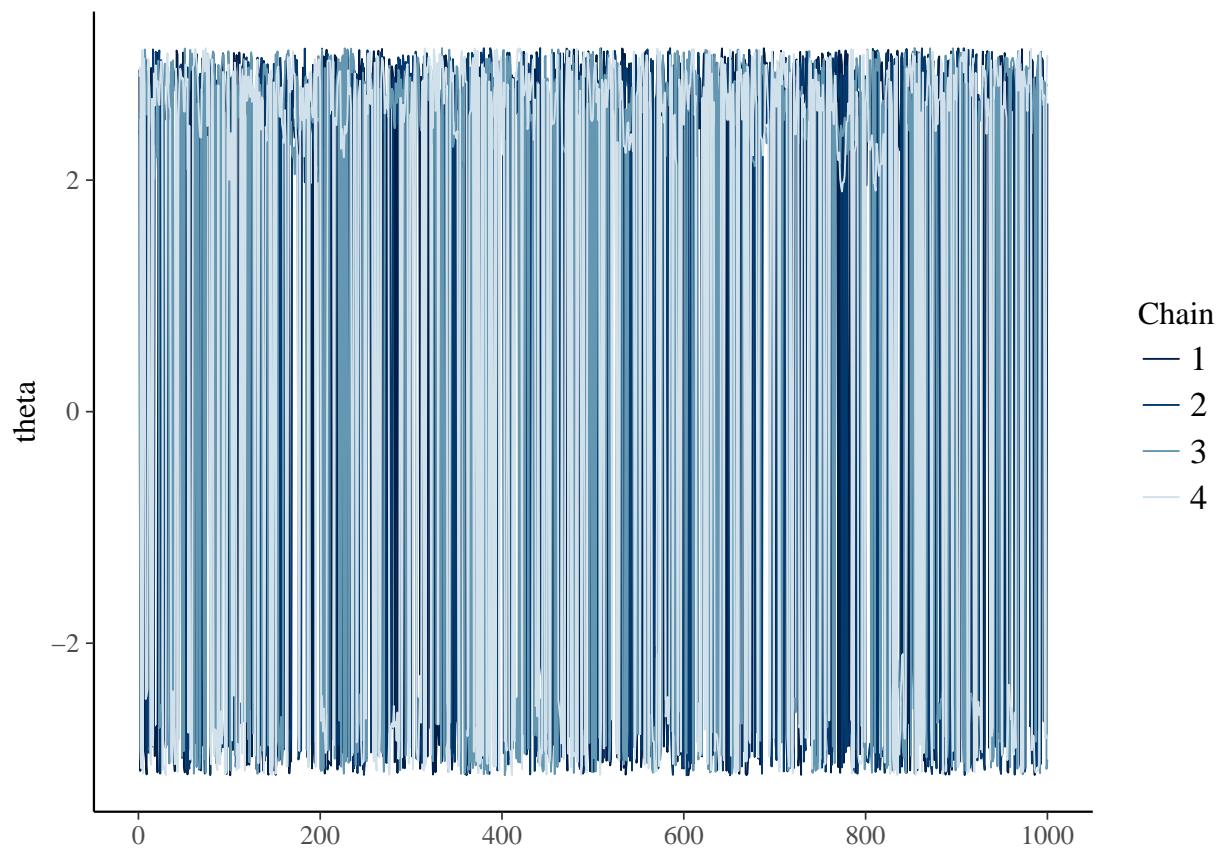
data {
  real mu;
  real< lower = 0 > kappa;
}
parameters {
  vector[2] x;
}
transformed parameters {
  real< lower = 0 > r = sqrt(dot_self(x));
  real< lower = - pi(), upper = pi() > theta = atan2(x[2], x[1]);
}
model {
  theta ~ von_mises(mu, kappa);
  // keep r close to unit length
  r ~ normal(1, 0.1);
  // correct for Jacobian of transformation
  target += - log(r); // log(1 / r)
}

sample_4 <- sampling(model_4,
                      data = list(mu = 0.95*pi, kappa = 8),
                      seed = 42)

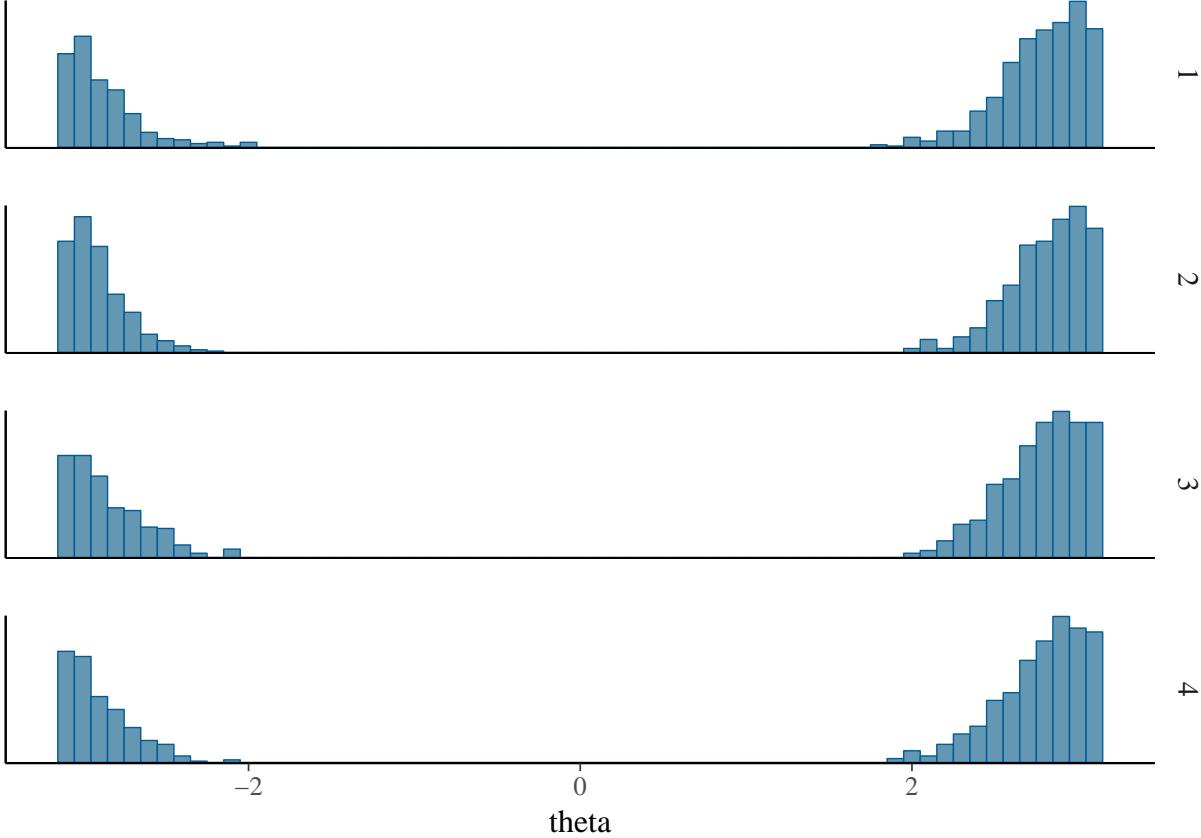
```

This works perfectly, i.e. the sampler is freely moving across the circular space without any divergent trajectories. It is actually freely moving across the larger two-dimensional space with its angle faithfully representing the geometry of the circular space.

```
mcmc_trace(as.array(sample_4,
                     pars = "theta"))
```



```
mcmc_hist_by_chain(as.array(sample_4,
                           pars = "theta"),
                           binwidth = 0.1)
```



We still need to take care when computing posterior averages, e.g. means, though. Also in this case, the geometry of the circular space has to be honored and simply taking Euclidean averages is not meaningful.

```
theta_4 <- rstan::extract(sample_4)$theta
mean(theta_4)

## [1] 0.8407073
```

Instead, moments of circular distributions are defined in terms of expectations over the position on the unit circle representing circular values. Formally, the n -th moment is defined as

$$\mathbb{E}[(e^{i\theta})^n] = \int_0^{2\pi} p(\theta)(e^{i\theta})^n d\theta$$

Correspondingly the sample mean (first moment) can be computed as

$$\frac{1}{N} \sum_i e^{i\theta_i}, \text{ i.e.}$$

the two-dimensional vectors, corresponding to the angles θ_i on the unit circle, are added (recall that addition of complex numbers works like addition of two-dimensional vectors). The resulting vector angle can then be considered as the mean angle. The R package *circular* defines useful functions to compute statistics over circular quantities.

```

library(circular)

print(mean.circular(theta_4))

## Warning in as.circular(x): an object is coerced to the class 'circular' using default value for the :
##   type: 'angles'
##   units: 'radians'
##   template: 'none'
##   modulo: 'asis'
##   zero: 0
##   rotation: 'counter'
## conversion.circularxradians

## Circular Data:
## Type = angles
## Units = radians
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
## [1] 2.995519

print(sd.circular(theta_4))

## [1] 0.3702522

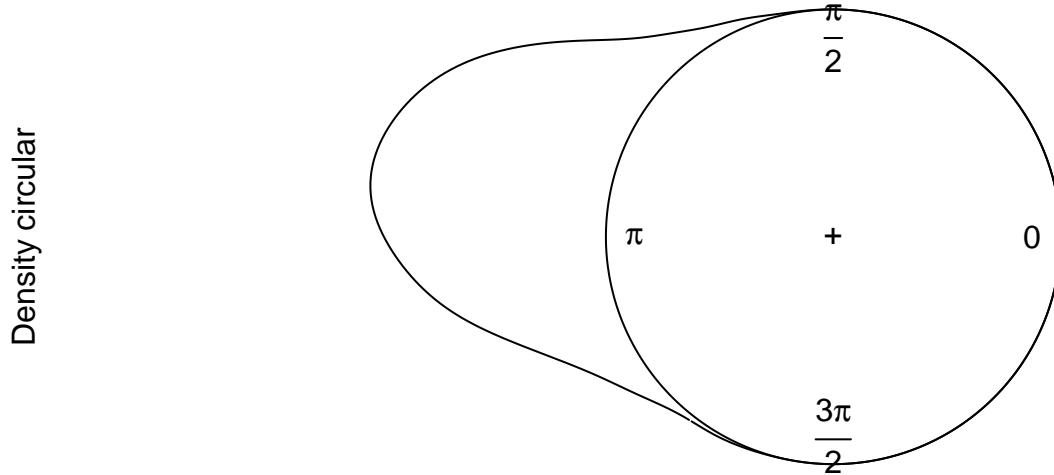
density.circular(theta_4, bw = 100) %>%
  plot(xlim = c(-2.5, 1))

## Warning in as.circular(x): an object is coerced to the class 'circular' using default value for the :
##   type: 'angles'
##   units: 'radians'
##   template: 'none'
##   modulo: 'asis'
##   zero: 0
##   rotation: 'counter'
## conversion.circularxradians0counter

## Warning in as.circular(x): an object is coerced to the class 'circular' using default value for the :
##   type: 'angles'
##   units: 'radians'
##   template: 'none'
##   modulo: 'asis'
##   zero: 0
##   rotation: 'counter'
## conversion.circularx$dataradians2pi

```

```
density.circular(x = theta_4, bw = 100)
```



N = 4000 Bandwidth = 100 Unit = radians

In the end, the above computations also explain why we don't need a Jacobian correction in the model parameterized in terms of a unit vector. In this case, the same transformation to polar coordinates is used, but as the length of the vector is fixed the corresponding constant Jacobian adjustment can just as well be dropped. The divergent trajectories presumably arise due to the internal implementation of the unit vector type. As far as I understand from the manual, it also uses an unconstrained vector which is then normalized. It appears that it is unable to keep the vector away from the origin though.

Example use case

TODO

Overall, I hope that you find the trick of using polar coordinates – with an informed prior on the length – useful when sampling from circular distributions.

```
devtools::session_info()
```

```
## Session info -----
##  setting  value
##  version  R version 3.3.3 (2017-03-06)
##  system   x86_64, linux-gnu
##  ui        X11
##  language (EN)
##  collate  de_DE.UTF-8
##  tz       Europe/Berlin
##  date     2018-08-25

## Packages -----
```

```

## package      * version date     source
## assertthat    0.2.0   2017-04-11 CRAN (R 3.3.3)
## backports     1.1.0   2017-05-22 CRAN (R 3.3.3)
## base         * 3.3.3  2017-03-11 local
## bayesplot     * 1.4.0   2017-09-12 CRAN (R 3.3.3)
## bindr          0.1     2016-11-13 CRAN (R 3.3.3)
## bindrcpp      * 0.2     2017-06-17 CRAN (R 3.3.3)
## boot          1.3-20   2017-07-30 CRAN (R 3.3.3)
## broom          0.4.2   2017-02-13 CRAN (R 3.3.3)
## cellranger    1.1.0   2016-07-27 CRAN (R 3.3.3)
## circular      * 0.4-93  2017-06-29 CRAN (R 3.3.3)
## cli            1.0.0   2017-11-05 CRAN (R 3.3.3)
## codetools      0.2-15  2016-10-05 CRAN (R 3.3.1)
## colorspace     1.3-2   2016-12-14 CRAN (R 3.3.3)
## compiler       3.3.3   2017-03-11 local
## crayon         1.3.4   2017-09-16 CRAN (R 3.3.3)
## datasets      * 3.3.3   2017-03-11 local
## devtools       1.13.5  2018-02-18 CRAN (R 3.3.3)
## digest          0.6.12  2017-01-27 CRAN (R 3.3.3)
## dplyr          * 0.7.4   2017-09-28 CRAN (R 3.3.3)
## evaluate       0.10.1  2017-06-24 CRAN (R 3.3.3)
##forcats        0.3.0   2018-02-19 cran (@0.3.0)
## foreign        0.8-69  2017-06-21 CRAN (R 3.3.3)
## ggplot2        * 2.2.1   2016-12-30 CRAN (R 3.3.3)
## ggthemes       * 3.4.0   2017-02-19 CRAN (R 3.3.3)
## glue            1.2.0   2017-10-29 cran (@1.2.0)
## graphics       * 3.3.3   2017-03-11 local
## grDevices      * 3.3.3   2017-03-11 local
## grid            3.3.3   2017-03-11 local
## gridExtra      2.2.1   2016-02-29 CRAN (R 3.3.0)
## gtable          0.2.0   2016-02-26 CRAN (R 3.3.0)
## haven           1.1.0   2017-07-09 CRAN (R 3.3.3)
## hms              0.4.0   2017-11-23 CRAN (R 3.3.3)
## htmltools       0.3.6   2017-04-28 CRAN (R 3.3.3)
## httr             1.2.1   2016-07-03 CRAN (R 3.3.1)
## inline           0.3.14  2015-04-13 CRAN (R 3.3.0)
## jsonlite         1.5     2017-06-01 CRAN (R 3.3.3)
## knitr            1.17    2017-08-10 CRAN (R 3.3.3)
## labeling          0.3     2014-08-23 CRAN (R 3.3.0)
## lattice          0.20-35 2017-03-25 CRAN (R 3.3.3)
## lazyeval         0.2.1   2017-10-29 cran (@0.2.1)
## lubridate        1.6.0   2016-09-13 CRAN (R 3.3.1)
## magrittr          1.5     2014-11-22 CRAN (R 3.3.0)
## memoise          1.1.0   2017-04-21 CRAN (R 3.3.3)
## methods         * 3.3.3   2017-03-11 local
## mnormt           1.5-5   2016-10-15 CRAN (R 3.3.1)
## modelr            0.1.1   2017-07-24 CRAN (R 3.3.3)
## munsell          0.4.3   2016-02-13 CRAN (R 3.3.0)
## mvtnorm          1.0-6   2017-03-02 CRAN (R 3.3.3)
## nlme             3.1-131 2017-02-06 CRAN (R 3.3.3)
## parallel          3.3.3   2017-03-11 local
## pillar            1.2.2   2018-04-26 cran (@1.2.2)
## pkgconfig        2.0.1   2017-03-21 CRAN (R 3.3.3)
## plyr              1.8.4   2016-06-08 cran (@1.8.4)

```

```
## psych      1.7.5   2017-05-03 CRAN (R 3.3.3)
## purrr      * 0.2.4   2017-10-18 cran (@0.2.4)
## R6          2.2.2   2017-06-17 CRAN (R 3.3.3)
## Rcpp         0.12.17  2018-05-18 CRAN (R 3.3.3)
## readr        * 1.1.1   2017-05-16 CRAN (R 3.3.3)
## readxl       1.0.0   2017-04-18 CRAN (R 3.3.3)
## reshape2     1.4.2   2016-10-22 cran (@1.4.2)
## rlang         0.2.0   2018-02-20 cran (@0.2.0)
## rmarkdown    * 1.6     2017-06-15 CRAN (R 3.3.3)
## rprojroot    1.2     2017-01-16 CRAN (R 3.3.3)
## rstan         * 2.17.3  2018-01-20 CRAN (R 3.3.3)
## rstudioapi   0.7     2017-09-07 CRAN (R 3.3.3)
## rvest         0.3.2   2016-06-17 CRAN (R 3.3.1)
## scales        0.5.0   2017-08-24 cran (@0.5.0)
## StanHeaders * 2.17.2  2018-01-20 CRAN (R 3.3.3)
## stats         * 3.3.3   2017-03-11 local
## stats4        3.3.3   2017-03-11 local
## stringi       1.1.7   2018-03-12 cran (@1.1.7)
## stringr       1.3.0   2018-02-19 cran (@1.3.0)
## tibble        * 1.4.2   2018-01-22 cran (@1.4.2)
## tidyverse     * 0.8.0   2018-01-29 cran (@0.8.0)
## tidyselect    0.2.4   2018-02-26 cran (@0.2.4)
## tidyverse     * 1.1.1   2017-01-27 CRAN (R 3.3.3)
## tools         3.3.3   2017-03-11 local
## utf8          1.1.3   2018-01-03 CRAN (R 3.3.3)
## utils         * 3.3.3   2017-03-11 local
## viridis       * 0.4.0   2017-03-27 CRAN (R 3.3.3)
## viridisLite  * 0.2.0   2017-03-24 CRAN (R 3.3.3)
## withr          2.1.2   2018-03-15 cran (@2.1.2)
## xml2          1.1.1   2017-01-24 CRAN (R 3.3.3)
## yaml          2.1.14   2016-11-12 cran (@2.1.14)
```