Example of the use of :dimarray and :dimarrays in read_samples().

Unfortunately this conflict with InferenceObjects.jl, hence it is currently not directly provided by StanSample.

```
html"""
<style>
    main {
        margin: 0 auto;
        max-width: 2000px;
        padding-left: max(160px,
10%);
        padding-right: max(160px,
30%);
    }
</style>
"""
```

```
using Pkg
```

```
begin
    using CSV    , DataFrames
    using Statistics    , Test
    using DimensionalData
    using StanSample
end
```

```
"/Users/rob/.julia/packages/StanSample/t
```

```
pkgdir(StanSample)
```

```
matrix (generic function with 3 methods)
```

```
include(joinpath(pkgdir(StanSample),
 "src", "utils", "dimarray.jl"))
```

```
df =
CSV.read(joinpath(pkgdir(StanSample)
, "data", "chimpanzees.csv"),
DataFrame);
```

```
# Define the Stan language model

stan10_4 = "
data{
    int N;
    int N_actors;
    int pulled_left[N];
    int prosoc_left[N];
    int condition[N];
    int actor[N];
}
parameters{
    vector[N_actors] a;
    real bp;
    real bpC;
}
model{
    vector[N] p;
    bpC ~ normal( 0 , 10 );
    bp ~ normal( 0 , 10 );
    a ~ normal( 0 , 10 );
    for ( i in 1:504 ) {
        p[i] = a[actor[i]] + (bp +
bpC * condition[i]) *
prosoc_left[i];
        p[i] = inv_logit(p[i]);
    }
    pulled_left ~ binomial( 1 , p );
}
";
```

```
data10_4 = (N = size(df, 1),
N_actors = length(unique(df.actor)),
    actor = df.actor, pulled_left =
df.pulled_left,
    prosoc_left = df.prosoc_left,
condition = df.condition);
```

```
# Sample using cmdstan

begin
    m10_4s = SampleModel("m10.4s",
    stan10_4)
    rc10_4s = stan_sample(m10_4s;
    data=data10_4);
end;
```

```
/var/folders/l7/pr04h0650q5dvqt
tnvs8s2c00000gn/T/jl_PFME6U/m10
.4s.stan updated.
```

```
4000×9 DimArray{Float64,2} draws with di
  iteration,
  param Categorical{Symbol} Symbol[a.1,
   Symbol("a.1")    Symbol("a.2")  …   S
 -0.506372        13.6865             2.
 -0.989641         4.06724            1.
 -0.808525        14.3335             1.
  ⋮                                  ⋱
 -0.921158         9.73791        …   1.
 -1.13568         11.2229             1.
 -0.438065         4.76962            1.
 -0.725417        17.8329             1.
```

```
if success(rc10_4s)
    read_samples(m10_4s, :dimarray)
end
```

```
1000×4 DimArray{Float64,2} draws with di
and reference dimensions:
  param Categorical{Symbol} Symbol[a.1]
 -0.506372  -1.01807   -0.938232  -0.737
 -0.989641  -0.724854  -0.603351  -0.914
 -0.808525  -0.623464  -0.691444  -1.130
 -0.849623  -0.950849  -0.869242  -1.235
 -0.529504  -0.542207  -0.29571   -0.869
  ⋮
 -0.572928  -0.507648  -0.647815  -0.441
 -1.14644   -1.00913   -0.842183  -0.921
 -0.79639   -0.614459  -0.944309  -1.135
 -0.69907   -0.640256  -0.978273  -0.438
 -0.77012   -0.646343  -0.704752  -0.725
```

```
if success(rc10_4s)
    da = read_samples(m10_4s,
    :dimarrays)
    da1 =
    da[param=At(Symbol("a.1"))]
end
```

```
Test Passed
```

```
if success(rc10_4s)
    # Other manipulations

    @test Tables.istable(da) == true

    # All of parameters
    dar = reshape(Array(da), 4000,
    9);
    @test size(dar) == (4000, 9)

    # Check :param axis names
    @test dims(da, :param).val ==
    vcat([Symbol("a.$i") for i in
    1:7], :bp, :bpC)

    # Test combining vector param
    'a'
    ma = matrix(da, "a");
    rma = reshape(ma, 4000, size(ma,
    3))
    @test mean(rma, dims=1) ≈ [-0.7
    10.9 -1 -1 -0.7 0.2 1.8]
    atol=0.7
end
```

```
4000×9 DimArray{Float64,2} draws with di
  iteration,
  param Categorical{Symbol} Symbol[a.1,
   Symbol("a.1")    Symbol("a.2")  …   S
 -0.506372         13.6865              2.
 -0.989641          4.06724             1.
 -0.808525         14.3335              1.
  ⋮                                    ⋱
 -0.921158          9.73791         …   1.
 -1.13568          11.2229              1.
 -0.438065          4.76962             1.
 -0.725417         17.8329              1.
```

```
if success(rc10_4s)
    da2 = read_samples(m10_4s,
    :dimarray)
    da2
end
```

Test Passed

```
if success(rc10_4s)
    da3=
    da2[param=At(Symbol("a.1"))]

    # Other manipulations

    @test Tables.istable(da3) ==
    true

    # Check :param axis names
    @test dims(da2, :param).val ==
    vcat([Symbol("a.$i") for i in
    1:7], :bp, :bpC)

    # Test combining vector param
    'a'
    ma3 = matrix(da2, "a");
    @test mean(ma3, dims=1) ≈ [-0.7
    10.9 -1 -1 -0.7 0.2 1.8]
    atol=0.7
end
```