Comparison with unmarked package

```
library(roccupy)
```

The goal of this vignette is to show how one can convert from the data format used in `unmarked` to the one used in `roccupy`, and that the maximum likelihood algorithm in `roccupy` closely matches the estimates from `occu` in `unmarked`.

First, load the example data in the `unmarked` package:

```
library(unmarked)
#> Loading required package: lattice
wt <- read.csv(system.file("csv","widewt.csv", package="unmarked"))
```

The data looks as follows:

```
head(wt)
#>   site y.1 y.2 y.3       elev       forest   length    date.1     date.2
#> 1    1   0   0   0 -1.1729446 -1.156228147 1.824549 -1.761481  0.3099471
#> 2    2   0   0   0 -1.1265010 -0.501483710 1.629241 -2.904339 -1.0471958
#> 3    3   0   0   0 -0.1976283 -0.101362109 1.458615 -1.690053 -0.4757672
#> 4    4   0   0   0 -0.1047411  0.007761963 1.686399 -2.190053 -0.6900529
#> 5    5   0   0   0 -1.0336137 -1.192602838 1.280934 -1.832910  0.1670899
#> 6    6   0   0   0 -0.8478392  0.917129237 1.808289 -2.618624  0.1670899
#>      date.3     ivel.1     ivel.2     ivel.3
#> 1 1.3813757 -0.5060353 -0.5060353 -0.5060353
#> 2 0.5956614 -0.9336151 -0.9907486 -1.1621491
#> 3 1.4528042 -1.1355754 -1.3388644 -1.6099164
#> 4 1.2385185 -0.8193481 -0.9272669 -1.1970640
#> 5 1.3813757  0.6375563  0.8803737  1.0422520
#> 6 1.3813757 -1.3288666 -1.0422624 -0.8989603
```

While `unmarked` uses what one might call a "wide" data format, `roccupy` uses a "long" format. We will have to melt the observation covariates. To do so, let us first pull out the columns that belong together:

```
site_covs <- wt[, c('elev', 'forest', 'length')]

obs_cov_1 <- wt[, c('site', 'date.1', 'date.2', 'date.3')]
obs_cov_2 <- wt[, c('site', 'ivel.1', 'ivel.2', 'ivel.3')]

y <- wt[, c('site', 'y.1', 'y.2', 'y.3')]

head(obs_cov_1)
#>   site    date.1     date.2    date.3
#> 1    1 -1.761481  0.3099471 1.3813757
#> 2    2 -2.904339 -1.0471958 0.5956614
#> 3    3 -1.690053 -0.4757672 1.4528042
#> 4    4 -2.190053 -0.6900529 1.2385185
#> 5    5 -1.832910  0.1670899 1.3813757
#> 6    6 -2.618624  0.1670899 1.3813757
```

Now, we *melt* each of them so that each row in the result corresponds to a single visit.

```
library(reshape2)

# Melt date covariate
melted_cov_1 <- melt(obs_cov_1, id.vars='site')
sites_1 <- melted_cov_1$site
vals_1 <- melted_cov_1$value

# Melt ivel covariate
melted_cov_2 <- melt(obs_cov_2, id.vars='site')
sites_2 <- melted_cov_2$site
vals_2 <- melted_cov_2$value

# Melt presence/absence
melted_y <- melt(y, id.vars='site')
sites_y <- melted_y$site
vals_y <- melted_y$value

# Double-check that the order appears unchanged:
all(sites_1 == sites_2)
#> [1] TRUE
all(sites_2 == sites_y)
#> [1] TRUE
```

The key motivation for using the long format is that it is *sparse*. Rather than having to fill missing visits with `NA`, they are simply omitted in the long format. We now put together all the observation-level information and discard the `NA` rows. Note that we subtract 1 from each site so that numbering is *zero-based* – this is because `roccupy` uses python under the hood, which uses zero-based indexing.

```
cur_data <- data.frame(cbind(date=vals_1, ivel=vals_2, site=sites_1 - 1, y=vals_y))

no_nas <- na.omit(cur_data)

# Check how many were removed
print(dim(cur_data))
#> [1] 711   4
print(dim(no_nas))
#> [1] 665   4
```

In this case, the savings are negligible. For citizen science data like eBird, however, they are considerable.

Now, we can pull out the arrays we need for the `roccupy` model:

```
X_obs <- no_nas[, c('date', 'ivel')]
checklist_cell_ids <- no_nas$site
X_env <- site_covs
y_checklist <- data.frame(cur_species=no_nas$y)
```

Now, fit the model:

```
result <- ssod_ml('forest + elev + length', 'date + ivel', X_env, X_obs,
                  y_checklist, checklist_cell_ids)
```

We can extract the estimated coefficients:

```
coef(result)
#> $env_coefs
```

```
#>  Intercept     forest       elev     length
#> -1.8893658  1.2124166  1.3794048  0.5252191
#>
#> $obs_coefs
#>  Intercept       date       ivel
#> 1.22950602 0.08734791 0.15325810
```

And we can compare this with what `unmarked` produces:

```
y <- wt[ ,2:4]

siteCovs <-  wt[ ,5:7]

obsCovs <- list(date = wt[ ,8:10],
                ivel = wt[ ,11:13])

umf <- unmarkedFrameOccu(y = y, siteCovs = siteCovs, obsCovs = obsCovs)

fm1 <- occu(formula = ~ date + ivel
                      ~ forest + elev + length,
            data = umf)
```

Agreement is very close:

```
print(coef(fm1, 'state'))
#>    psi(Int) psi(forest)   psi(elev) psi(length)
#>  -1.8901055   1.2132748   1.3803024   0.5253698
print(coef(fm1, 'det'))
#>     p(Int)    p(date)    p(ivel)
#> 1.22957382 0.08735256 0.15316974
```

Prediction works as follows:

```
# Predict suitability:
suit_pred <- predict(result, X_env, type='env')

# Predict probability of observing:
# Note that here X_env and X_obs lengths must match.
obs_pred <- predict(result, X_env[checklist_cell_ids + 1, ], X_obs, type='obs')

head(obs_pred)
#>   cur_species
#> 1  0.01379148
#> 2  0.02741241
#> 3  0.12800547
#> 4  0.17307471
#> 5  0.01257659
#> 6  0.18577793
```