

Comparing SSF and deepSSF Predictions

Scott Forrest

2025-02-13

In this script we are comparing the next-step ahead predictions of the SSF (with and without temporal dynamics) and deepSSF models. We are comparing the probabilities of movement, habitat selection and next-step selection, and how they change throughout time.

By comparing the predictions of each process across the entire tracking period or for each hour of the day, we can critically evaluate the covariates that are used by the models and allow for model refinement.

As we expected, the deepSSF models outperformed the SSF models on the in-sample data, which was particularly the case for when the model was trained with Sentinel-2 spectral bands and slope as the spatial covariates (deepSSF S2). The performance dropped for out-of-sample data for all models (including SSFs), and the deepSSF trained with the derived covariates (NDVI, canopy cover, herbaceous vegetation and slope) performed worse than the SSF models, and was only marginally better than a null model, which bears some evidence of overfitting. However, the deepSSF S2 model, trained on ‘raw’ Sentinel-2 layers rather than derived quantities, retained greater accuracy than all other approaches for out-of-sample data, suggesting that these inputs contain more information that is relevant to buffalo movement and habitat selection than derived quantities like NDVI and slope.

Table of contents

Loading packages	3
Script setup	3
Step selection function probabilities	3
SSF models fitted with and without temporal dynamics	3
Lengthening data frames to stack together for plotting	4
deepSSF probabilities	6
Lengthening data frames to stack together for plotting	6

deepSSF Sentinel 2 probabilities	6
Lengthening data frames to stack together for plotting	7
Compare the probabilities	7
Combine the data frames	7
Prepare data frame for plotting	8
Prepare sliding window	8
All IDs	9
Out-of-sample validation	10
Calculating mean and quantiles	10
Habitat selection across the tracking period	32
All models	32
deepSSF models	34
SSF models	36
Movement probability across the tracking period	39
All models	39
deepSSF models	41
SSF models	43
Next-step probabilities across the tracking period	46
All models	46
deepSSF models	48
SSF models	50
Hourly probabilities	52
Habitat selection across the day	54
All models	54
deepSSF models	56
SSF models	58
Movement probability across the day	61
All models	61
deepSSF models	63
SSF models	65
Next-step probability across the day	67
All models	67
deepSSF models	69
SSF models	71

Loading packages

```
library(tidyverse)
packages <- c("amt", "sf", "terra", "beepR", "tictoc", "circular",
             "matrixStats", "progress", "paletteer", "slider")
walk(packages, require, character.only = T)

options(scipen = 999) # Prevent scientific notation in plots
options(scipen = 0)  # Reset
```

Script setup

Specify the focal id for selecting the in-sample predictions.

```
focal_id <- 2005
```

Step selection function probabilities

SSF models fitted with and without temporal dynamics

```
# create vector of GPS data filenames
validation_ssf <- list.files(path = "outputs/next_step_validation", pattern = "next_step_pro
validation_ids <- substr(validation_ssf, 23, 26)

# import data
validation_ssf_list <- vector(mode = "list", length = length(validation_ssf))

for(i in 1:length(validation_ssf)){
  validation_ssf_list[[i]] <- read_csv(paste("outputs/next_step_validation/",
                                             validation_ssf[[i]],
                                             sep = ""))

  # validation_ssf_list[i]$id <- validation_ids[i]
  attr(validation_ssf_list[[i]]$t_, "tzone") <- "Australia/Queensland"
  attr(validation_ssf_list[[i]]$t2_, "tzone") <- "Australia/Queensland"

  print(sum(is.na(validation_ssf_list[[i]]$prob_next_step_ssf_0p)))
}
```

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

[1] 1

```
# check that the data has been imported correctly
# validation_ssf_list

validation_ssf_all <- bind_rows(validation_ssf_list)
```

Lengthening data frames to stack together for plotting

```

validation_ssf_move <- validation_ssf_all %>%
  dplyr::select(id, x_, y_, t_, x2_, y2_, t2_, hour_t2, yday_t2, year_t2, contains("prob_mov"),
    pivot_longer(cols = contains("movement"),
      names_to = "full_name",
      values_to = "value") %>%
  mutate(model = gsub("prob_movement_", "", full_name),
    probability = "move",
    .after = "full_name")

validation_ssf_habitat <- validation_ssf_all %>%
  dplyr::select(id, x_, y_, t_, x2_, y2_, t2_, hour_t2, yday_t2, year_t2, contains("prob_hab"),
    pivot_longer(cols = contains("habitat"),
      names_to = "full_name",
      values_to = "value") %>%
  mutate(model = gsub("prob_habitat_", "", full_name),
    probability = "habitat",
    .after = "full_name")

validation_ssf_next_step <- validation_ssf_all %>%
  dplyr::select(id, x_, y_, t_, x2_, y2_, t2_, hour_t2, yday_t2, year_t2, contains("prob_next"),
    pivot_longer(cols = contains("next_step"),
      names_to = "full_name",
      values_to = "value") %>%
  mutate(model = gsub("prob_next_step_", "", full_name),
    probability = "next_step",
    .after = "full_name")

validation_ssf_long <- bind_rows(validation_ssf_move,
  validation_ssf_habitat,
  validation_ssf_next_step)

head(validation_ssf_long)

```

```
# A tibble: 6 x 14
```

	id	x_	y_	t_	x2_	y2_	t2_	
	<dbl>	<dbl>	<dbl>	<dtm>	<dbl>	<dbl>	<dtm>	
1	2005	41969.	-1435671.	2018-07-25 11:04:23	41922.	-1.44e6	2018-07-25 12:04:39	
2	2005	41969.	-1435671.	2018-07-25 11:04:23	41922.	-1.44e6	2018-07-25 12:04:39	
3	2005	41922.	-1435654.	2018-07-25 12:04:39	41779.	-1.44e6	2018-07-25 13:04:17	
4	2005	41922.	-1435654.	2018-07-25 12:04:39	41779.	-1.44e6	2018-07-25 13:04:17	
5	2005	41779.	-1435601.	2018-07-25 13:04:17	41841.	-1.44e6	2018-07-25 14:04:39	
6	2005	41779.	-1435601.	2018-07-25 13:04:17	41841.	-1.44e6	2018-07-25 14:04:39	

```

# i 7 more variables: hour_t2 <dbl>, yday_t2 <dbl>, year_t2 <dbl>,
#   full_name <chr>, model <chr>, probability <chr>, value <dbl>

```

deepSSF probabilities

```
# create vector of GPS data filenames
validation_deepssf <- list.files(path = "outputs/next_step_validation", pattern = "next_step")
validation_ids <- substr(validation_deepssf, 25, 28)

# import data
validation_deepssf_list <- vector(mode = "list", length = length(validation_deepssf))

for(i in 1:length(validation_deepssf)){
  validation_deepssf_list[[i]] <- read_csv(paste("outputs/next_step_validation/",
                                                validation_deepssf[[i]],
                                                sep = ""))

  # validation_deepssf_list[i]$id <- validation_ids[i]
  attr(validation_deepssf_list[[i]]$t_, "tzone") <- "Australia/Queensland"
  attr(validation_deepssf_list[[i]]$t2_, "tzone") <- "Australia/Queensland"
}

# To check that the data has been imported correctly
# validation_deepssf_list

validation_deepssf_all <- bind_rows(validation_deepssf_list)
```

Lengthening data frames to stack together for plotting

```
validation_deepssf_long <- validation_deepssf_all %>%
  dplyr::select(id, x_, y_, t_, x2_, y2_, t2_, hour_t2, yday_t2, year_t2, contains("probs"))
  pivot_longer(cols = contains("probs"),
               names_to = "full_name",
               values_to = "value") %>%
  mutate(model = "deepSSF",
         probability = gsub("_probs", "", full_name),
         .after = "full_name")
```

deepSSF Sentinel 2 probabilities

```

# create vector of GPS data filenames
validation_deepssf_s2 <- list.files(path = "outputs/next_step_validation", pattern = "next_s
validation_ids <- substr(validation_deepssf_s2, 22, 25)

# import data
validation_deepssf_s2_list <- vector(mode = "list", length = length(validation_deepssf_s2))

for(i in 1:length(validation_deepssf_s2)){
  validation_deepssf_s2_list[[i]] <- read_csv(paste("outputs/next_step_validation/",
                                                    validation_deepssf_s2[[i]],
                                                    sep = ""))

  # validation_deepssf_s2_list[i]$id <- validation_ids[i]
  attr(validation_deepssf_s2_list[[i]]$t_, "tzone") <- "Australia/Queensland"
  attr(validation_deepssf_s2_list[[i]]$t2_, "tzone") <- "Australia/Queensland"
}

# To check that the data has been imported correctly
# validation_deepssf_s2_list

validation_deepssf_s2_all <- bind_rows(validation_deepssf_s2_list)

```

Lengthening data frames to stack together for plotting

```

validation_deepssf_s2_long <- validation_deepssf_s2_all %>%
  dplyr::select(id, x_, y_, t_, x2_, y2_, t2_, hour_t2, yday_t2, year_t2, contains("probs"))
  pivot_longer(cols = contains("probs"),
               names_to = "full_name",
               values_to = "value") %>%
  mutate(model = "deepSSF_S2",
         probability = gsub("_probs", "", full_name),
         .after = "full_name")

```

Compare the probabilities

Combine the data frames

```
validation_all_long <- bind_rows(validation_ssf_long,
                                validation_deepssf_long,
                                validation_deepssf_s2_long)

validation_all_long <- validation_all_long %>% filter(value > 0)
```

Prepare data frame for plotting

```
# Wet season is from November to April
months_wet <- c(1:4, 11:12)

validation_all_long <- validation_all_long %>%
  mutate(sample = ifelse(id == focal_id, "in_sample", "out_sample"),
         yday_t2_2018 = ifelse(year_t2 == 2018, yday_t2, yday_t2 + 365),
         week_t2 = week(t2_),
         month_t2 = month(t2_),
         season = ifelse(month_t2 %in% months_wet, "wet", "dry"))

head(validation_all_long)
```

```
# A tibble: 6 x 19
   id      x_      y_ t_      x2_      y2_ t2_
<dbl> <dbl> <dbl> <dtm> <dbl> <dbl> <dtm>
1  2005 41922. -1435654. 2018-07-25 12:04:39 41779. -1.44e6 2018-07-25 13:04:17
2  2005 41922. -1435654. 2018-07-25 12:04:39 41779. -1.44e6 2018-07-25 13:04:17
3  2005 41779. -1435601. 2018-07-25 13:04:17 41841. -1.44e6 2018-07-25 14:04:39
4  2005 41779. -1435601. 2018-07-25 13:04:17 41841. -1.44e6 2018-07-25 14:04:39
5  2005 41841. -1435635. 2018-07-25 14:04:39 41655. -1.44e6 2018-07-25 15:04:27
6  2005 41841. -1435635. 2018-07-25 14:04:39 41655. -1.44e6 2018-07-25 15:04:27
# i 12 more variables: hour_t2 <dbl>, yday_t2 <dbl>, year_t2 <dbl>,
#   full_name <chr>, model <chr>, probability <chr>, value <dbl>, sample <chr>,
#   yday_t2_2018 <dbl>, week_t2 <dbl>, month_t2 <dbl>, season <chr>
```

Prepare sliding window

The predicted probabilities are very noisy, so we apply some smoothing by using a sliding window (rolling mean).

We first need a function to calculate the summary statistics for each window.


```

window_summary <- function(data) {
  summarise(data,
    average_time = mean(t2_, na.rm = T),
    average_prob = mean(value, na.rm = T),
    q025 = quantile(value, probs = 0.025, na.rm = T),
    q25 = quantile(value, probs = 0.25, na.rm = T),
    q50 = quantile(value, probs = 0.5, na.rm = T),
    q75 = quantile(value, probs = 0.75, na.rm = T),
    q975 = quantile(value, probs = 0.975, na.rm = T)
  )
}

```

All IDs

```

window_width <- 15 # number of days in each window - should be odd
# how many observations before and after the current observation
before_after <- (window_width - 1) / 2

# ensure that the data is sorted by time (while respecting the id and model grouping)
all_data <- validation_all_long %>%
  group_by(id, model, probability) %>%
  arrange(t2_)

# apply the sliding window function
validation_all_sliding_period <- all_data %>%
  slide_period_dfr(
    all_data$t2_,
    # specify that we want to split by days (and slide across at daily intervals)
    "day",
    # our window function (calculates mean and quantiles for each window)
    window_summary,
    # how many days before and after the current observation we want to include in the window
    .before = before_after,
    .after = before_after
  )

head(validation_all_sliding_period, 10)

```

A tibble: 10 x 10

Groups: id, model [4]

id	model	probability	average_time	average_prob	q025	q25
<dbl>	<chr>	<chr>	<dtm>	<dbl>	<dbl>	<dbl>

```

1 2005 deepSSF habitat 2018-07-29 02:13:57 0.000107 4.60e-5 1.03e-4
2 2005 deepSSF move 2018-07-29 02:13:57 0.0917 2.01e-5 3.27e-4
3 2005 deepSSF next_step 2018-07-29 02:13:57 0.0922 1.17e-5 3.38e-4
4 2005 deepSSF_S2 habitat 2018-07-29 06:37:58 0.000152 3.60e-5 7.60e-5
5 2005 deepSSF_S2 move 2018-07-29 06:37:58 0.0866 1.61e-5 3.16e-4
6 2005 deepSSF_S2 next_step 2018-07-29 06:37:58 0.0895 7.31e-6 3.42e-4
7 2005 ssf_0p habitat 2018-07-29 06:37:58 0.000101 7.88e-5 9.57e-5
8 2005 ssf_0p move 2018-07-29 06:37:58 0.00167 6.77e-5 3.82e-4
9 2005 ssf_0p next_step 2018-07-29 06:37:58 0.00171 7.05e-5 3.80e-4
10 2005 ssf_2p habitat 2018-07-29 06:37:58 0.000107 6.79e-5 9.13e-5
# i 3 more variables: q50 <dbl>, q75 <dbl>, q975 <dbl>

```

We can see from the function above that we have the average time, average probability, and quantiles for each overlapping window, for each model and probability surface (habitat, movement and next-step).

Out-of-sample validation

The above sliding windows are for each individual separately, but we also want to calculate the average and quantiles for all out-of-sample individuals together.

Calculating mean and quantiles

```

# out-of-sample data - all ids but the focal id
OOS_data <- validation_all_long %>%
  filter(!id == focal_id) %>%
  group_by(model, probability) %>%
  arrange(t2_)

validation_all_sliding_period_OOS <- OOS_data %>%
  slide_period_dfr(
    OOS_data$t2_,
    "day",
    window_summary,
    .before = before_after,
    .after = before_after
  )

```

``summarise()`` has grouped output by 'model'. You can override using the ``.groups`` argument.
``summarise()`` has grouped output by 'model'. You can override using the ``.groups`` argument.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
`summarise()` has grouped output by 'model'. You can override using the
`.groups` argument.
```

```
head(validation_all_sliding_period_OOS, 10)
```

```
# A tibble: 10 x 9
# Groups:   model [4]
  model probability average_time average_prob q025 q25 q50
  <chr> <chr> <dtm> <dbl> <dbl> <dbl> <dbl>
1 deepSSF habitat 2018-07-29 06:09:54 0.000102 2.67e-5 7.51e-5 1.04e-4
2 deepSSF move 2018-07-29 06:09:54 0.0600 1.30e-5 3.02e-4 2.03e-3
3 deepSSF next_step 2018-07-29 06:09:54 0.0594 8.58e-6 2.84e-4 1.93e-3
4 deepSSF~ habitat 2018-07-29 06:09:54 0.000119 8.44e-6 5.32e-5 9.43e-5
5 deepSSF~ move 2018-07-29 06:09:54 0.0774 8.19e-6 2.74e-4 1.98e-3
6 deepSSF~ next_step 2018-07-29 06:09:54 0.0752 4.24e-6 2.33e-4 1.85e-3
7 ssf_0p habitat 2018-07-29 06:09:54 0.0000982 7.17e-5 8.64e-5 9.65e-5
8 ssf_0p move 2018-07-29 06:09:54 0.00174 3.70e-5 3.85e-4 1.24e-3
9 ssf_0p next_step 2018-07-29 06:09:54 0.00176 3.63e-5 3.79e-4 1.27e-3
10 ssf_2p habitat 2018-07-29 06:09:54 0.000105 5.50e-5 8.60e-5 1.01e-4
# i 2 more variables: q75 <dbl>, q975 <dbl>
```

Now the summaries are calculated for all out-of-sample individuals together.

Habitat selection across the tracking period

All models

The solid coloured lines show the average probability for the focal individual that the model was fitted to, and the shaded ribbon is the 50% quantile (there is high variability between probability values, so for clarity we omitted the 95% quantiles). The thin coloured lines are the average probability values for 12 individuals that the model was not fitted to, and are therefore out-of-sample validation data. The dashed coloured lines are the mean values for each model for all of the out-of-sample individuals.

We also show the ‘null’ probability, i.e. if the selection was completely random, which is just the probability divided equally between all cells.

```
# if there were uniform probabilities (i.e. no selection)
uniform_prob <- 1/(101*101)

ribbon_50_alpha <- 0.2
primary_linewidth <- 0.5
OOS_mean_linewidth <- 0.5
secondary_linewidth <- 0.04

ggplot() +

  # dashed lines containing the SSF probabilities (for zooming into in the paper)
  geom_hline(yintercept = 0.6e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
  geom_hline(yintercept = 1.5e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

  # in sample 50% ribbon
  geom_ribbon(data = validation_all_sliding_period %>%
    filter(probability == "habitat" &
      id == focal_id),
    aes(x = average_time,
      ymin = q25,
      ymax = q75,
      fill = model,
      group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

  # out-of-sample thin line for each individual
  geom_line(data = validation_all_sliding_period %>%
    filter(probability == "habitat" &
```



```

        !id == focal_id),
    aes(x = average_time,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
    filter(probability == "habitat"),
    aes(x = average_time,
        y = average_prob,
        colour = model),
    linewidth = OOS_mean_linewidth,
    linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
    filter(probability == "habitat" &
        id == focal_id),
    aes(x = average_time,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
    values = c("#E25834", "#000000", "#0096B5", "#26185F"),
    labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
    values = c("#E25834", "#000000", "#0096B5", "#26185F"),
    labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +
theme_bw() +
theme(legend.position = "bottom",
    axis.text.x = element_text(angle = 30, hjust = 1))

```



```
# ggsave(paste0("outputs/validation_all_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

The first thing to note is difference in magnitude between the deepSSF and the SSF predictions. The deepSSF predicted probabilities are often much higher, but can also be much lower, suggesting that the deepSSF models are more ‘confident’.

The deepSSF and deepSSF S2 models both performed particularly well between December 2018 and July 2019, (wet-season and early dry-season), although only the deepSSF S2 model performs well outside of this period (for most of the dry season). This suggests that the derived covariates may lack information that is relevant to buffalo during this period, such as a representation of water.

The higher performance of the deepSSF S2 predictions is also echoed in the out-of-sample predictions, which are generally quite a lot higher than the other models.

deepSSF models

```
ggplot() +

# dashed lines containing the SSF probabilities (for zooming into in the paper)
geom_hline(yintercept = 0.6e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 1.5e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
```

```

geom_ribbon(data = validation_all_sliding_period %>%
  filter(probability == "habitat" &
    id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
  alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "habitat" &
    !id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
  filter(probability == "habitat" &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "habitat" &
    id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability

```

```

geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
                  values = c("#E25834", "#000000"),
                  labels = c("deepSSF", "deepSSF S2")) +

scale_color_manual(name = "Model",
                  values = c("#E25834", "#000000"),
                  labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))

```



```

# ggsave(paste0("outputs/validation_deepSSF_sliding_", window_width, "days.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

SSF models

```

ggplot() +

```

```

# in sample 50% ribbon
geom_ribbon(data = validation_all_sliding_period %>%
  filter(probability == "habitat",
    id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = average_time,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
  alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "habitat",
    !id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
  filter(probability == "habitat" &
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "habitat",
    id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

```

```
# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
                  values = c("#0096B5", "#26185F"),
                  labels = c("SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
                  values = c("#0096B5", "#26185F"),
                  labels = c("SSF", "SSF 2p")) +

scale_y_continuous("Probability value",
                  position = "right",
                  labels = function(x) format(x, scientific = TRUE)) +

scale_x_datetime("Date") +
coord_cartesian(ylim = c(0.6e-4, 1.5e-4)) +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))
```



```
# ggsave(paste0("outputs/validation_SSF_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

There isn't a clear seasonal trend with the SSF predictions, but in general the models do

better than the null model, although the out-of-sample predictions vary around the null model.

Movement probability across the tracking period

All models

```
ggplot() +  
  
  # dashed lines containing the SSF probabilities  
  geom_hline(yintercept = 0.6e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  geom_hline(yintercept = 1.5e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  
  # in sample 50% ribbon  
  geom_ribbon(data = validation_all_sliding_period %>%  
    filter(probability == "move" &  
      id == focal_id),  
    aes(x = average_time,  
        ymin = q25,  
        ymax = q75,  
        fill = model,  
        group = interaction(id, model)),  
    alpha = ribbon_50_alpha) +  
  
  # out-of-sample thin line for each individual  
  geom_line(data = validation_all_sliding_period %>%  
    filter(probability == "move" &  
      !id == focal_id),  
    aes(x = average_time,  
        y = average_prob,  
        colour = model,  
        group = interaction(id, model)),  
    linewidth = secondary_linewidth) +  
  
  # out-of-sample mean line for all individuals  
  geom_line(data = validation_all_sliding_period_OOS %>%  
    filter(probability == "move"),  
    aes(x = average_time,  
        y = average_prob,  
        colour = model),  
    linewidth = OOS_mean_linewidth,  
    linetype = "dashed") +
```

```

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "move" &
    id == focal_id),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
  values = c("#E25834", "#000000", "#0096B5", "#26185F"),
  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
  values = c("#E25834", "#000000", "#0096B5", "#26185F"),
  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +
theme_bw() +
theme(legend.position = "bottom",
  axis.text.x = element_text(angle = 30, hjust = 1))

```




```
# ggsave(paste0("outputs/validation_all_move_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

The movement probabilities for the deepSSF models are much higher than the for the SSF models, which I suspect is mostly due to the mixture of distributions in the deepSSF models. When the buffalo are in a low movement period, there can be very high probability in the few cells close to the buffalo, which is often accurate (when the predicted probability is 0.2 that means all of the probability mass is shared between only 5 cells. I don't think the SSF movement kernel has the same flexibility to capture this.

This also comes out very clearly in the hourly predictions, with much higher predicted probabilities during the low movement periods.

deepSSF models

```
ggplot() +

# dashed lines containing the SSF probabilities
geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 9e-3, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_sliding_period %>%
           filter(probability == "move" &
```

```

        id == focal_id &
        grepl("deepSSF", model)),
aes(x = average_time,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_sliding_period %>%
    filter(probability == "move" &
        !id == focal_id &
        grepl("deepSSF", model)),
aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
    filter(probability == "move" &
        grepl("deepSSF", model)),
aes(x = average_time,
    y = average_prob,
    colour = model),
linewidth = OOS_mean_linewidth,
linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
    filter(probability == "move" &
        id == focal_id &
        grepl("deepSSF", model)),
aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

```

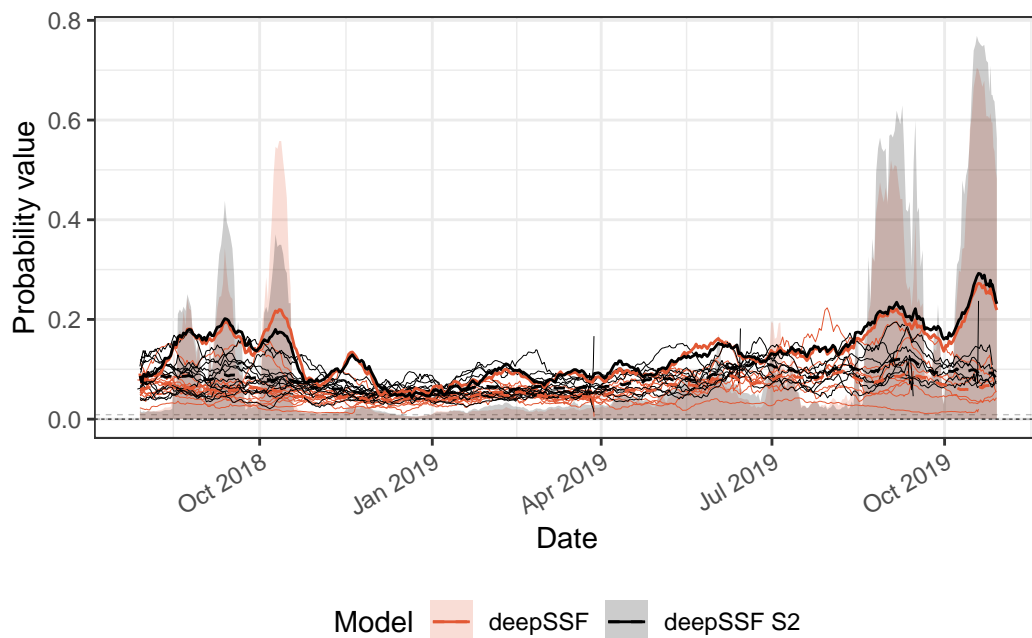
```

scale_fill_manual(name = "Model",
                  values = c("#E25834", "#000000"),
                  labels = c("deepSSF", "deepSSF S2")) +

scale_color_manual(name = "Model",
                   values = c("#E25834", "#000000"),
                   labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))

```



```

# ggsave(paste0("outputs/validation_deepSSF_move_sliding_", window_width, "days.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

SSF models

```

ggplot() +

# in sample 50% ribbon
geom_ribbon(data = validation_all_sliding_period %>%

```

```

        filter(probability == "move",
               id == focal_id,
               !grepl("deepSSF", model)),
    aes(x = average_time,
        ymin = q25,
        ymax = q75,
        fill = model,
        group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_sliding_period %>%
          filter(probability == "move",
                 !id == focal_id,
                 !grepl("deepSSF", model)),
          aes(x = average_time,
              y = average_prob,
              colour = model,
              group = interaction(id, model)),
          linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
          filter(probability == "move" &
                 !grepl("deepSSF", model)),
          aes(x = average_time,
              y = average_prob,
              colour = model),
          linewidth = OOS_mean_linewidth,
          linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
          filter(probability == "move",
                 id == focal_id,
                 !grepl("deepSSF", model)),
          aes(x = average_time,
              y = average_prob,
              colour = model,
              group = interaction(id, model)),
          linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

```

```

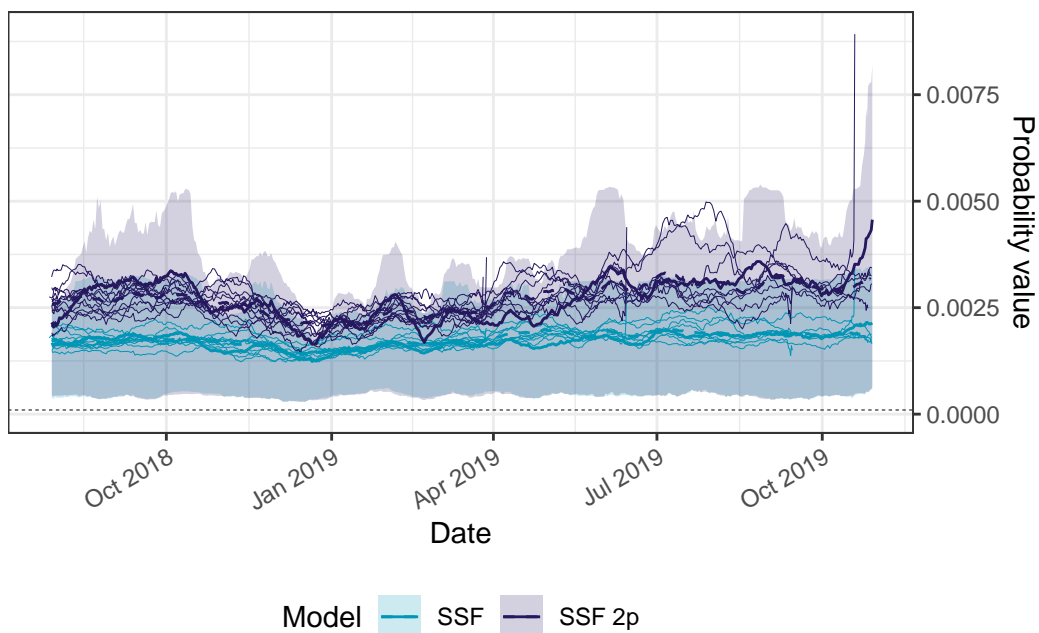
scale_fill_manual(name = "Model",
                  values = c("#0096B5", "#26185F"),
                  labels = c("SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
                  values = c("#0096B5", "#26185F"),
                  labels = c("SSF", "SSF 2p")) +

scale_y_continuous("Probability value",
                  position = "right") +

scale_x_datetime("Date") +
coord_cartesian(ylim = c(0, 9e-3)) +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))

```



```

# ggsave(paste0("outputs/validation_SSF_move_sliding_", window_width, "days.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

The temporally dynamic models have consistently higher probabilities of movement than the static models, which again is likely due to the concentration of the movement kernel during the low movement periods, where the probability values are distributed across much fewer cells.

Next-step probabilities across the tracking period

The next-step probability values are very similar to the movement probabilities due to the concentration of the probability mass in fewer cells that are closer to the buffalo, whereas the habitat selection probabilities are distributed across all of the local layers, so they're not actually that informative beyond the movement probabilities.

All models

```
ggplot() +  
  
# dashed lines containing the SSF probability values  
geom_hline(yintercept = 0.6e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
geom_hline(yintercept = 1.5e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  
# in sample 50% ribbon  
geom_ribbon(data = validation_all_sliding_period %>%  
  filter(probability == "next_step" &  
    id == focal_id),  
  aes(x = average_time,  
    ymin = q25,  
    ymax = q75,  
    fill = model,  
    group = interaction(id, model)),  
  alpha = ribbon_50_alpha) +  
  
# out-of-sample thin line for each individual  
geom_line(data = validation_all_sliding_period %>%  
  filter(probability == "next_step" &  
    !id == focal_id),  
  aes(x = average_time,  
    y = average_prob,  
    colour = model,  
    group = interaction(id, model)),  
  linewidth = secondary_linewidth) +  
  
# out-of-sample mean line for all individuals  
geom_line(data = validation_all_sliding_period_OOS %>%  
  filter(probability == "next_step"),  
  aes(x = average_time,  
    y = average_prob,  
    colour = model),  
  linewidth = OOS_mean_linewidth,
```

```

        linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
          filter(probability == "next_step" &
                 id == focal_id),
          aes(x = average_time,
              y = average_prob,
              colour = model,
              group = interaction(id, model)),
          linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
                  values = c("#E25834", "#000000", "#0096B5", "#26185F"),
                  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
                  values = c("#E25834", "#000000", "#0096B5", "#26185F"),
                  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))

```



```
# ggsave(paste0("outputs/validation_all_next_step_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

deepSSF models

```
ggplot() +

# dashed lines containing the SSF probability values
geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 9e-3, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_sliding_period %>%
  filter(probability == "next_step" &
    id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
  alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
```



```

geom_line(data = validation_all_sliding_period %>%
  filter(probability == "next_step" &
    !id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
  filter(probability == "next_step" &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "next_step" &
    id == focal_id &
    grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
  values = c("#E25834", "#000000"),
  labels = c("deepSSF", "deepSSF S2")) +

scale_color_manual(name = "Model",
  values = c("#E25834", "#000000"),
  labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_datetime("Date") +

```

```
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))
```



```
# ggsave(paste0("outputs/validation_deepSSF_next_step_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

SSF models

```
ggplot() +

# in sample 50% ribbon
geom_ribbon(data = validation_all_sliding_period %>%
            filter(probability == "next_step",
                   id == focal_id,
                   !grepl("deepSSF", model)),
            aes(x = average_time,
                ymin = q25,
                ymax = q75,
                fill = model,
                group = interaction(id, model)),
            alpha = ribbon_50_alpha) +
```

```

# out-of-sample thin line for each individual
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "next_step",
    !id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line for all individuals
geom_line(data = validation_all_sliding_period_OOS %>%
  filter(probability == "next_step" &
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_sliding_period %>%
  filter(probability == "next_step",
    id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = average_time,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.15) +

scale_fill_manual(name = "Model",
  values = c("#0096B5", "#26185F"),
  labels = c("SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
  values = c("#0096B5", "#26185F"),
  labels = c("SSF", "SSF 2p")) +

scale_y_continuous("Probability value",

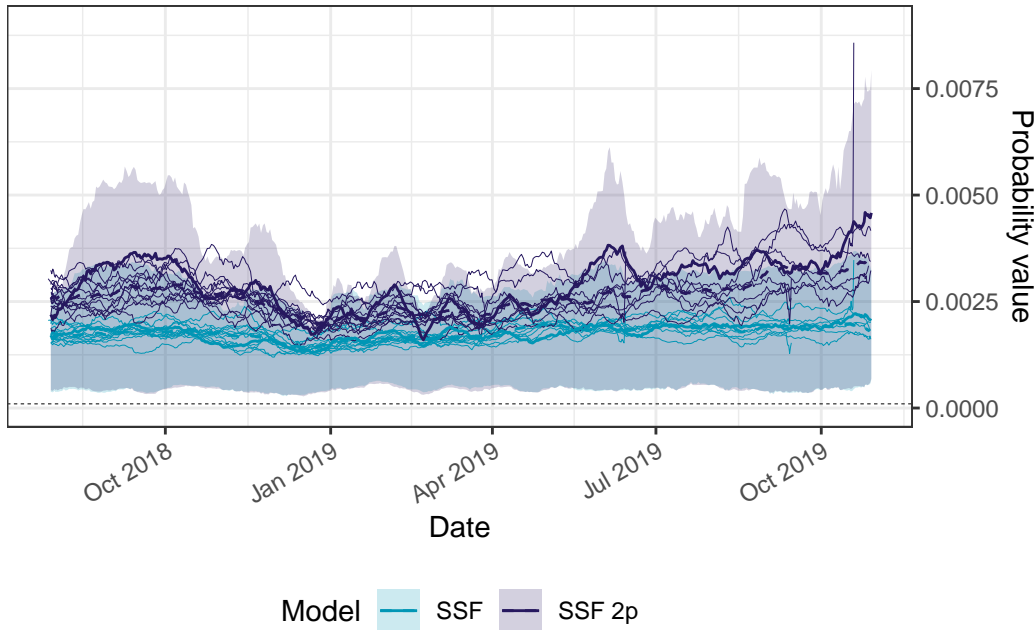
```

```

      position = "right") +

scale_x_datetime("Date") +
coord_cartesian(ylim = c(0, 9e-3)) +
theme_bw() +
theme(legend.position = "bottom",
      axis.text.x = element_text(angle = 30, hjust = 1))

```



```

# ggsave(paste0("outputs/validation_SSF_next_step_sliding_", window_width, "days.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)

```

Hourly probabilities

We can also calculate the habitat selection, movement and next-step probabilities for each hour of the day, indicating when the models are accurate (or not) at different times of the day.

For this we don't need a sliding window, we will just bin by the hour of the day.

Here we just show the hourly probabilities across the whole tracking period, but we could also split this up into seasons and assess how accurate the models are during the wet and dry seasons.

```
# grouping by each individual, model and hour of the day
validation_all_quantiles_hourly <- validation_all_long %>%
  group_by(id, model, probability, hour_t2) %>%
  summarise(average_prob = mean(value, na.rm = T),
            sd_prob = sd(value, na.rm = T),
            q025 = quantile(value, probs = 0.025, na.rm = T),
            q25 = quantile(value, probs = 0.25, na.rm = T),
            q50 = quantile(value, probs = 0.5, na.rm = T),
            q75 = quantile(value, probs = 0.75, na.rm = T),
            q975 = quantile(value, probs = 0.975, na.rm = T)
  )
```

`summarise()` has grouped output by 'id', 'model', 'probability'. You can override using the `.groups` argument.

```
head(validation_all_quantiles_hourly)
```

```
# A tibble: 6 x 11
# Groups:   id, model, probability [1]
   id model probability hour_t2 average_prob sd_prob q025 q25 q50
<dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 2005 deepSSF habitat 0 0.000249 3.21e-4 2.50e-5 1.03e-4 1.34e-4
2 2005 deepSSF habitat 1 0.000233 2.95e-4 2.77e-5 9.78e-5 1.31e-4
3 2005 deepSSF habitat 2 0.000239 3.05e-4 2.69e-5 9.89e-5 1.32e-4
4 2005 deepSSF habitat 3 0.000232 3.09e-4 1.89e-5 1.04e-4 1.31e-4
5 2005 deepSSF habitat 4 0.000241 3.21e-4 2.86e-5 1.06e-4 1.30e-4
6 2005 deepSSF habitat 5 0.000229 3.20e-4 2.96e-5 1.06e-4 1.24e-4
# i 2 more variables: q75 <dbl>, q975 <dbl>
```

```
# out-of-sample ids
# grouping by each model and hour of the day (combining all OOS individuals to get the average)
validation_all_quantiles_hourly_OOS <- validation_all_long %>%
  filter(!id == focal_id) %>%
  group_by(model, probability, hour_t2) %>%
  summarise(average_prob = mean(value, na.rm = T),
            sd_prob = sd(value, na.rm = T),
            q025 = quantile(value, probs = 0.025, na.rm = T),
            q25 = quantile(value, probs = 0.25, na.rm = T),
            q50 = quantile(value, probs = 0.5, na.rm = T),
            q75 = quantile(value, probs = 0.75, na.rm = T),
            q975 = quantile(value, probs = 0.975, na.rm = T)
  )
```

`summarise()` has grouped output by 'model', 'probability'. You can override using the `.groups` argument.

```
head(validation_all_quantiles_hourly_OOS)
```

```
# A tibble: 6 x 10
# Groups:   model, probability [1]
  model probability hour_t2 average_prob sd_prob q025 q25 q50 q75
  <chr> <chr>         <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl>
1 deep~ habitat      0  0.0000941 1.08e-4 4.44e-6 3.73e-5 7.43e-5 1.25e-4
2 deep~ habitat      1  0.0000952 1.22e-4 4.44e-6 3.61e-5 7.41e-5 1.25e-4
3 deep~ habitat      2  0.0000968 1.19e-4 4.54e-6 3.64e-5 7.53e-5 1.26e-4
4 deep~ habitat      3  0.0000989 1.18e-4 4.64e-6 3.75e-5 7.95e-5 1.30e-4
5 deep~ habitat      4  0.000101  1.05e-4 5.22e-6 4.17e-5 8.81e-5 1.35e-4
6 deep~ habitat      5  0.000102  9.73e-5 6.81e-6 4.57e-5 9.63e-5 1.34e-4
# i 1 more variable: q975 <dbl>
```

Habitat selection across the day

All models

```
ggplot() +

# dashed lines containing the SSF probabilities
geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 3e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    id == focal_id),
  aes(x = hour_t2,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
  alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    !id == focal_id),
```

```

    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = 0.075) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
    filter(probability == "habitat"),
    aes(x = hour_t2,
        y = average_prob,
        colour = model),
    linewidth = OOS_mean_linewidth,
    linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "habitat",
        id == focal_id),
    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
    values = c("#AF3602", "#000000", "#0096B5", "#000000"),
    labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
    values = c("#AF3602", "#000000", "#0096B5", "#000000"),
    labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", breaks = seq(0,24,6)) +
theme_bw() +
theme(legend.position = "bottom")

```



```
# ggsave(paste0("outputs/validation_all_hourly.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

The deepSSF models both predict well in the evening (at least in-sample), but only the deepSSF S2 predicts well during the middle of the day, again suggesting that there is information in the Sentinel-2 layers that isn't present in the derived covariates.

The out-of-sample predictions are also much higher for the deepSSF S2 model, suggesting that it is better at generalising to new data.

deepSSF models

```
ggplot() +

# dashed lines containing the SSF probabilities
geom_hline(yintercept = 0.6e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 1.5e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
            filter(probability == "habitat",
                   id == focal_id,
                   grepl("deepSSF", model)),
            aes(x = hour_t2,
                ymin = q25,
```



```

        ymax = q75,
        fill = model,
        group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    !id == focal_id,
    grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
  filter(probability == "habitat" &
    grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    id == focal_id,
    grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
  values = c("#E25834", "#000000"),
  labels = c("deepSSF", "deepSSF S2")) +

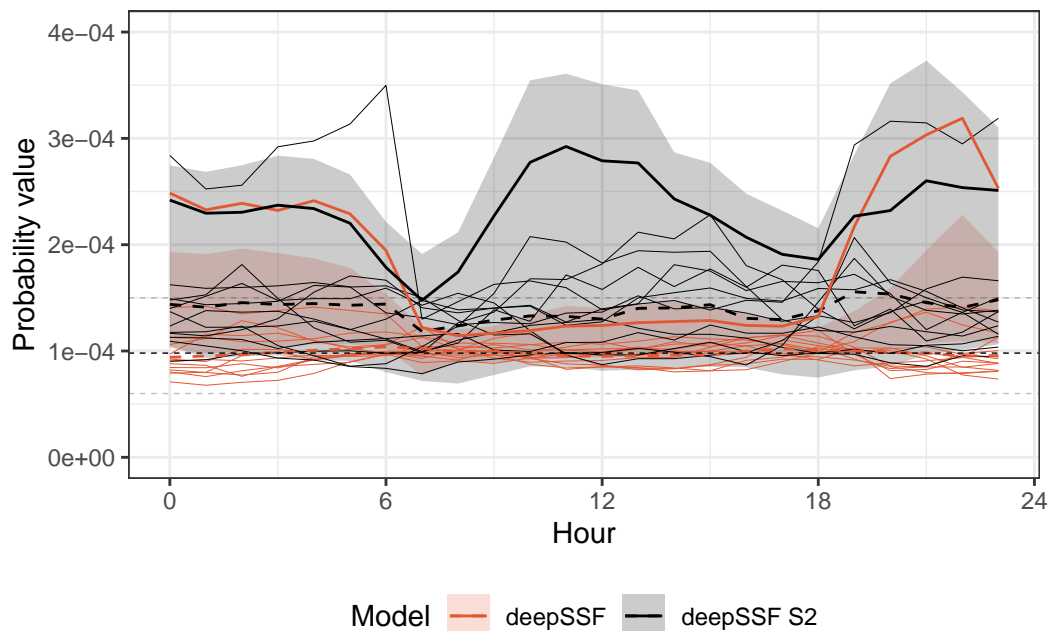
```

```

scale_color_manual(name = "Model",
                   values = c("#E25834", "#000000"),
                   labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", seq(0,24,6)) +
coord_cartesian(ylim = c(0, 4e-4)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_deepSSF_hourly.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

SSF models

```

ggplot() +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
           filter(probability == "habitat",
                  id == focal_id,
                  !grepl("deepSSF", model)),
           aes(x = hour_t2,

```

```

        ymin = q25,
        ymax = q75,
        fill = model,
        group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    !id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = secondary_linewidth) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
  filter(probability == "habitat" &
    !grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "habitat",
    id == focal_id,
    !grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = primary_linewidth) +

# dashed line for the null probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
  values = c("#0096B5", "#26185F"),
  labels = c("SSF", "SSF 2p")) +

```

```

scale_color_manual(name = "Model",
                   values = c("#0096B5", "#26185F"),
                   labels = c("SSF", "SSF 2p")) +

scale_y_continuous("Probability value",
                   position = "right",
                   labels = function(x) format(x, scientific = TRUE)) +

scale_x_continuous("Hour", seq(0,24,6)) +
coord_cartesian(ylim = c(0.6e-4, 1.5e-4)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_SSF_hourly.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)

```

The SSF model with temporal dynamics had higher prediction accuracy than the SSF model without temporal dynamics overall (in- and out-of-sample), but it was more variable throughout the day, and was quite poor between midnight and about 5am.

Movement probability across the day

All models

```
ggplot() +  
  
  # dashed lines containing the SSF probabilities  
  geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  geom_hline(yintercept = 3e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  
  # in sample 50% ribbon  
  geom_ribbon(data = validation_all_quantiles_hourly %>%  
    filter(probability == "move",  
           id == focal_id),  
    aes(x = hour_t2,  
        ymin = q25,  
        ymax = q75,  
        fill = model,  
        group = interaction(id, model)),  
    alpha = ribbon_50_alpha) +  
  
  # out-of-sample thin line for each individual  
  geom_line(data = validation_all_quantiles_hourly %>%  
    filter(probability == "move",  
           !id == focal_id),  
    aes(x = hour_t2,  
        y = average_prob,  
        colour = model,  
        group = interaction(id, model)),  
    linewidth = 0.075) +  
  
  # out-of-sample mean line  
  geom_line(data = validation_all_quantiles_hourly_OOS %>%  
    filter(probability == "move"),  
    aes(x = hour_t2,  
        y = average_prob,  
        colour = model),  
    linewidth = OOS_mean_linewidth,  
    linetype = "dashed") +  
  
  # in sample mean line  
  geom_line(data = validation_all_quantiles_hourly %>%  
    filter(probability == "move",
```

```

      id == focal_id),
    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = 0.35) +

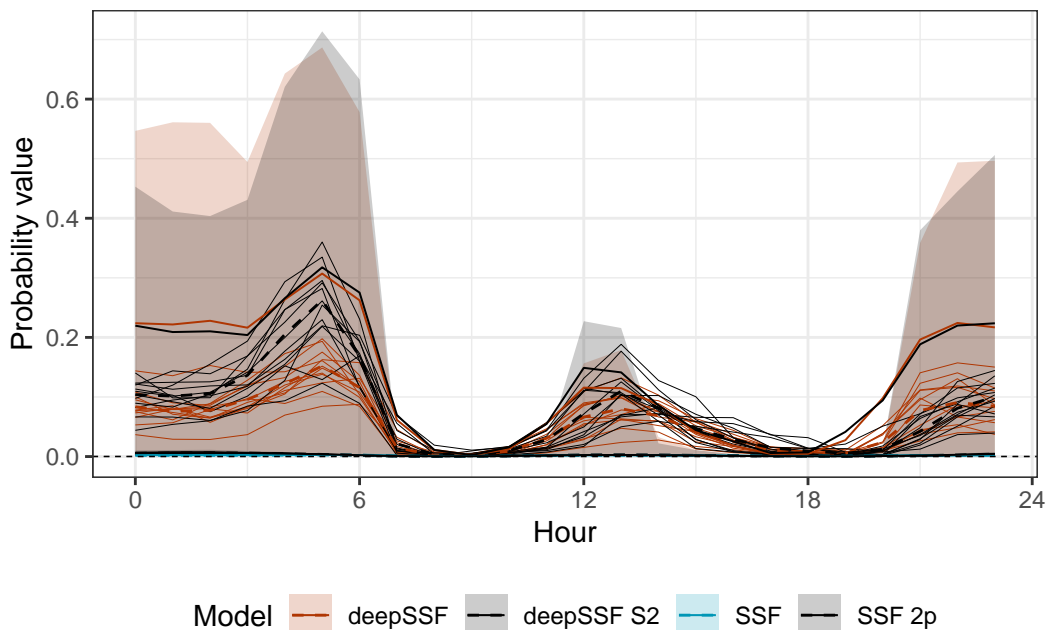
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
                  values = c("#AF3602", "#000000", "#0096B5", "#000000"),
                  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
                   values = c("#AF3602", "#000000", "#0096B5", "#000000"),
                   labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", breaks = seq(0,24,6)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_all_move_hourly.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

The movement probabilities are much lower during the high movement periods. This is be-

cause there are many more cells that the buffalo are likely to move to, and so the probability of moving to any one cell is lower, and the model must spread the prediction probability across many more cells.

The SSF probabilities are also much much lower than the deepSSF probabilities, as explained for the movement probabilities across the tracking period.

deepSSF models

```
ggplot() +  
  
  # dashed lines containing the SSF probabilities  
  geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  geom_hline(yintercept = 1.25e-2, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +  
  
  # in sample 50% ribbon  
  geom_ribbon(data = validation_all_quantiles_hourly %>%  
    filter(probability == "move",  
           id == focal_id,  
           grepl("deepSSF", model)),  
    aes(x = hour_t2,  
        ymin = q25,  
        ymax = q75,  
        fill = model,  
        group = interaction(id, model)),  
    alpha = ribbon_50_alpha) +  
  
  # out-of-sample thin line for each individual  
  geom_line(data = validation_all_quantiles_hourly %>%  
    filter(probability == "move",  
           !id == focal_id,  
           grepl("deepSSF", model)),  
    aes(x = hour_t2,  
        y = average_prob,  
        colour = model,  
        group = interaction(id, model)),  
    linewidth = secondary_linewidth) +  
  
  # out-of-sample mean line  
  geom_line(data = validation_all_quantiles_hourly_OOS %>%  
    filter(probability == "move" &  
           grepl("deepSSF", model)),  
    aes(x = hour_t2,  
        y = average_prob,
```

```

        colour = model),
        linewidth = OOS_mean_linewidth,
        linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
          filter(probability == "move",
                 id == focal_id,
                 grepl("deepSSF", model)),
          aes(x = hour_t2,
              y = average_prob,
              colour = model,
              group = interaction(id, model)),
          linewidth = primary_linewidth) +

# dashed line for uniform probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
                  values = c("#E25834", "#000000"),
                  labels = c("deepSSF", "deepSSF S2")) +

scale_color_manual(name = "Model",
                   values = c("#E25834", "#000000"),
                   labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", seq(0,24,6)) +
theme_bw() +
theme(legend.position = "bottom")

```




```
# ggsave(paste0("outputs/validation_deepSSF_move_hourly.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

SSF models

```
ggplot() +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
            filter(probability == "move",
                   id == focal_id,
                   !grepl("deepSSF", model)),
            aes(x = hour_t2,
                ymin = q25,
                ymax = q75,
                fill = model,
                group = interaction(id, model)),
            alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
           filter(probability == "move",
                  !id == focal_id,
                  !grepl("deepSSF", model)),
```

```

    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = secondary_linewidth) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
    filter(probability == "move" &
        !grepl("deepSSF", model)),
    aes(x = hour_t2,
        y = average_prob,
        colour = model),
    linewidth = OOS_mean_linewidth,
    linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "move",
        id == focal_id,
        !grepl("deepSSF", model)),
    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = primary_linewidth) +

# dashed line for uniform probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
    values = c("#0096B5", "#26185F"),
    labels = c("SSF", "SSF 2p")) +

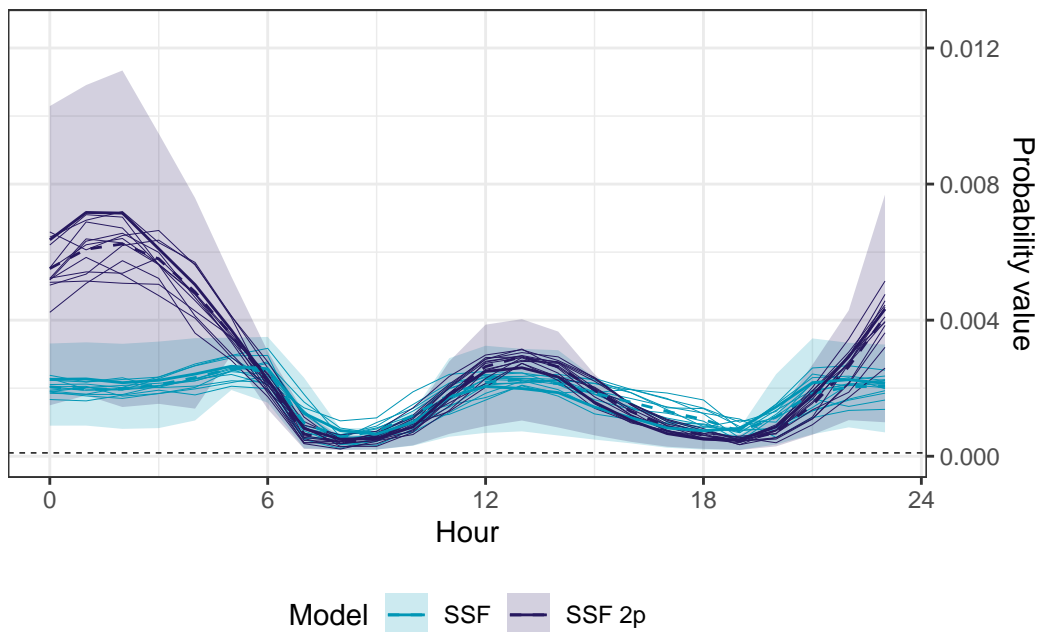
scale_color_manual(name = "Model",
    values = c("#0096B5", "#26185F"),
    labels = c("SSF", "SSF 2p")) +

    scale_y_continuous("Probability value",
        position = "right") +

scale_x_continuous("Hour", seq(0,24,6)) +
coord_cartesian(ylim = c(0, 1.25e-2)) +
theme_bw() +

```

```
theme(legend.position = "bottom")
```



```
# ggsave(paste0("outputs/validation_SSF_move_hourly.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)
```

The SSF models perform similarly, except at night, where the temporally dynamic performs much better in- and out-of-sample.

Next-step probability across the day

All models

```
ggplot() +

# dashed lines containing the SSF probabilities
geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 3e-4, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
            filter(probability == "next_step",
                   id == focal_id),
            aes(x = hour_t2,
```

```

        ymin = q25,
        ymax = q75,
        fill = model,
        group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "next_step",
    !id == focal_id),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = 0.075) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
  filter(probability == "move" &
    !grepl("deepSSF", model)),
  aes(x = hour_t2,
    y = average_prob,
    colour = model),
  linewidth = OOS_mean_linewidth,
  linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
  filter(probability == "next_step",
    id == focal_id),
  aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
  linewidth = 0.35) +

# dashed line for uniform probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",
  values = c("#AF3602", "#000000", "#0096B5", "#000000"),
  labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_color_manual(name = "Model",

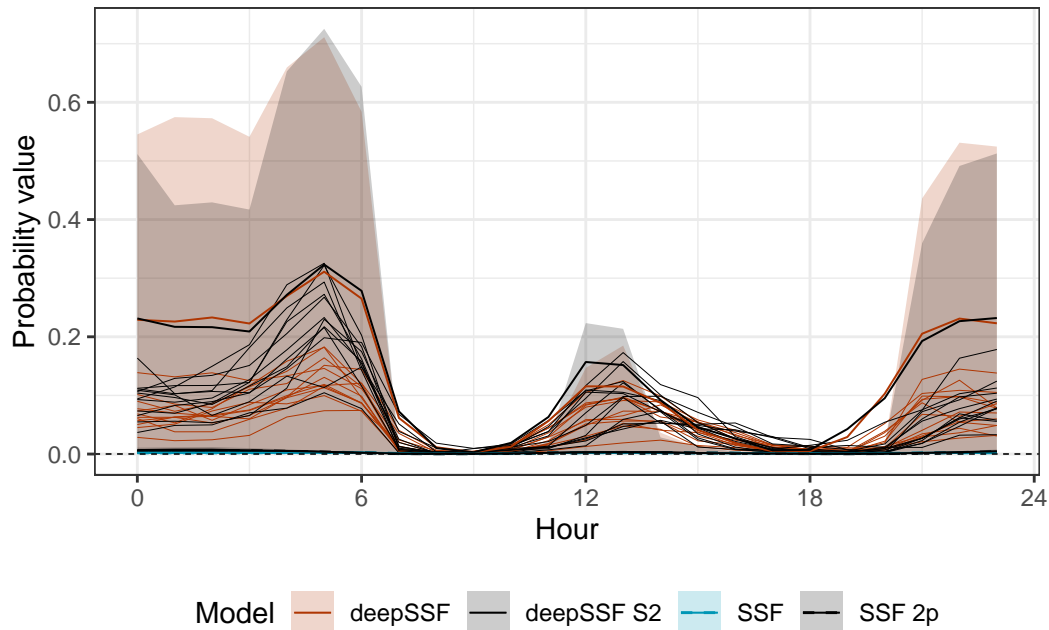
```

```

    values = c("#AF3602", "#000000", "#0096B5", "#000000"),
    labels = c("deepSSF", "deepSSF S2", "SSF", "SSF 2p")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", breaks = seq(0,24,6)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_all_next_step_hourly.png"),
#        width = 80, height = 80, units = "mm", dpi = 600)

```

deepSSF models

```

ggplot() +

# dashed lines containing the SSF probabilities
geom_hline(yintercept = 0, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +
geom_hline(yintercept = 1.25e-2, alpha = 0.25, linetype = "dashed", linewidth = 0.25) +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
           filter(probability == "next_step",
                  id == focal_id,

```

```

        grepl("deepSSF", model)),
    aes(x = hour_t2,
        ymin = q25,
        ymax = q75,
        fill = model,
        group = interaction(id, model)),
    alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "next_step",
        !id == focal_id,
        grepl("deepSSF", model)),
    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = secondary_linewidth) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
    filter(probability == "next_step" &
        grepl("deepSSF", model)),
    aes(x = hour_t2,
        y = average_prob,
        colour = model),
    linewidth = OOS_mean_linewidth,
    linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "next_step",
        id == focal_id,
        grepl("deepSSF", model)),
    aes(x = hour_t2,
        y = average_prob,
        colour = model,
        group = interaction(id, model)),
    linewidth = primary_linewidth) +

# dashed line for uniform probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",

```

```

      values = c("#E25834", "#000000"),
      labels = c("deepSSF", "deepSSF S2")) +

scale_color_manual(name = "Model",
      values = c("#E25834", "#000000"),
      labels = c("deepSSF", "deepSSF S2")) +

scale_y_continuous("Probability value") +
scale_x_continuous("Hour", seq(0,24,6)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_deepSSF_next_step_hourly.png"),
#       width = 80, height = 80, units = "mm", dpi = 600)

```

SSF models

```

ggplot() +

# in sample 50% ribbon
geom_ribbon(data = validation_all_quantiles_hourly %>%
  filter(probability == "next_step",
    id == focal_id,

```

```

        !grepl("deepSSF", model)),
aes(x = hour_t2,
    ymin = q25,
    ymax = q75,
    fill = model,
    group = interaction(id, model)),
alpha = ribbon_50_alpha) +

# out-of-sample thin line for each individual
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "next_step",
        !id == focal_id,
        !grepl("deepSSF", model)),
aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
linewidth = secondary_linewidth) +

# out-of-sample mean line
geom_line(data = validation_all_quantiles_hourly_OOS %>%
    filter(probability == "next_step" &
        !grepl("deepSSF", model)),
aes(x = hour_t2,
    y = average_prob,
    colour = model),
linewidth = OOS_mean_linewidth,
linetype = "dashed") +

# in sample mean line
geom_line(data = validation_all_quantiles_hourly %>%
    filter(probability == "next_step",
        id == focal_id,
        !grepl("deepSSF", model)),
aes(x = hour_t2,
    y = average_prob,
    colour = model,
    group = interaction(id, model)),
linewidth = primary_linewidth) +

# dashed line for uniform probability
geom_hline(yintercept = uniform_prob, linetype = "dashed", linewidth = 0.25) +

scale_fill_manual(name = "Model",

```



```

      values = c("#0096B5", "#26185F"),
      labels = c("SSF", "SSF 2p")) +

scale_color_manual(name = "Model",
  values = c("#0096B5", "#26185F"),
  labels = c("SSF", "SSF 2p")) +

  scale_y_continuous("Probability value",
    position = "right") +

scale_x_continuous("Hour", seq(0,24,6)) +
coord_cartesian(ylim = c(0, 1.25e-2)) +
theme_bw() +
theme(legend.position = "bottom")

```



```

# ggsave(paste0("outputs/validation_SSF_next_step_hourly.png"),
#         width = 80, height = 80, units = "mm", dpi = 600)

```