Supplementary Material for:

Predicting Missing Values in Spatio-Temporal Remote Sensing Data

Florian Gerber, Rogier de Jong, Michael E. Schaepman,
Gabriela Schaepman-Strub, Reinhard Furrer.

This document contains additional information about the R package *gapfill* (Section S1), tables (Section S2), figures (Section S3), code listings (Section S4), and an additional comparison study using a MODIS EVI data set from the Amazon region (Section S5).

S1 Software

To following description gives insights in the implementation of the proposed prediction method in the R package *gapfill* (Gerber, 2016) and is primarily interesting to users of the package. We chose the programming language R (R Core Team, 2015) to implement the method because it is a popular open-source software with convenient code sharing infrastructures such as the Comprehensive R Archive Network (CRAN), http://cran.r-project.org. Moreover, R provides a wide range of add-on packages for processing and analyzing remote sensing data. A non-exhaustive overview is given in the CRAN task views "handling and analyzing spatio-temporal data" (Pebesma, 2015), "analysis of spatial data" (Bivand, 2015) and "time series analysis" (Hyndman, 2015).

S1.1 Worked Through Example with a NDVI Data Set

To illustrate the R package *gapfill*, we go through an example use-case. The package is available on CRAN and can be installed and loaded into R with

```
R> install.packages("gapfill")
R> library("gapfill", quiet = TRUE)
```

We consider the MODIS NDVI data set shown in Fig 1-b. The data set is contained in *gapfill* and stored as a four dimensional array of numeric values. Missing values are encoded with `NA`.

```
R> data(ndvi)
R> str(ndvi)
 num [1:21, 1:21, 1:4, 1:4] 0.522 0.526 0.459 0.492 0.476 ...
```

```
 - attr(*, "dimnames")=List of 4
  ..$ : chr [1:21] "-153.032" "-153.012" "-152.992" "-152.973" ...
  ..$ : chr [1:21] "69.1" "69.12" "69.14" "69.16" ...
  ..$ : chr [1:4] "145" "161" "177" "193"
  ..$ : chr [1:4] "2004" "2005" "2006" "2007"
> mean(is.na(ndvi))
[1] 0.2271825
```

The data set can be visualized with the function `Image()`. The following call returns a plot similar to Fig. 1.b.

```
R> Image(ndvi)
```

The main function of the package is `Gapfill()`. By default, it predicts the missing values in the input data as described in Section 2.

```
> out <- Gapfill(data = ndvi)
data has 7056 values: 5453 (77.3%) observed,
                      1603 (22.7%) missing,
                      1603 (22.7%) to predict.
started at 2016-02-20 12:33:08.
elapsed time is 0.248 mins (0.0093 secs per NA).
```

Besides the printed summary output, the function returns a list of four objects.

```
R> names(out)
[1] "fill" "mps" "time" "call"
```

The first object `fill` contains the input data set, where the missing values therein are replaced by the predicted values. `fill` has the same array structure as the input data set. In the case, where prediction errors and/or diagnostic information are returned too, `fill` has an additional dimension to capture those values. The other returned objects contain the indices of the predicted values (`mps`), time measurements of the predictions (`time`), and the function call (`call`).
The following call additionally returns the 90% prediction interval (Section 2.3).

```
> out <- Gapfill(data = ndvi, nPredict = 3, predictionInterval = TRUE)
```

In that case the returned `out$fill` object additionally contains the lower and the upper bound of the prediction intervals. `Gapfill()` has the arguments listed in Table S4 allowing the user to tune the method to specific settings. More information is also given in the package manual (Gerber, 2016).

S1.2 Software Design and Implementation

Modular Frame-Work The function `Gapfill()` provides the frame-work of the prediction method. Its main task is to apply the prediction method to all (or some) missing values in `data`. Depending on the argument `dopar`

this is either done with a conventional loop or with a parallelized version using the R package *foreach* (Analytics and Weston, 2015). For each missing value `Gapfill()` cycles though the calls of `fnSubset()` and `fnPredict()` until one of the following stopping criteria is met: (1) `fnPredict()` returns a non-`NA` value, (2) the maximum number of tries specified with the argument `iMax` is reached, (3) `fnSubset()` returns the same subset two times in a row. This procedure is schematized in Figure S15. Note that the possibility to return to the subset step after an unsuccessful prediction is used to adapt the subsets for each missing value. By default, the subset and predict functions provided by the package are used. However, users can pass alternative functions for one or both tasks through the arguments `fnSubset` and `fnPredict`. Hence, the algorithm can be customized with little effort; see Section S1.3 for an example.

The role of the `fnSubset()` function is to provide a search strategy for suitable subsets. The search starts with a small initial subset containing enough information for the prediction when many values therein are observed. If the `fnPredict()` function returns `NA` indicating that the subset was not suitable, `i` is increased and another (larger) subset is tried. The default implementation starts with a four dimensional array around the missing values to predict. The extent of that subset is $\lambda_x = \lambda_y = 5$ pixels in each spatial direction from the missing value, $\lambda_s = 1$ step in both direction of the seasonal index and $\lambda_a = 5$ years in both direction of the year index; see Section 2 and the default value for the argument `initialSize` of `Subset()`. The search strategy consists of enlarging the spatial extent of the subset with increasing `i`. This becomes visible from the definition of `Subset()`.

```
R> Subset <- function (data, mp, i, initialSize = c(5, 5, 1, 5))
+     ArrayAround(data = data, mp = mp, size = initialSize + c(i, i, 0, 0))
```

The function consists of a call to `ArrayAround()`, which returns a four dimensional "box" neighborhood of size `size` around the value at position `mp`.

The `Predict()` function has the arguments `a` (the subset returned by `fnSubset()`), `i` (the number of unsuccessful tries), and the tuning parameters `nTargetImage` (θ_1), `nImages` (θ_2), and `nQuant` (ϑ). The function first checks, if `a` contains enough information for the prediction by verifying the criteria (C1) and (C2) from Section 2. In the case where too little information is contained in `a`, `Predict()` returns `NA` and the process shown in Figure S15 continues. Otherwise, the function scores the images of `a` as described in Algorithm 1 and ranks them accordingly. The scoring is implemented in the function `Score()` and involves several nested loops implemented in C++ (Eddelbuettel et al., 2015; Eddelbuettel, 2013). Then the quantile of the missing value is estimated with the function `EstimateQuantile()`, see Algorithm 2. The returned prediction is obtained with a quantile regression, which is fitted using the R package *quantreg* (Koenker, 2015).

S1.3 Customize Parts of the Prediction Procedure

To facilitate the implementation of alternative methods, `Gapfill()` provides a high degree of flexibility allowing the user to customize the subset and/or the prediction parts of the method. More precisely, user-defined `Subset()` and `Predict()` functions can be passed to `Gapfill()` via the arguments `fnSubset` and `fnPredict`. For illustration, we give an example of a modified prediction method: Assume that the default subset strategy should be changed so that it includes all images having the same seasonal index s as the missing value to predict. Since the default

`Subset()` function has the argument `initialSize` controlling that feature, this behavior is achieved by specifying the argument `initialSize = c(5, 5, 0, Inf)` in the call to `Gapfill()`. Furthermore, assume that the alternative prediction function is defined as the mean of all non-`NA` values in the subset. A function with that behavior is defined as follows.

```
R> PredictMean <- function (a, i) mean(a, na.rm = TRUE)
```

This prediction method can be applied to the previously considered `ndvi` data with the following call.

```
R> out2 <- Gapfill(data = ndvi, fnPredict = PredictMean, verbose = FALSE,
                   initialSize = c(5, 5, 0, Inf))
```

More example and explanations are given in the "Extend" help page of the package manual (Gerber, 2016).

References

Analytics, R. and Weston, S. (2015). *foreach: Foreach looping construct for R*. R package version 1.4.2. S3

Bivand, R. (2015). CRAN task view: Analysis of spatial data. S1

Eddelbuettel, D. (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York. ISBN 978-1-4614-6867-7. S3

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Bates, D., and Chambers, J. (2015). *Rcpp: Seamless R and C++ Integration*. R package version 0.12.1. S3

Gerber, F. (2016). *gapfill: Fill Missing Values in Satellite Data*. R package version 0.9.1. S1, S2, S4

Hyndman, R. J. (2015). CRAN task view: Time series analysis. S1

Koenker, R. (2015). *quantreg: Quantile Regression*. R package version 5.05. S3

Pebesma, E. (2015). CRAN task view: Handling and analyzing spatio-temporal data. S1

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. S1

S2 Tables

Table S1: The predicted values of the four test scenarios obtained with *gapfill*, gapfill-MAP, and TIMESAT are summarized in terms of the number and percentage of predicted values and the MAPE $\cdot 10^3$. To get comparable results, the MAPEs of *gapfill* are also given for the subset with available predictions from gapfill-MAP (MAPE_{MAP}) and TIMESAT (MAPE_{T}).

	gapfill				gapfill-MAP		TIMESAT	
	# predicted	MAPE	MAPE_{MAP}	MAPE_{T}	# predicted	MAPE	# predicted	MAPE
20%	92'822 (100 %)	26.98	27.00	26.40	90'307 (97%)	29.84	59'948 (65%)	58.22
30%	147'827 (100 %)	27.44	27.36	24.25	146'686 (99%)	30.31	42'892 (29%)	49.37
40%	192'456 (100 %)	26.80	26.20	23.23	169'998 (88%)	28.41	31'279 (16%)	49.12
50%	240'326 (100 %)	35.43	27.78	24.90	134'540 (56%)	30.23	14'127 (6%)	59.81

Table S2: The predicted values of four additional test scenarios having randomly removed values are summarized. The predictions from *gapfill*, gapfill-MAP, and TIMESAT are summarized in terms of the number and percentage of predicted values and the RMSPE $\cdot 10^3$. To get comparable results, the RMSPEs of *gapfill* are also given for the subset with available predictions from gapfill-MAP ($\text{RMSPE}_{\text{MAP}}$) and TIMESAT ($\text{RMSPE}_{\text{T}}$).

	gapfill				gapfill-MAP		TIMESAT	
	# predicted	RMSPE	$\text{RMSPE}_{\text{MAP}}$	RMSPE_{T}	# predicted	RMSPE	# predicted	RMSPE
20%	96'000 (100 %)	35.70	35.70	32.65	95'775 (100%)	36.76	62'728 (65%)	68.80
30%	144'000 (100 %)	35.62	35.55	31.29	143'682 (100%)	36.91	47'656 (33%)	72.81
40%	192'000 (100 %)	36.12	36.11	32.21	191'624 (100%)	37.92	20'506 (11%)	77.07
50%	240'000 (100 %)	36.71	36.64	32.00	239'645 (100%)	39.03	4'210 (2%)	79.91

Table S3: The predicted values of four additional test scenarios having randomly removed values are summarized. The predictions obtained with *gapfill*, gapfill-MAP, and TIMESAT are summarized in terms of the number and percentage of predicted values and the MAPE $\cdot 10^3$. To get comparable results, the MAPEs of *gapfill* are also given for the subset with available predicted values from gapfill-MAP (MAPE_{MAP}) and TIMESAT (MAPE_{T}).

	gapfill				gapfill-MAP		TIMESAT	
	# predicted	MAPE	MAPE_{MAP}	MAPE_{T}	# predicted	MAPE	# predicted	MAPE
20%	96'000 (100 %)	22.37	22.37	21.12	95'775 (100%)	23.68	62'728 (65%)	46.69
30%	144'000 (100 %)	22.33	22.32	20.88	143'682 (100%)	23.96	47'656 (33%)	49.03
40%	192'000 (100 %)	22.50	22.49	21.19	191'624 (100%)	24.58	20'506 (11%)	51.62
50%	240'000 (100 %)	22.84	22.83	21.24	239'645 (100%)	25.36	4'210 (2%)	52.99

Table S4: Arguments of the `Gapfill()` function.

Argument	Description
`data`	Numeric array with four dimensions (including `NA`s).
	The input data set containing the missing values to predict.
`fnSubset`	Function that selects a subset of `data` around a missing value.
`fnPredict`	Function to predict a missing value based on the return subset of `fnSubset()`.
`iMax`	The maximum number of iterations until `NA` is returned as predicted value.
`nPredict`	The length of the vector returned from `fnPredict()`.
`subset`	The subset of `data` to be predicted.
`clipRange`	The lower and the upper bound of the data to clip the predictions accordingly.
`dopar`	If `TRUE` the prediction runs in parallel.
`verbose`	If `TRUE`, messages are printed.
`...`	Additional arguments passed to `fnSubset()` and `fnPredict()`.

S3 Figures

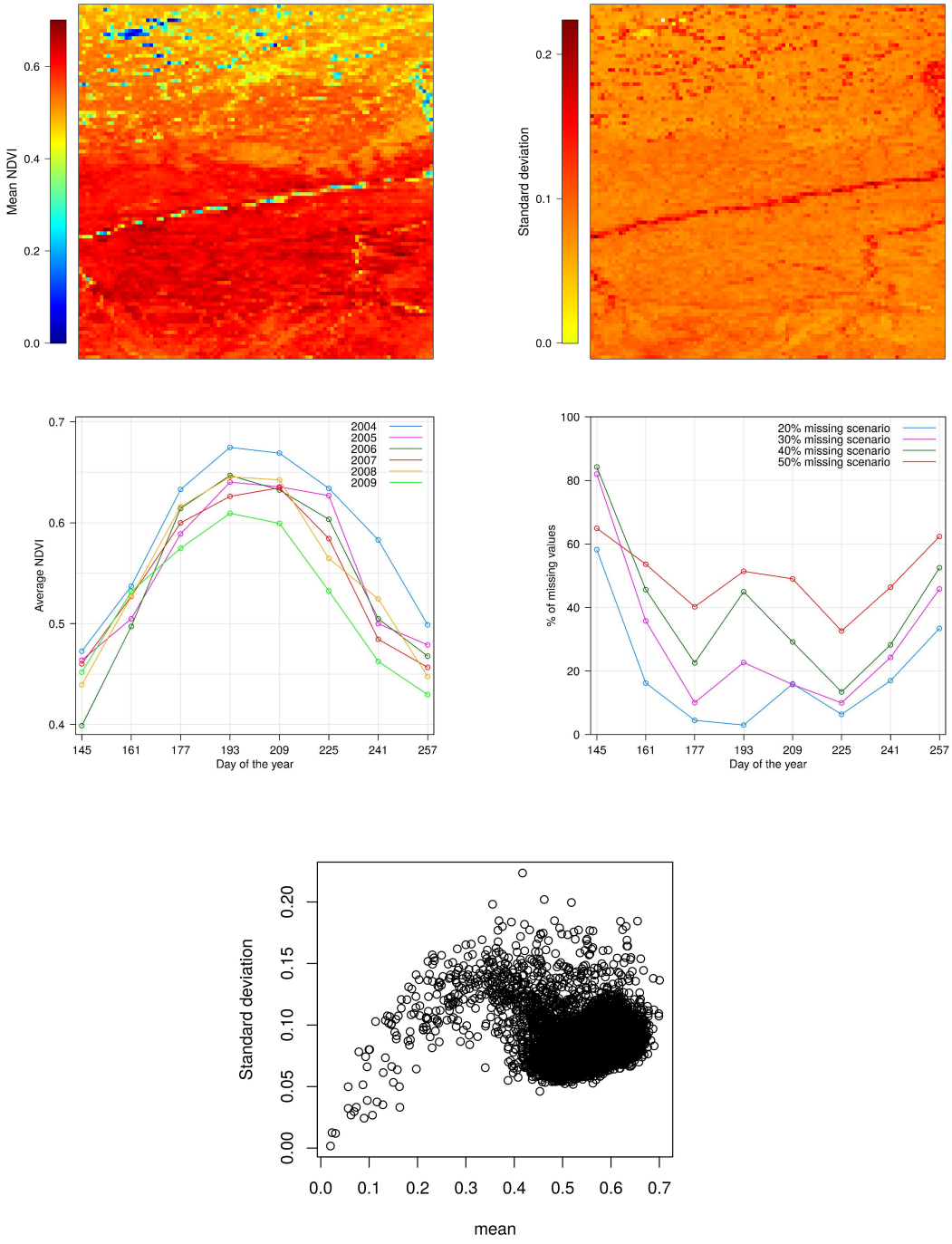

Figure S1: Top: images of the mean and the standard deviation of the considered NDVI test data set. The spatial extent is $100 \cdot 100$ values. Middle: average values per time point (left) and temporal distribution of the missing values (right). Bottom: mean and standard deviation of each location over time. Locations with mean values below 0.4 have an increased standard deviation.

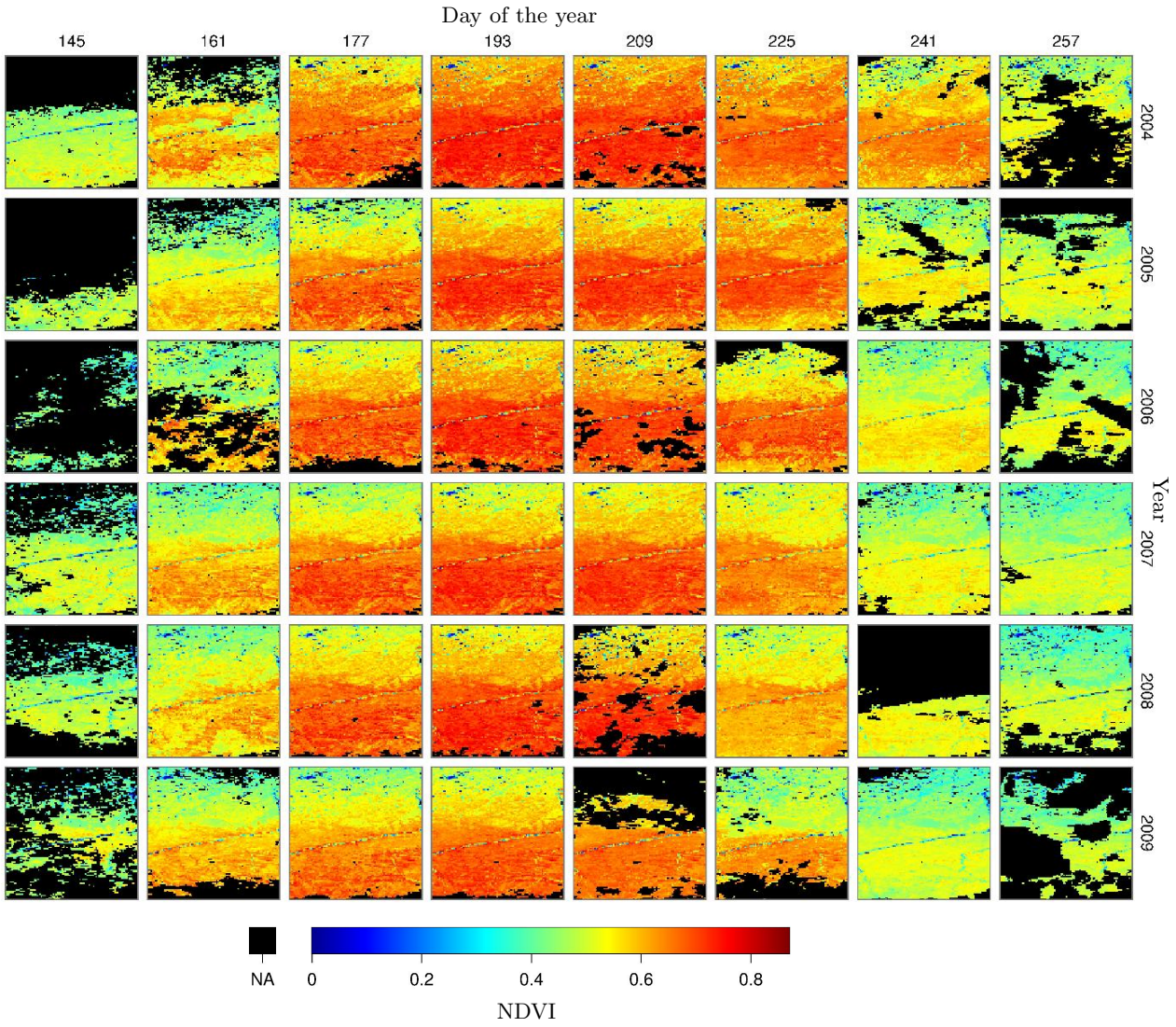

Figure S2: Test data set with 20% missing values. Shown are "good quality" NDVI values from the MODIS satellite product MOD13A1, which were transformed to the geographic map projection (WGS84). Some values were set to NA according to patterns of missing values observed at other locations (Section 3). Missing values are shown in black.

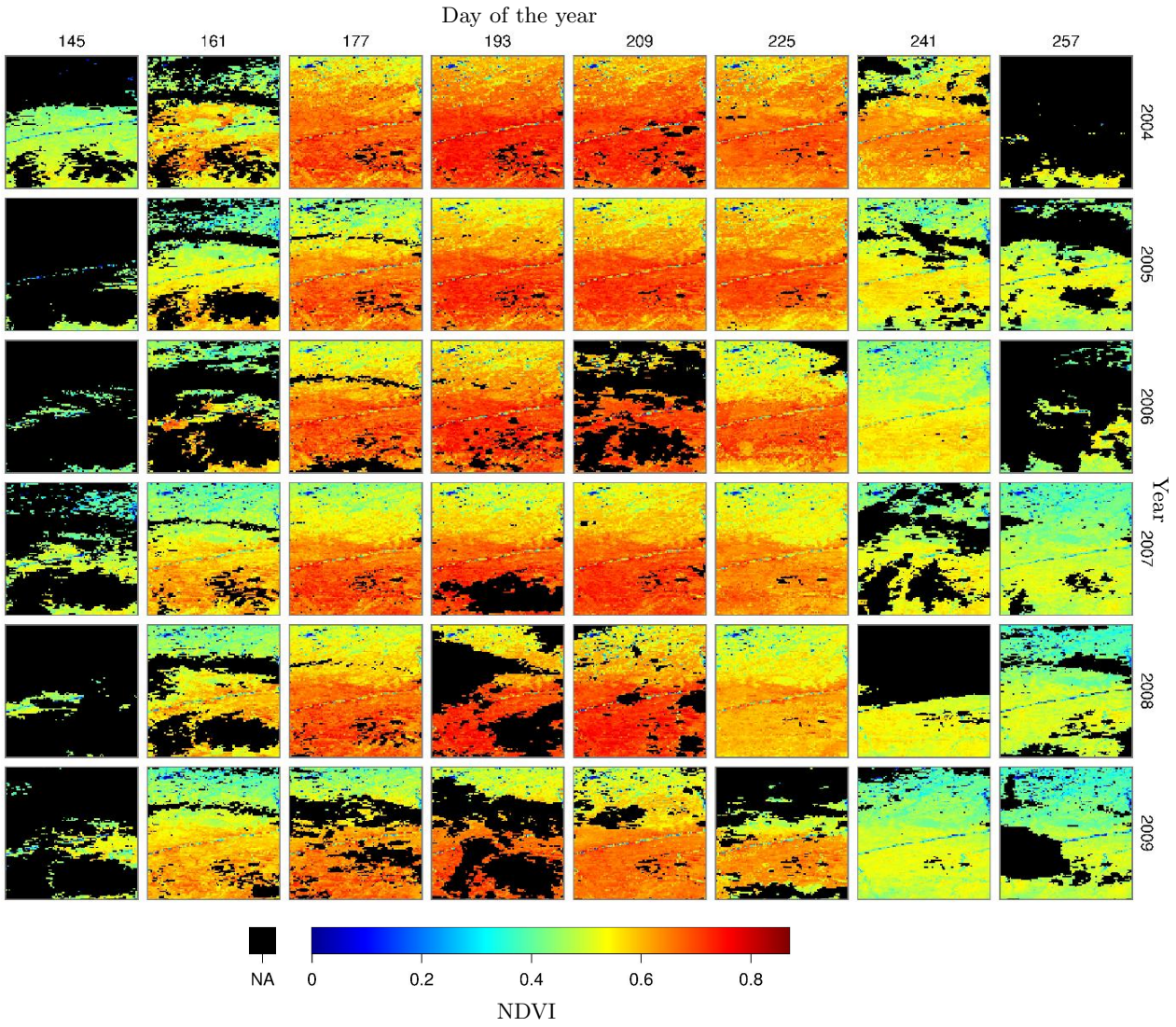

Figure S3: Test data set with 30% missing values. Shown are "good quality" NDVI values from the MODIS satellite product MOD13A1, which were transformed to the geographic map projection (WGS84). Some values were set to NA according to patterns of missing values observed at other locations (Section 3). Missing values are shown as black pixels.

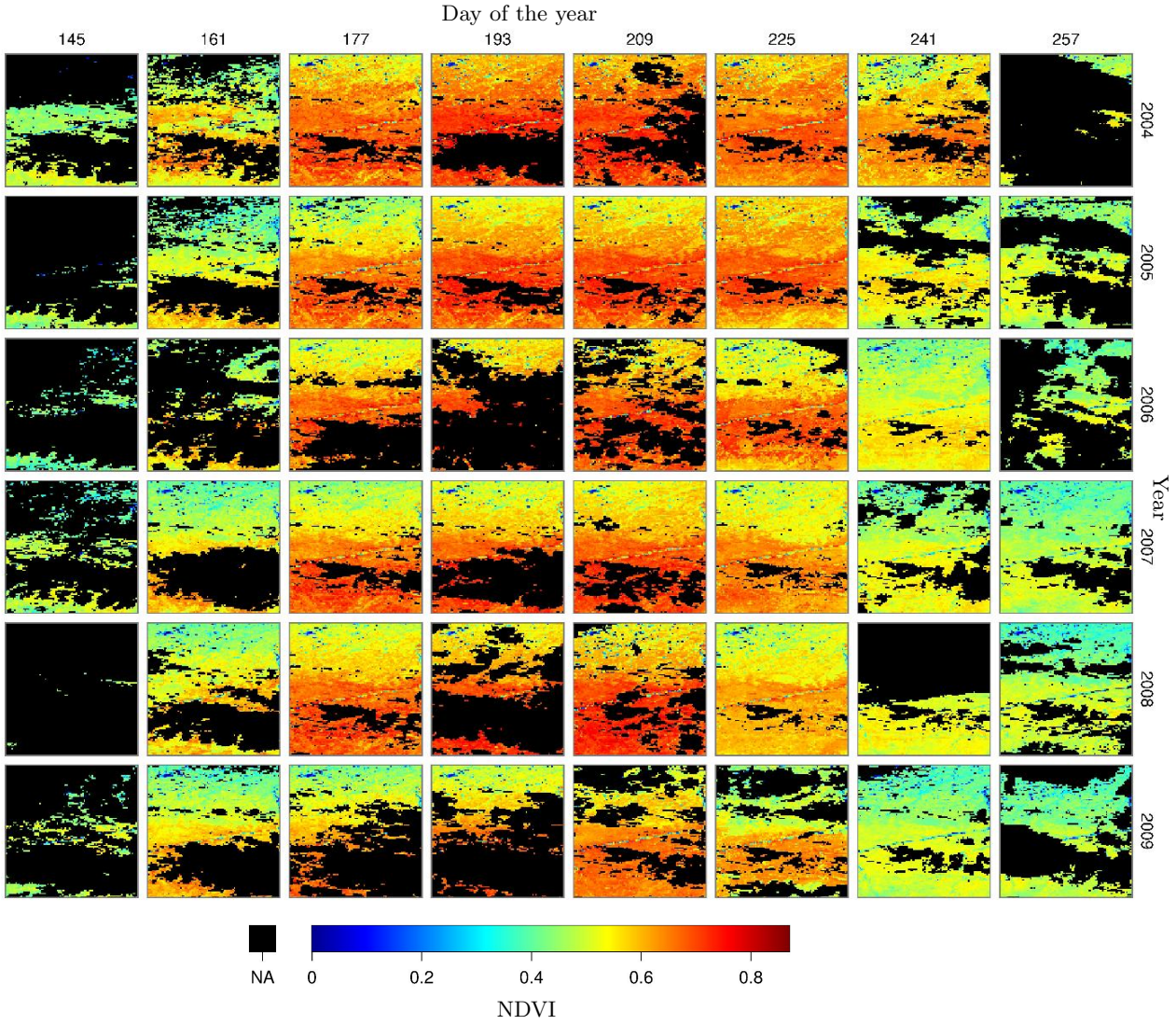

Figure S4: Test data set with 40% missing values. Shown are "good quality" NDVI values from the MODIS satellite product MOD13A1, which were transformed to the geographic map projection (WGS84). Some values were set to NA according to patterns of missing values observed at other locations (Section 3). Missing values are shown in black.

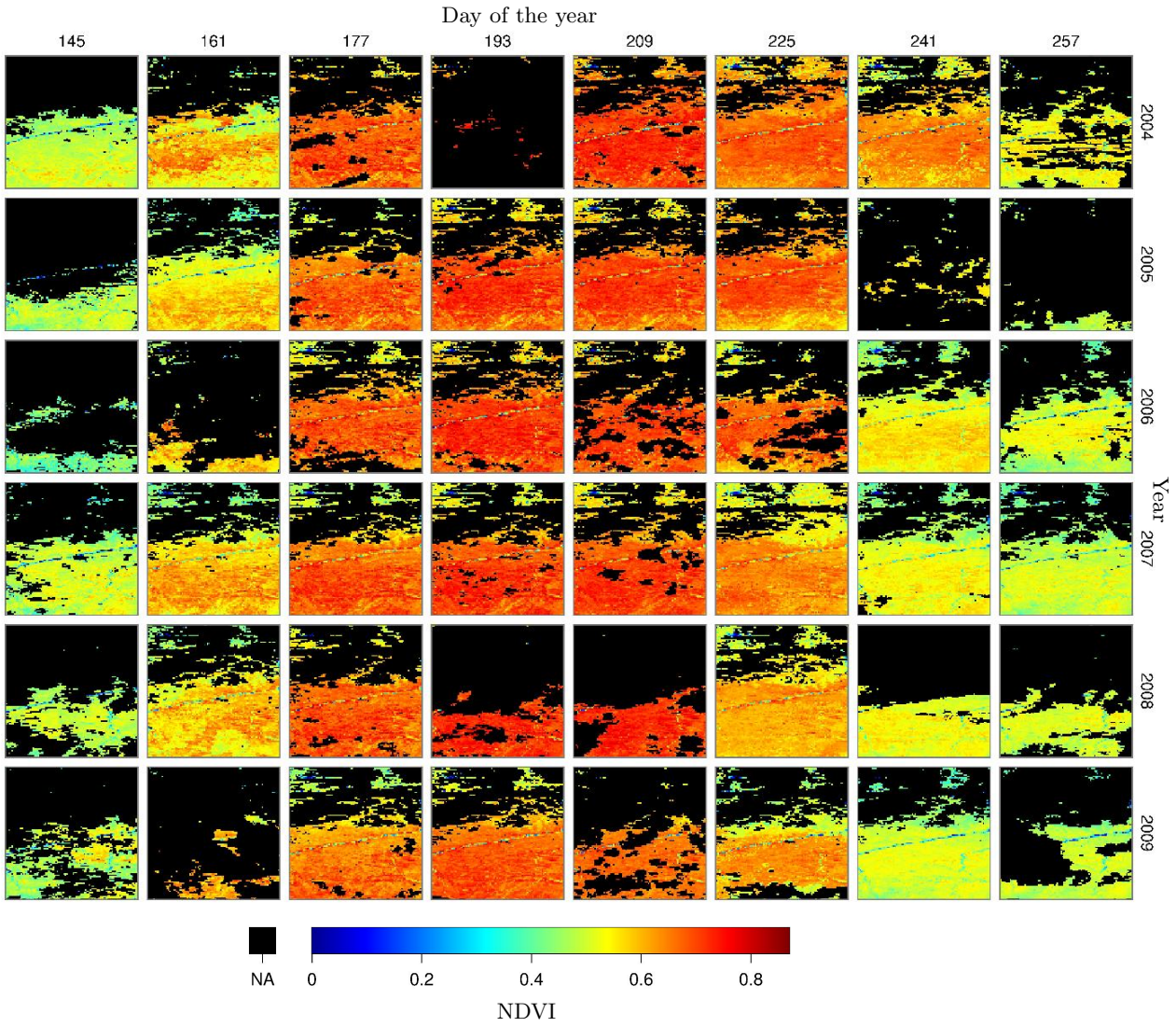

Figure S5: Test data set with 50% missing values. Shown are "good quality" NDVI values from the MODIS satellite product MOD13A1, which were transformed to the geographic map projection (WGS84). Some values were set to NA according to patterns of missing values observed at other locations (Section 3). Missing values are shown in black.

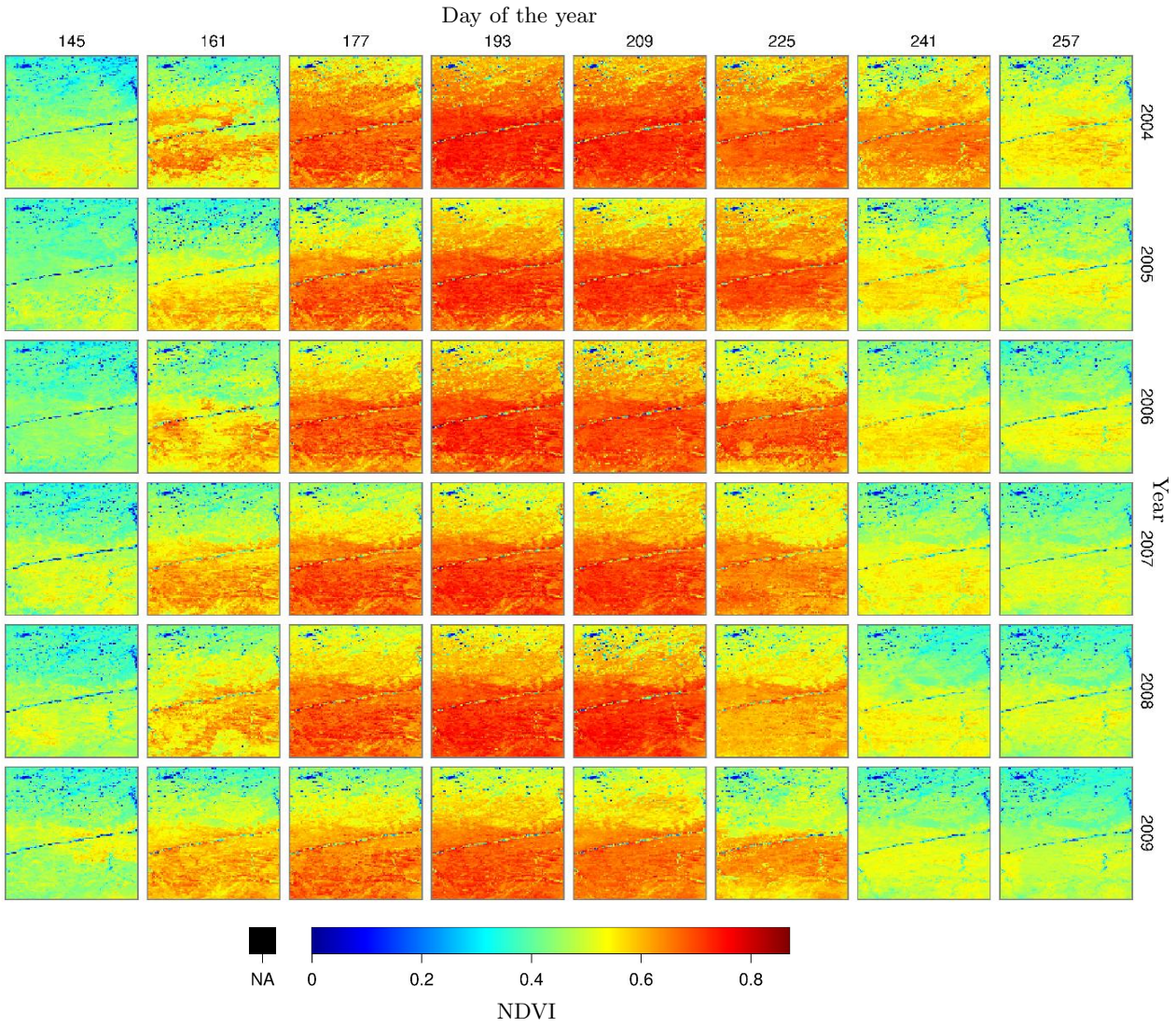

Figure S6: The test data set with 20% missing values (Fig. S2) and the predicted values from *gapfill*. No missing values remained in the output data set.

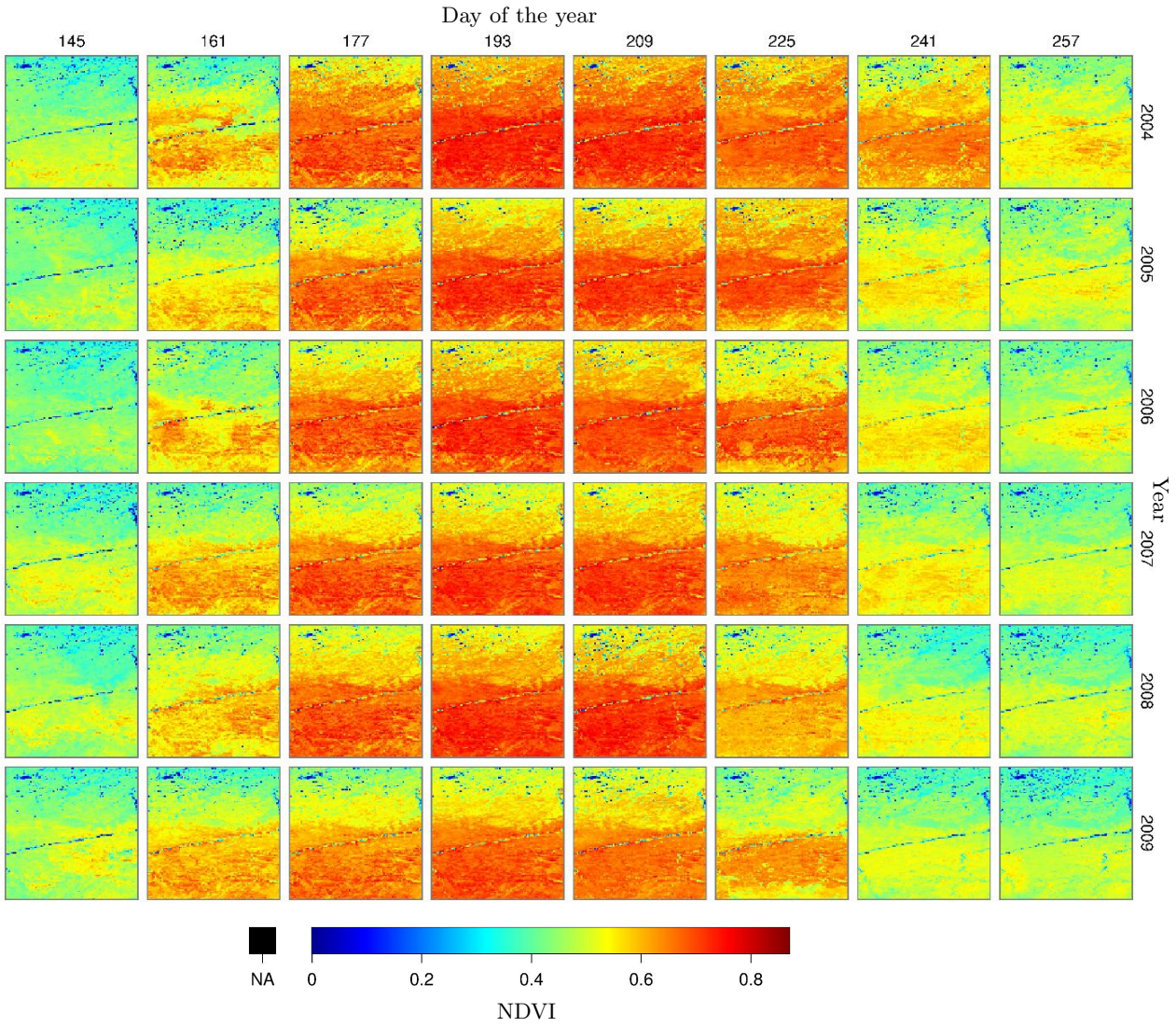

Figure S7: The test data set with 30% missing values (Figure S3) and the predicted values from *gapfill*. No missing values remained in the output data set.

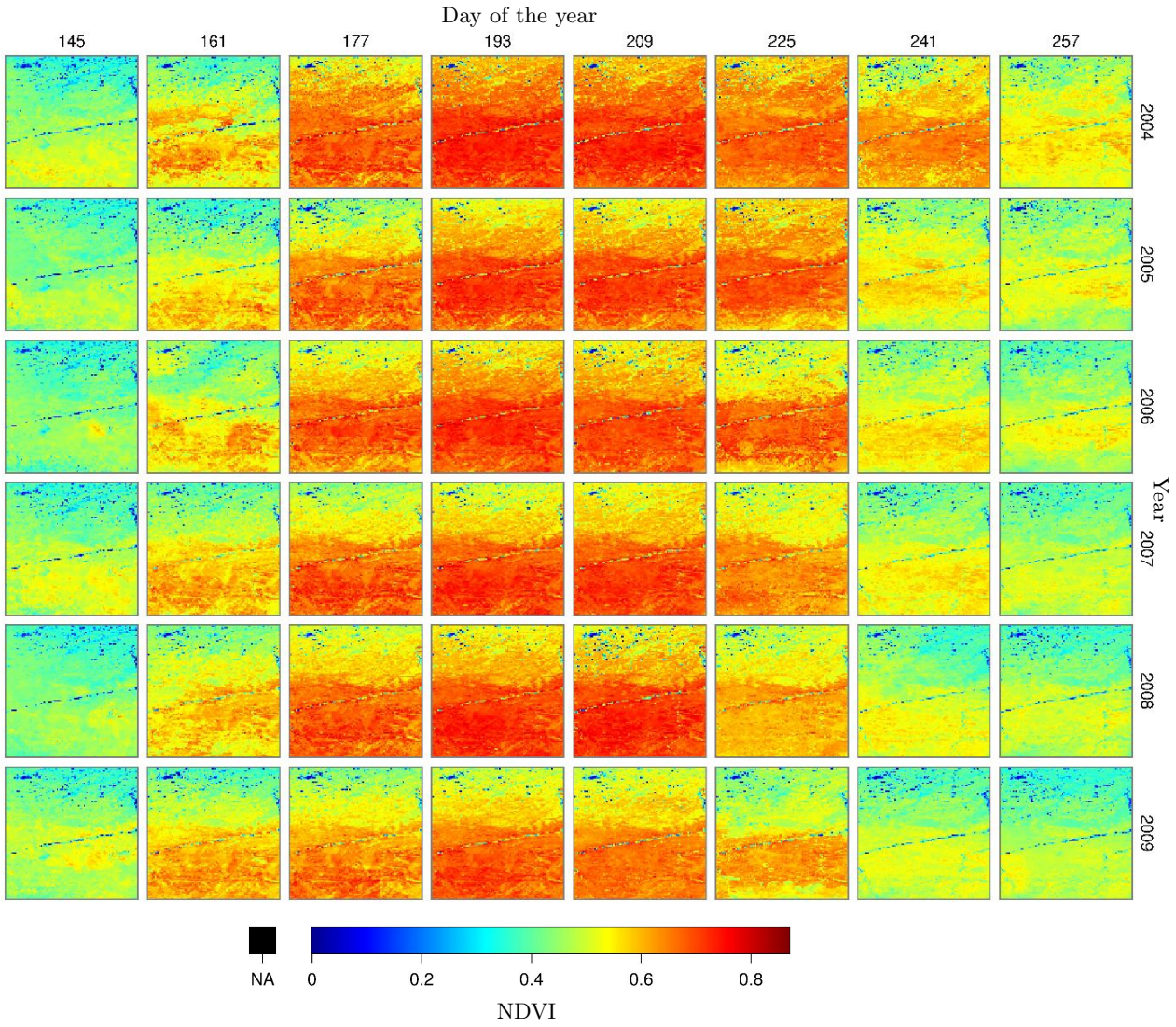

Figure S8: The test data set with 40% missing values (Figure S4) and the predicted values from *gapfill*. No missing values remained in the output data set.

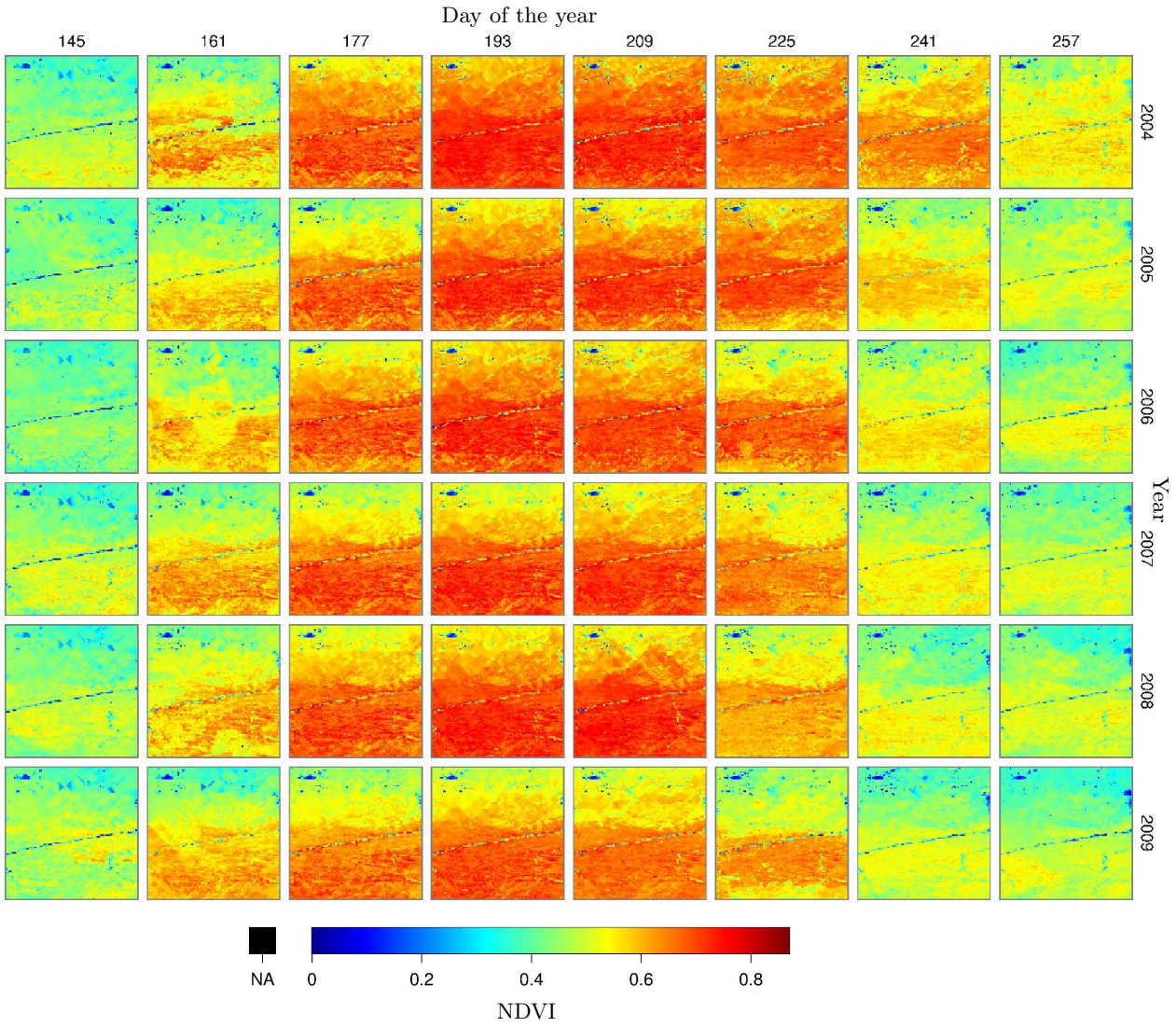

Figure S9: The test data set with 50% missing values (Figure S5) and the predicted values from *gapfill*. No missing values remained in the output data set.

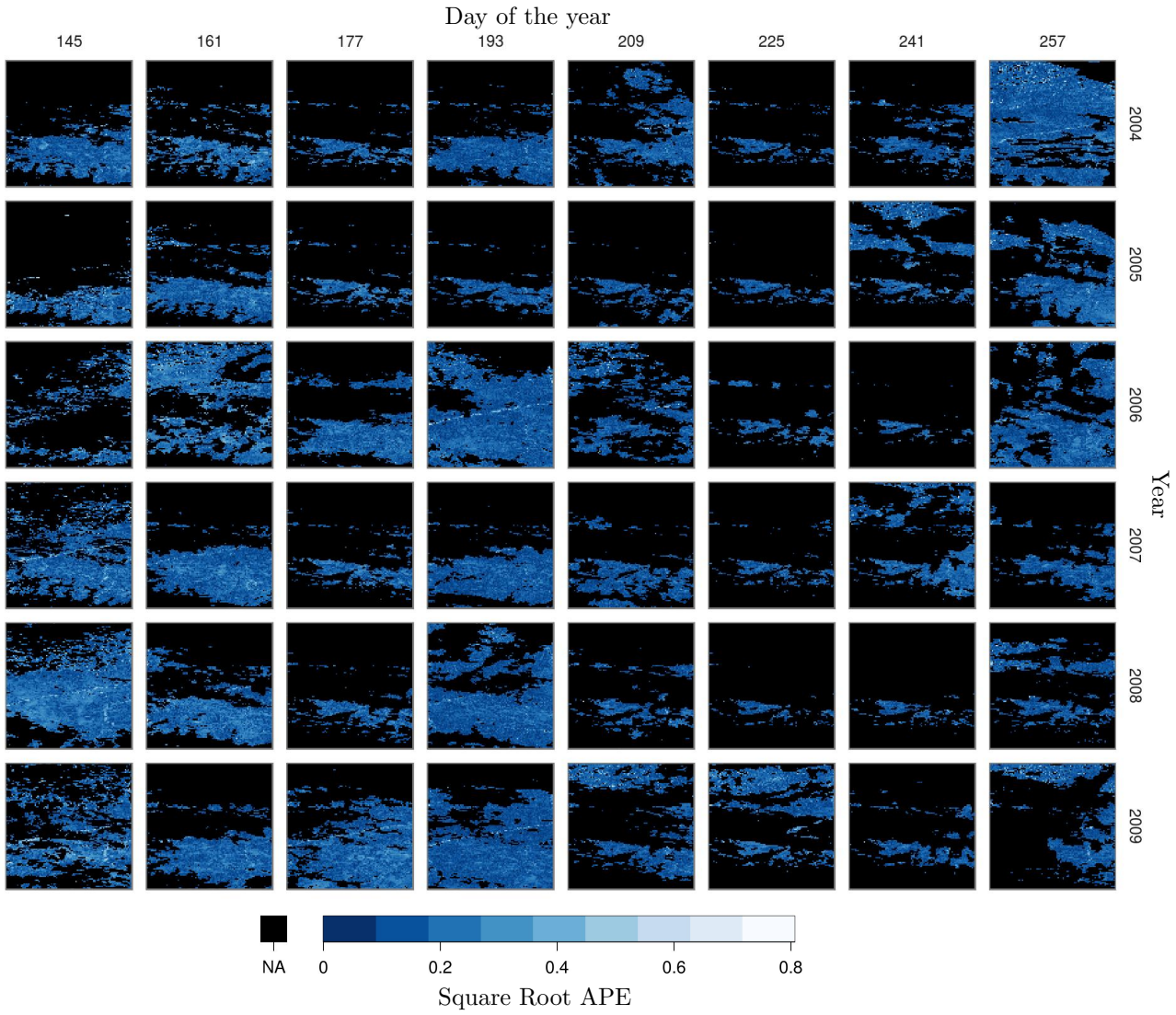

Figure S10: Square root APEs of the predicted values for the test data set with 40% missing values (Fig. S4). The R package *gapfill* was used to predict the missing values. All missing values are predicted, i.e., observed values of the test data set are shown in black.

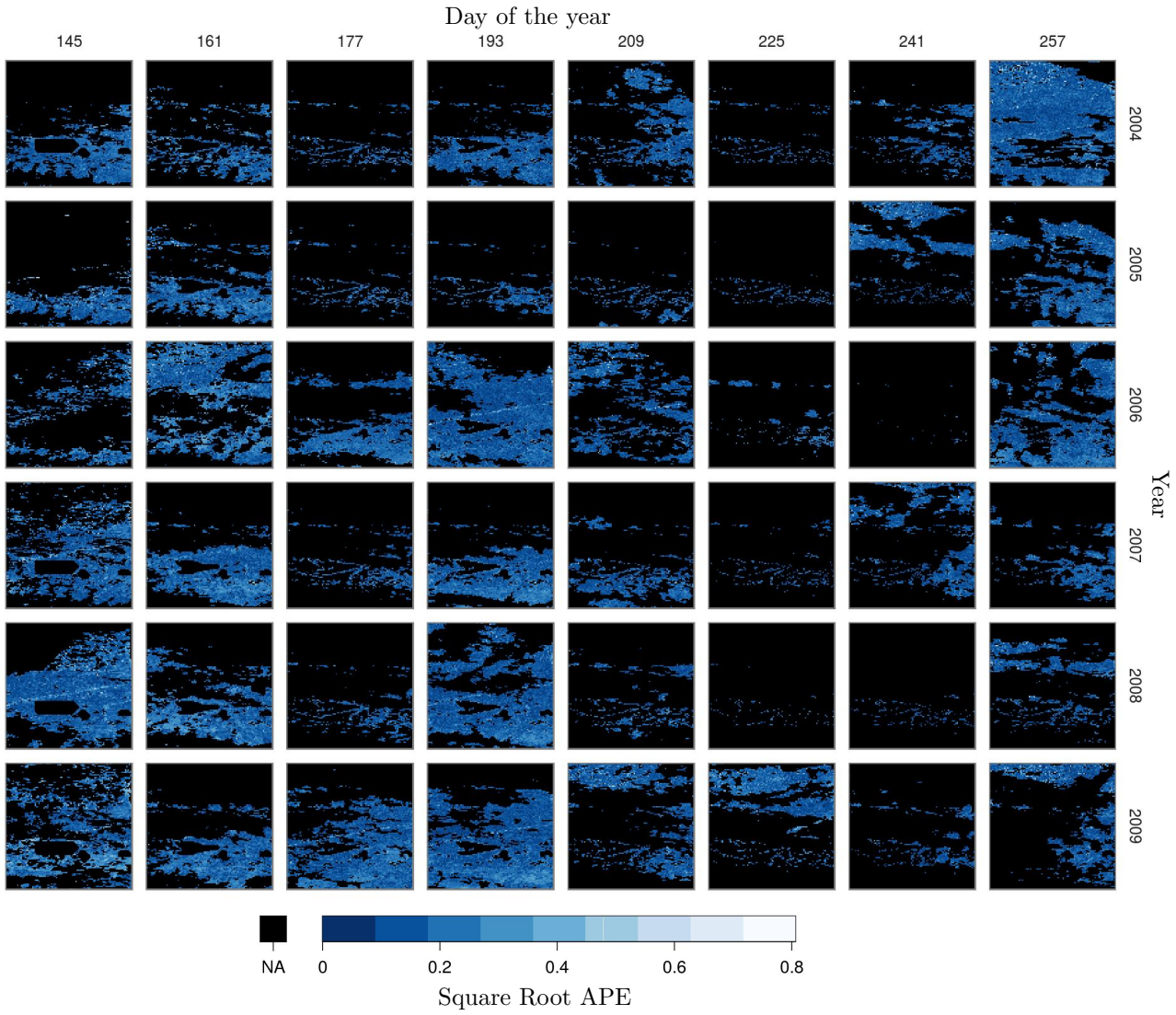

Figure S11: Square root APEs of the predicted values for the test data set with 40% missing values (Fig. S4). gapfill-MAP was used to predict the missing values. Only 88% of the missing values were predicted. Hence, black indicates either observed values of the test data set or missing values without predicted values.

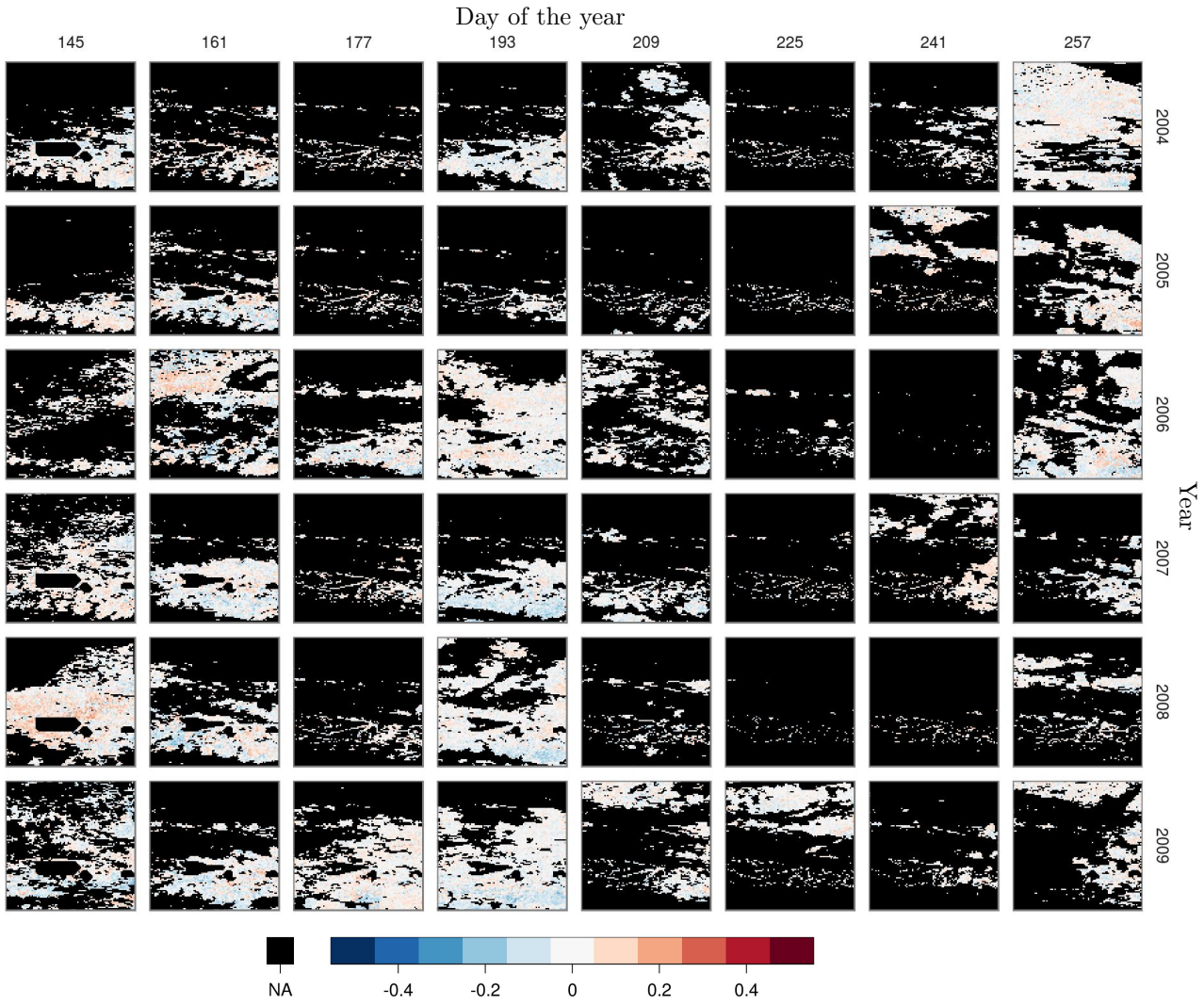

Figure S12: Differences between the square root APEs of the predicted values from *gapfill* and gapfill-MAP for the test data set with 40% missing values (Figure S4). Red indicates that the APE of *gapfill* is larger compared to the APE of gapfill-MAP. Conversely, blue indicates that the APE of *gapfill* is smaller compared to the APE of gapfill-MAP. Black indicates either observed values of the test data set or missing values without predicted values from gapfill-MAP.

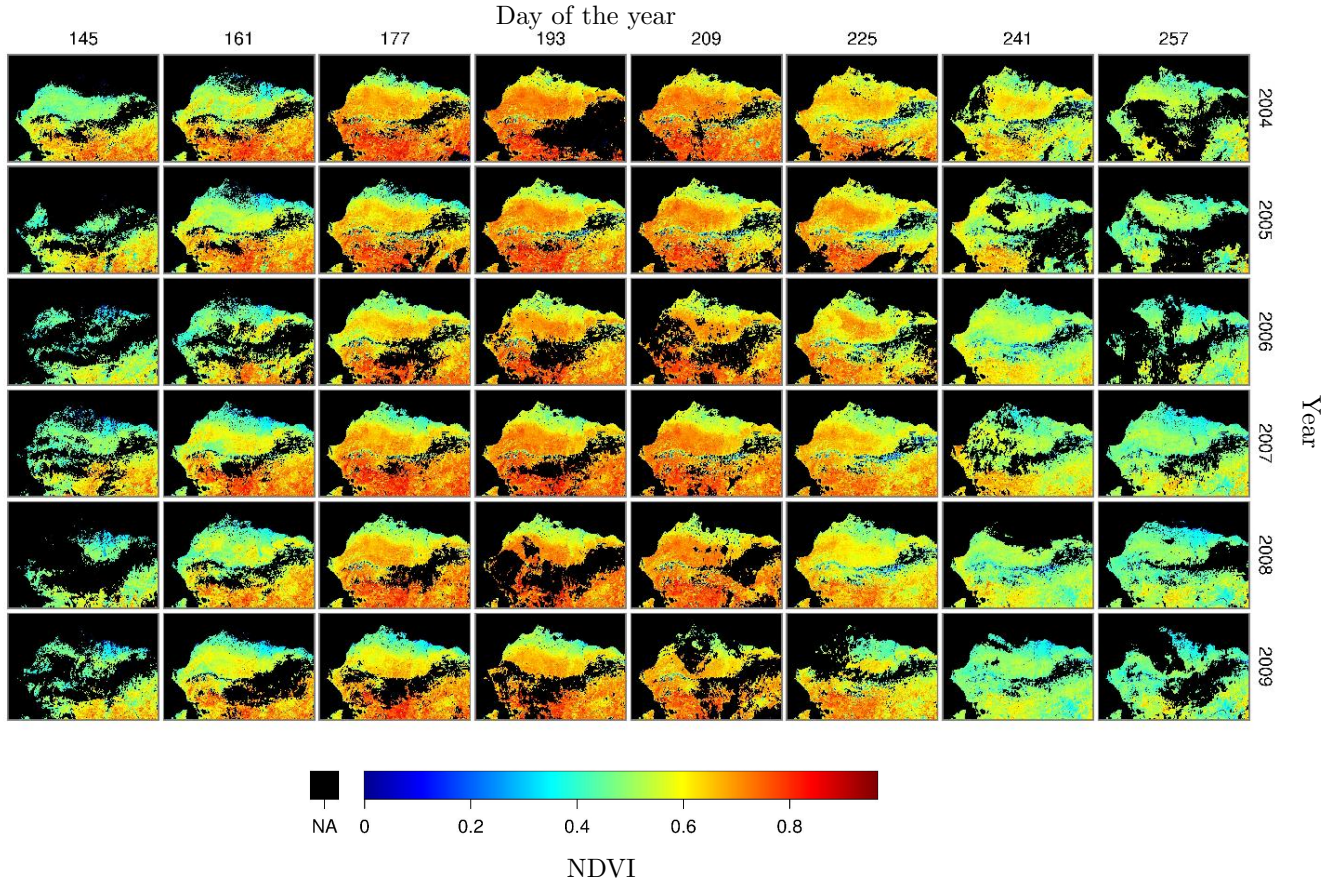

Figure S13: NDVI data consisting of the "good quality" values of the MODIS MOD13A1 satellite product transformed to the geographic map projection (WGS84). One image consists of 271′819 non-sea pixels. In total 28% of the non-sea values are missing and shown in black.

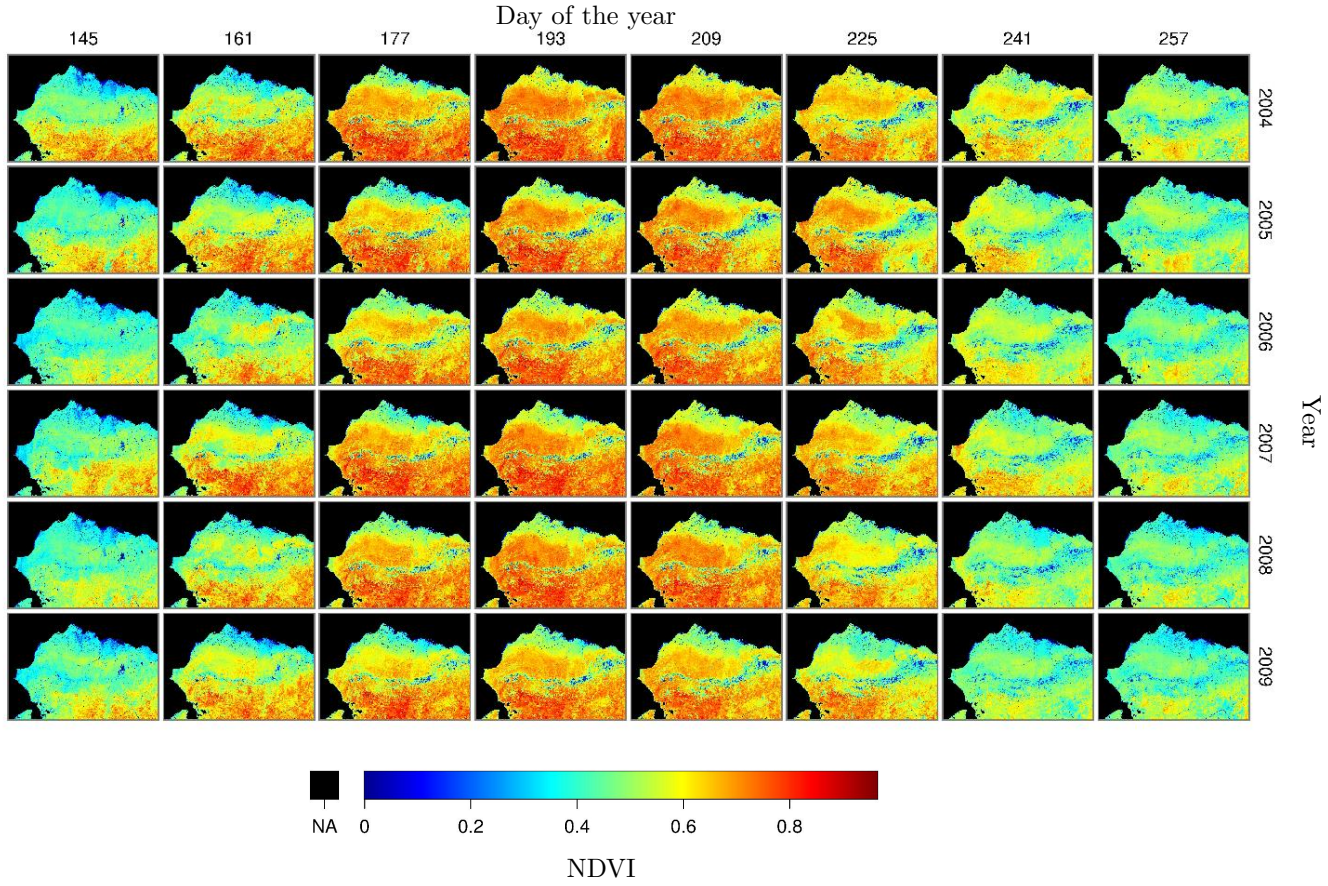

Figure S14: The Alaska NDVI data set (Figure S14) and the predicted values from *gapfill*. No missing non-sea values remained in the output data set.

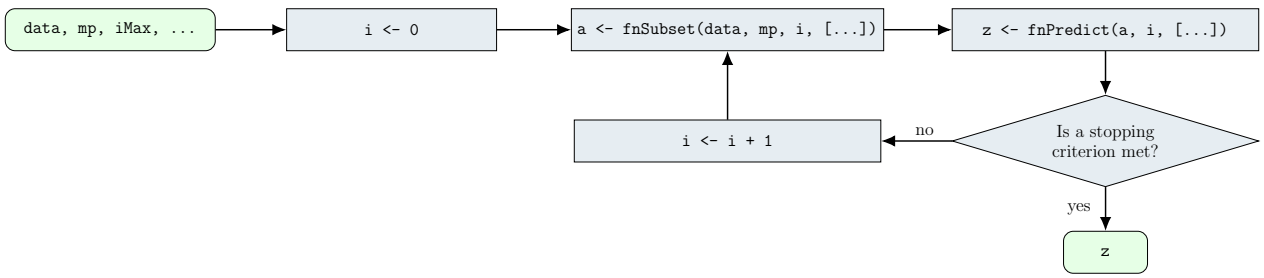

Figure S15: Schematic work-flow of the `Gapfill()` function. The main input R objects are `data`, `mp`, `iMax`, and further arguments "`...`" passed to the `fnSubset()` and/or `fnPredict()` functions. To predict the missing value at position `mp`, the `fnSubset()` and `fnPredict()` functions are called sequentially. `i` counts the number of unsuccessful predictions of the missing value. If one of the stopping criteria (1)–(3) is met, the algorithm stops and returns `z`.

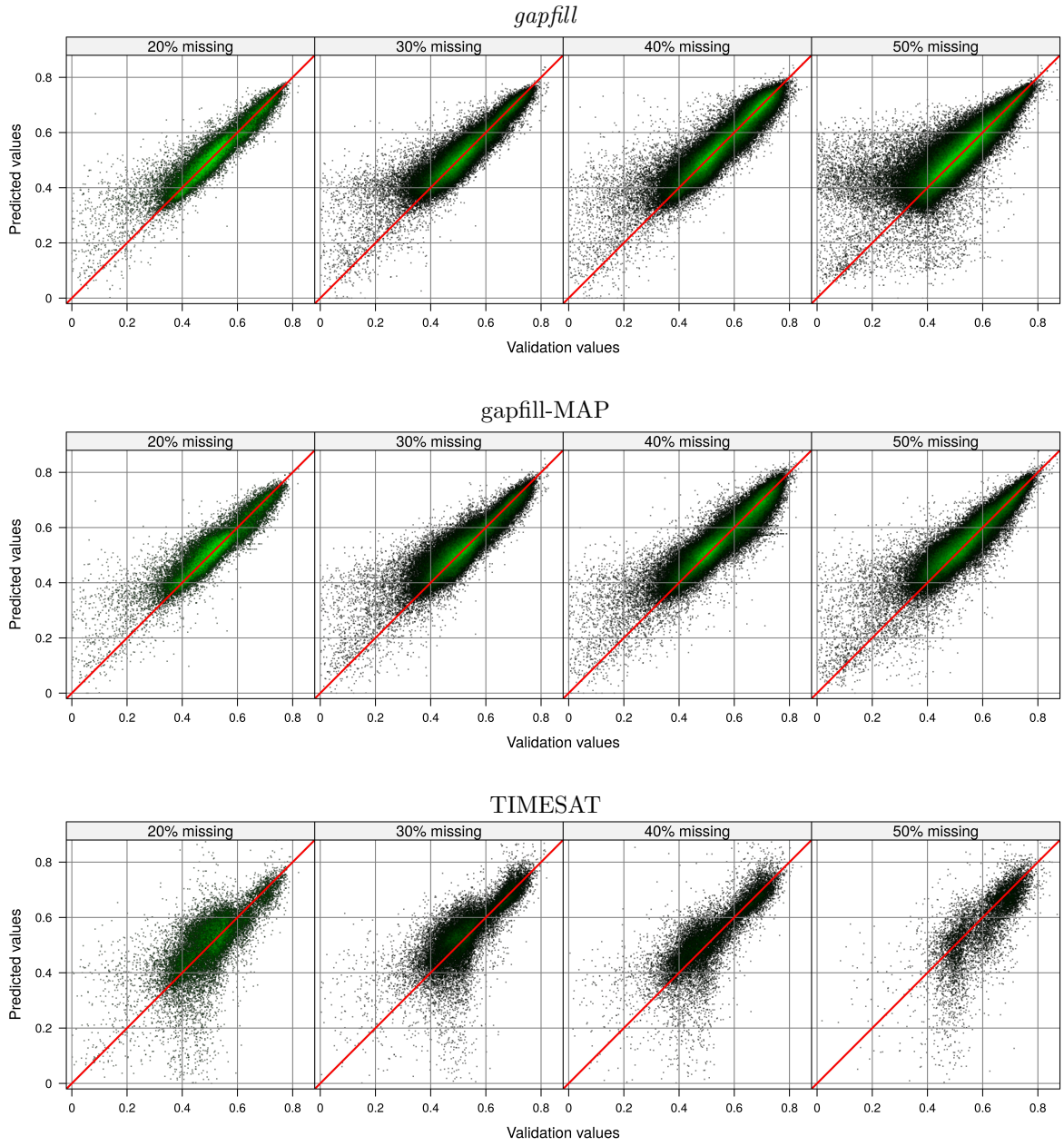

Figure S16: Scatter plots of the predicted values (y-axis) versus the validation values (x-axis) obtained from *gapfill*, gapfill-MAP, and TIMESAT. The green color shading indicates regions with a large density of points; light green corresponds to 100 overlaying points. Note that gapfill-MAP and TIMESAT did not return predictions for all missing values making it difficult to compare the methods with that figure.

S4 Listings

Listing S1: Example R script illustrating the function call used to obtain the predicted values of the test data sets.

```
## install and load package
install.packages("gapfill")
library("gapfill")

## load data (available upon request from the authors)

out <- Gapfill(data,
               clipRange = c(0, 1),
               dopar = TRUE,
               initialSize = c(10, 10, 1, 6))
```

Listing S2: Tuning parameters used to run the gapfill-MAP.

```
# Should the fill be based on the difference between neighbour values or the ratio between them?
# Ratio only works for ratiometric variables such as temperature in Kelvin
_FILL_BY_RATIO = 0

# If we are running by ratio then set appropriate values
# Ff necessary to put the values into a "more absolute" scale e.g. to turn celsius into kelvin.
# Also set a max ratio in case there are values close to zero (like with EVI).
_DATA_ZERO_OFFSET = 0.0
_MAX_ALLOWABLE_RATIO = 5.0 # values close to 0 will give large ratio, so define a limit

##### Other data "fixes"?
# if the data were converted wrong from HDF then fix that here
_DATA_CORRECTION_OFFSET = np.float32(0)

##### Fill algorithm thresholds and distances
_TRIM_MIN_MAX = 1
# Should the fill values be clipped?
_CLIP_TO_LIMITS = 1

##### Despeckle thresholds
# Number of stds beyond the mean beyond which a fill value will be clipped
_FLOOR_CEILING_VALUE = 2.58
# Number of stds beyond the mean beyond which a value will be unconditionally discarded in despeckle
_EXTREME_VALUE_THRESHOLD = 2.58
# Number of stds beyond the mean beyond which a value may be discarded in despeckly,
#        depending on neighborhood similarity
_SPECKLE_THRESHOLD = 1.64 #1.96
# Max difference in Z-score between a potential speckle and the average of its neighbours, for it
# to be accepted as not being a speckle
_SPECKLE_NBR_Z_THRESH = 0.2

##### Spiral search distance / density (-> search radius)
# How many cells should the spiral search in despeckle check up to (unless
# it finds enough nbrs beforehand)
# effective search radius is therefore sqrt(value / pi)
_DESPECKLE_SEARCH_NBRS = 31
# How many cells must be found within the above figure to succeed?
_DESPECKLE_MIN_USED_NBRS = 16
# How many cells will we go up to before we stop searching, if we find them
# before the end of the spiral search is reached?
_DESPECKLE_MAX_USED_NBRS = 32

# How many cells should the spiral search in A1 check up to (unless it finds
# enough nbrs beforehand)
# A1 search radius is therefore approx sqrt (value / pi)
_A1_SEARCH_NBRS = 31420 ## total image
# At least how many cells must be found within above numnbrs to succeed?
# Bear in mind that the spiral will run (n years - 1) times and this value
# applies to the entire search in total
_A1_MIN_USED_NBRS = 40
# Stop the spiral search when we have found how many nbrs?
_A1_MAX_USED_NBRS = 999999
```

Listing S3: Setting file used to run the TIMESAT software.

```
Version: 3.2
mask_study50    %Job_name (no blanks)
0               %Image /series mode (1/0)
0               %Trend (1/0)
0               %Use mask data (1/0)
/compute/flgerb/Gapfill/TIMESAT/data/data50.txt         %Data file list/name
none         %Mask file list/name
1               %Image file type
0               %Byte order (1/0)
0 0             %File dimension (nrow ncol)
0 0 0 0         %Processing window (start row stop row start col stop col)
6 8             %No. years and no. points per year
0 1             %Valid data range (lower upper)
0 0 0           %Mask range 1 and weight
0 0 0           %Mask range 2 and weight
0 0 0           %Mask range 3 and weight
0               %Amplitude cutoff value
0               %Print functions and weights (1/0)
1 1 0           %Output files (1/0 1/0 1/0)
0               %Use land cover (1/0)
none            %Name of landcover file
0               %Spike method
2               %Spike value
0               %Spatial half dimension
1               %No. of landcover classes
************
1               %Land cover code for class  1
1               %Season parameter
1               %No. of envelope iterations (1-3)
1               %Adaptation strength (1-10)
0 -99999        %Force minimum (1/0) and value
3               %Fitting method (1-4)
1               %Weight update method
1               %Window size for Sav-Gol.
1               %Spatio-temporal smoothing factor.
1               %Spatio-temporal adaptation factor.
1               %Season start method
0.5 0.5         %Season start / stop values
```

S5 Comparing *gapfill* and gapfill-MAP using a tropical MODIS EVI data set

We compare the predictions from *gapfill* and gapfill-MAP using a MODIS EVI data set from the Brazilian Amazon. We use the enhanced vegetation index EVI instead of the NDVI as it better captures vegetation activity in regions where the NDVI is expected to be saturated (Huete at al. "Overview of the radiometric and biophysical performance of the MODIS vegetation indices", DOI:10.1016/S0034-4257(02)00096-2). Despite considering EVI instead of the NDVI, the same MODIS data product and preparation steps as described in Section III-A of the manuscript were used. We consider a spatial region of $100 \cdot 100$ values located in the tropical forest around the river Rio Uatumã in the State of Amazonas in Brazil (about 58.7° south and 2.3° east). We consider values observed between 2004 and

2009. As shown in Figure S17, the data from DOY 1 to DOY 145 and DOY 305 to DOY 353 contain only very few (less than 20%) observed values. Therefore, we restrict the further analysis to the seasonal subset from DOY 161 to DOY 286. The corresponding data set is shown in Figure S18. We created a validation data sets with 45% and 50% missing data by randomly setting values to NA. The resulting test data sets are shown in Figure S19 and Figure S20, respectively.

The test data sets and the predicted values obtained using *gapfill* are shown in Figure S21 and Figure S22. In most cases the predictions recover the spatial structure of the images including small scale features. For 4 images there are no predicted values because those images contain less than 5 observed values. (Using the default configuration *gapfill* requires at least 5 observed values per image to make predictions for an image.) For some images with very few observed values, the predicted values exhibit artificial spatial structure; see e. g. , the lower left image in Figure S21. However, it is questionable whether predicting values of images that have only very view observed values can lead to sensible results.

For comparison, the missing values of both data sets were predicted using gapfill-MAP described in Section III-C of the manuscript. The predicted values from *gapfill* and gapfill-MAP are compared in Table S5. To make the numbers comparable, the RMSPEs and the MAPEs given in that table were calculated using only values that had non-NA predictions from both methods. The predicted values from *gapfill* are more accurate in terms of RMSPEs and MAPEs (Wilcoxon tests, all p-values $< 10^{-15}$). While gapfill-MAP was able to predict more missing values, it is questionable, if predictions for images with only a few observed values are helpful because of large prediction uncertainties. For illustration of the temporal profiles, the observed and predicted values from both methods are given for three spatial locations in Figure S23.

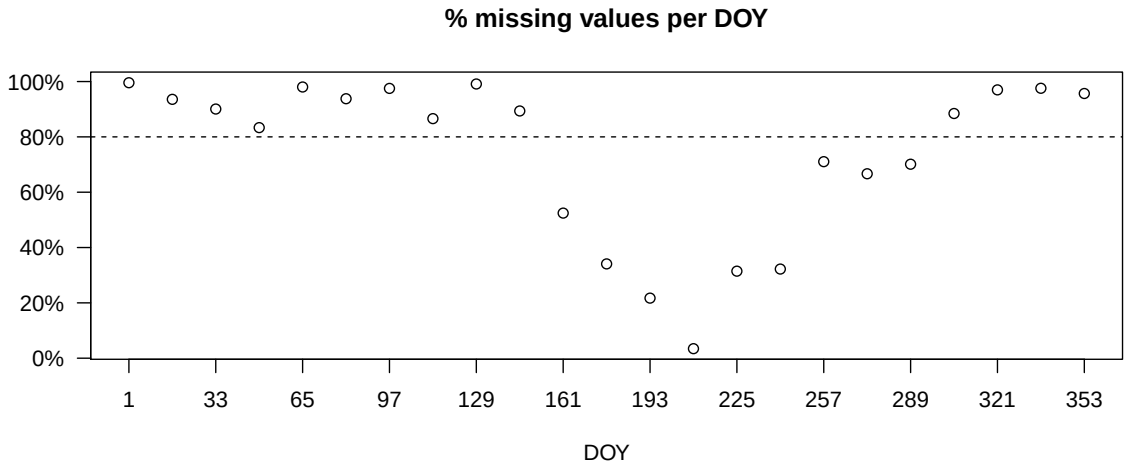

Figure S17: The percentage of missing values per DOY is shown for the selected MODIS EVI data set. We decided to restrict the comparison to DOYs with less than 80% missing values.

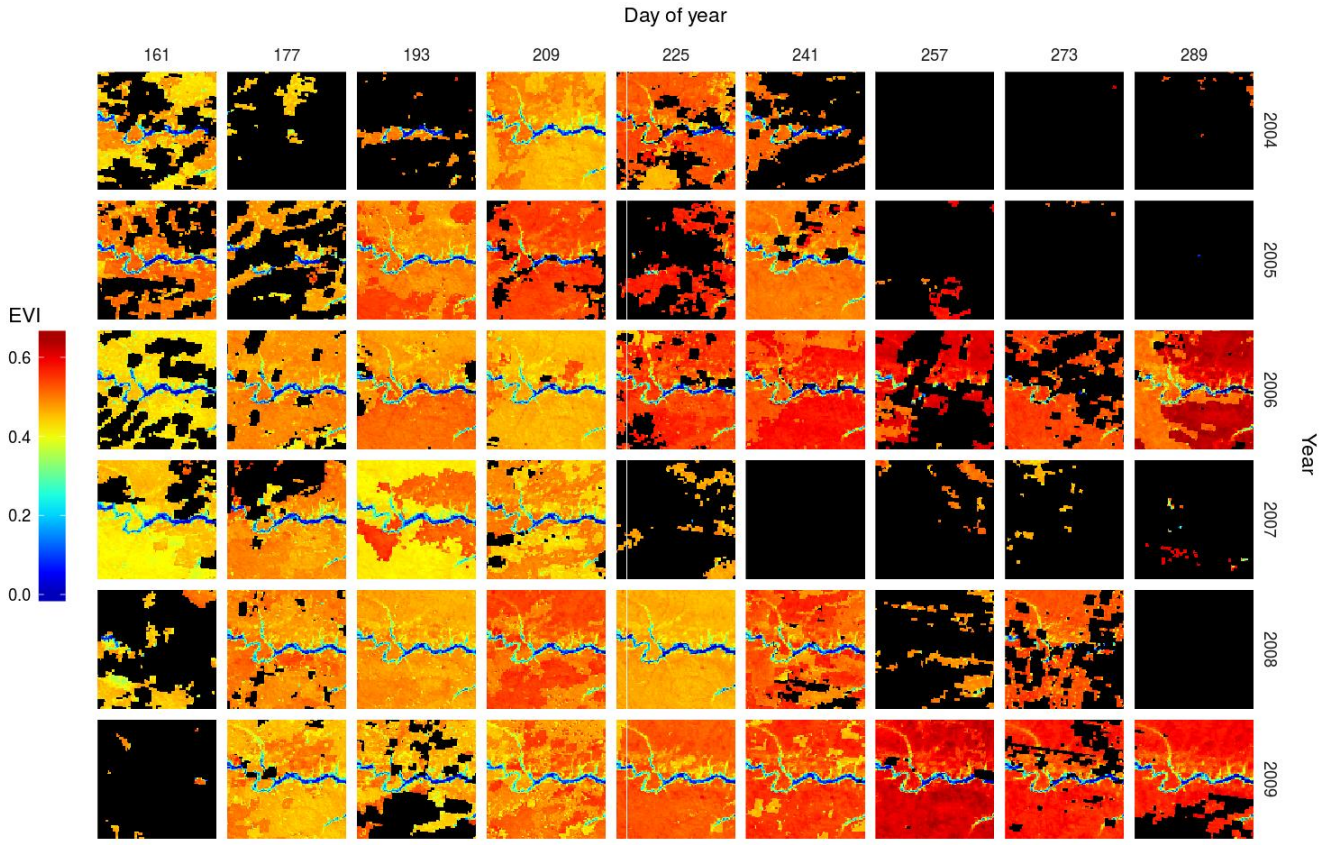

Figure S18: EVI data set used for the validation study. Missing values are shown in black.

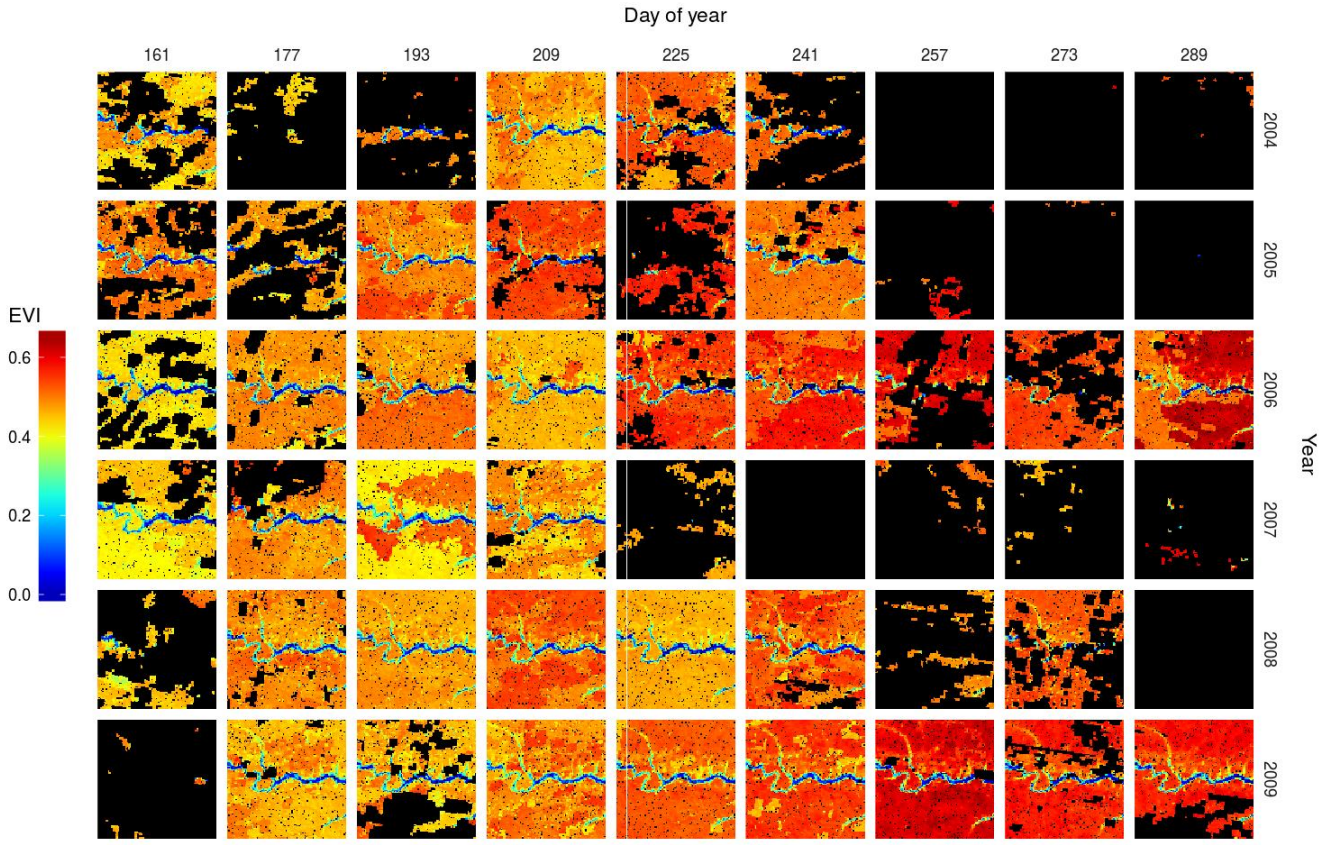

Figure S19: Data set of the validation study with 45% missing values. The data set was obtained by randomly removing values from the data set shown in Figure S18. Missing values are shown in black.

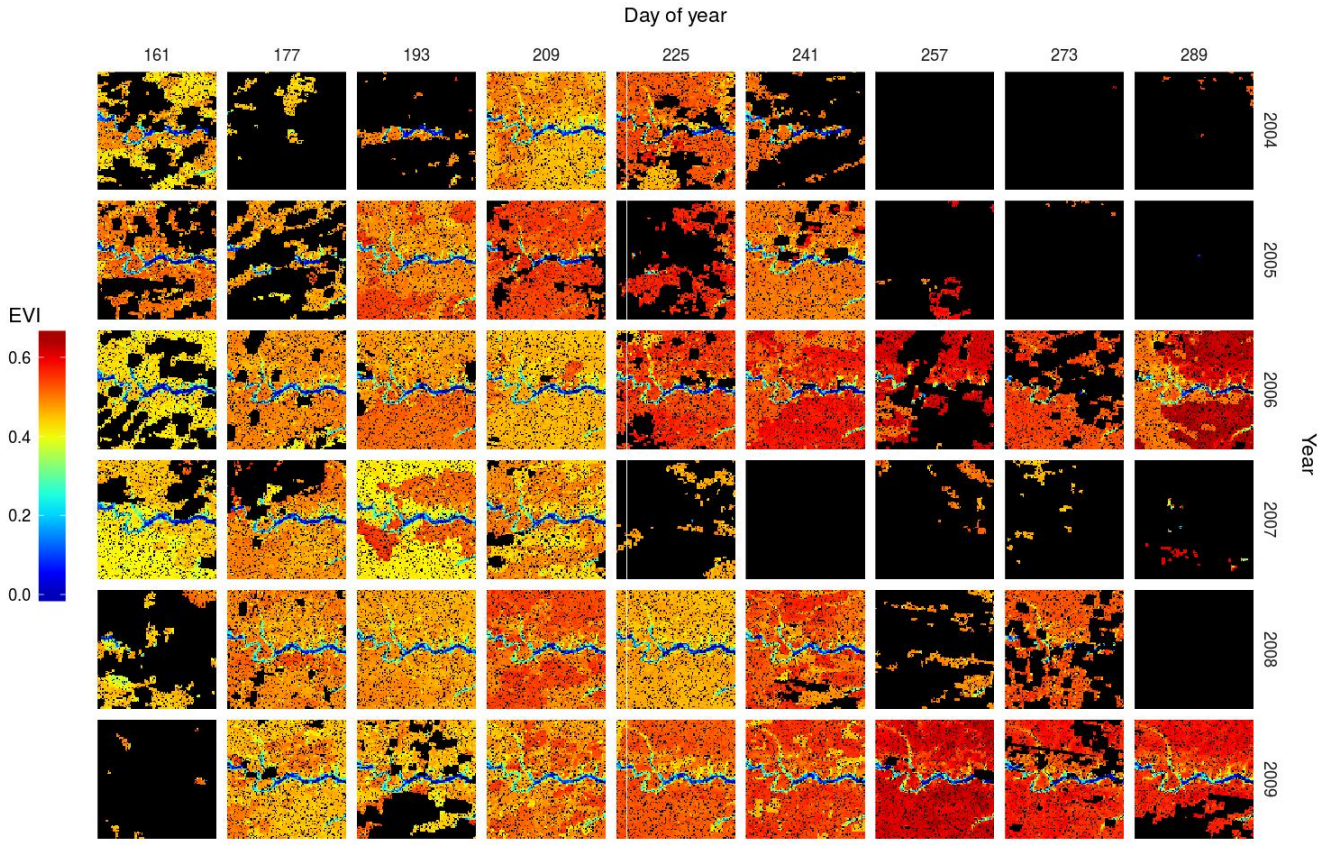

Figure S20: Data set of the validation study with 50% missing values. The data set was obtained by randomly removing values from the data set shown in Figure S18. Missing values are shown in black.

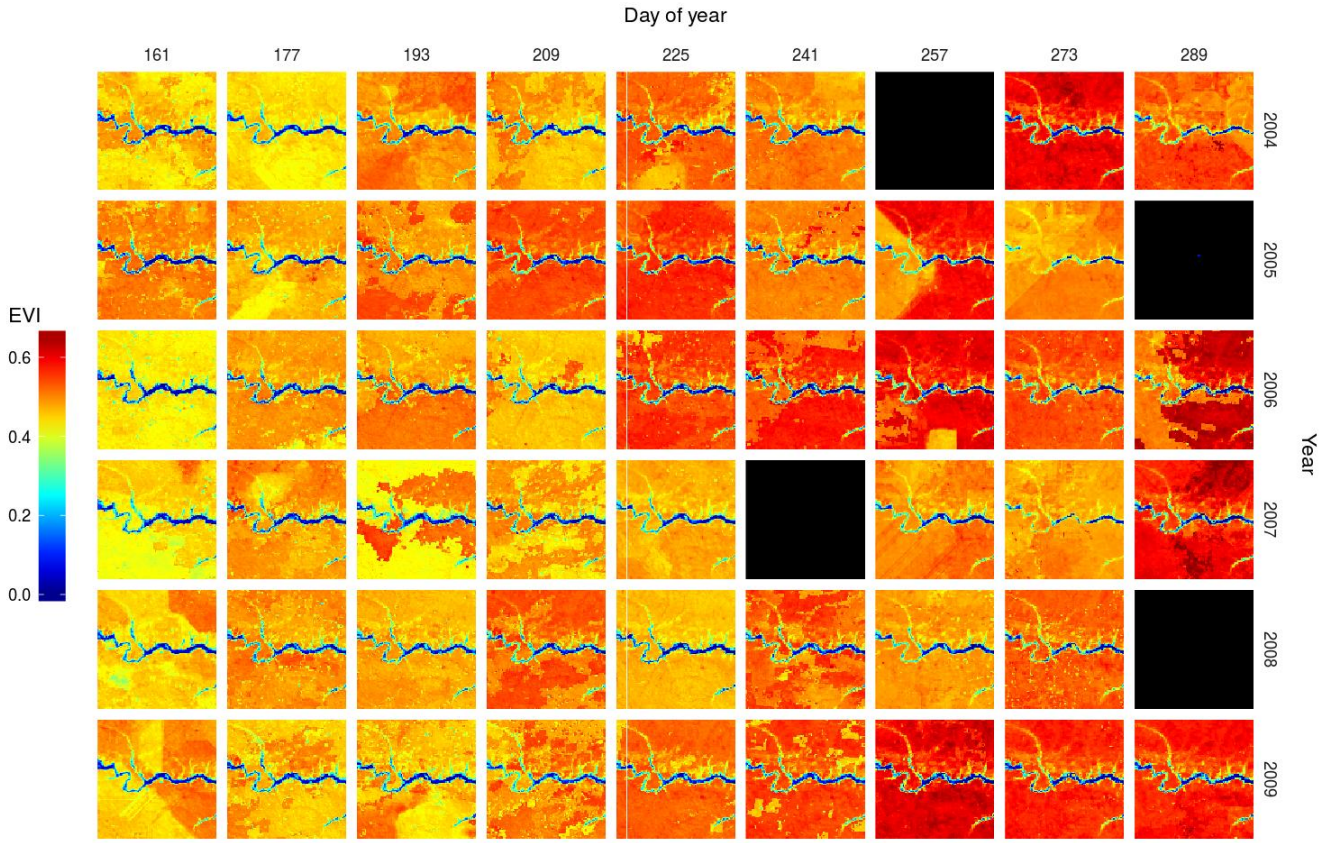

Figure S21: The test data set with 45% missing values (Figure S19) and the predicted values from *gapfill* are shown. Missing values are shown in black.

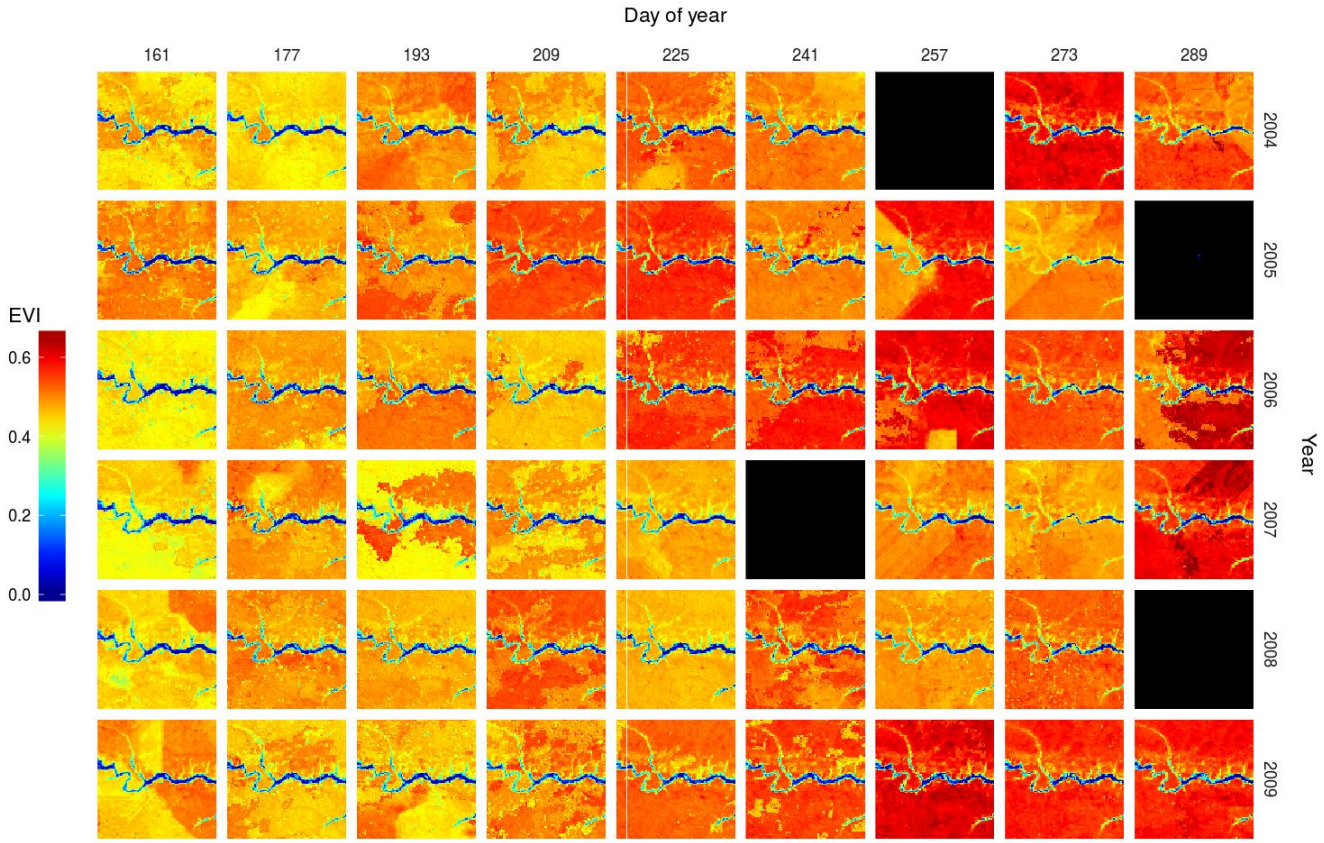

Figure S22: The test data set with 50% missing values (Figure S20) and the predicted values from *gapfill* are shown. Missing values are shown in black.

Table S5: The predicted values of the two test data sets with randomly removed values obtained with *gapfill* and gapfill-MAP are summarized in terms of the number and percentage of predicted values, the RMSPE·10^3 and the MAPE·10^3. The RMSPE and the MAPE were calculated for values with valid predictions from both methods in order to make them comparable.

	gapfill			gapfill-MAP		
	# predicted	RMSPE	MAPE	# predicted	RMSPE	MAPE
45%	203'004 (84%)	36.01	21.07	222'347 (92%)	73.82	56.22
50%	230'004 (85%)	36.60	21.16	248'346 (91%)	73.87	56.33

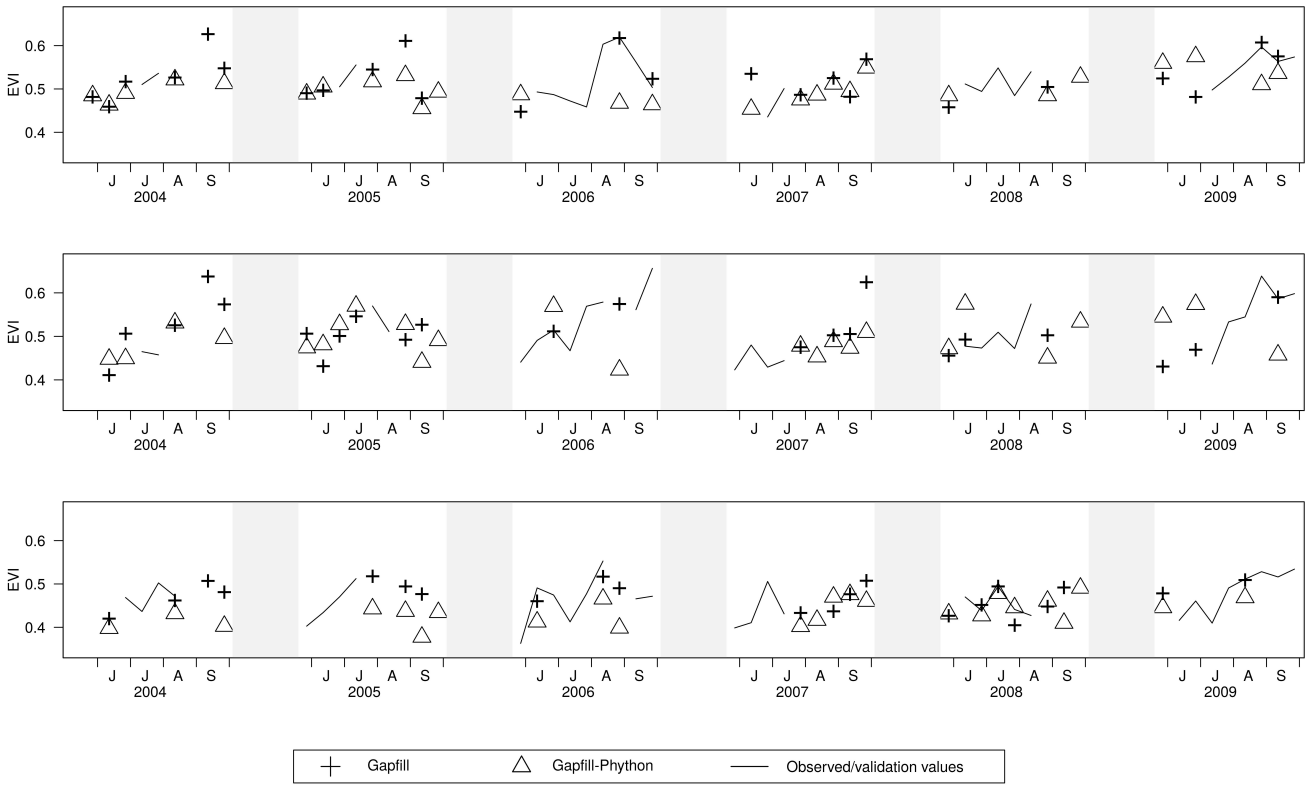

Figure S23: Temporal profiles for three spatial locations of the data set with 50% missing values shown in Figure S20. The locations are selected such that they represent locations with large (top), average (middle), and low (bottom) values. When validation or observed values were available, they are displayed with a black line. The gray areas represent the time span from November to mid May, which was excluded from the analysis due to the very large proportion of missing values.