Supplementary Material for *Geophysical constraints on the reliability of solar and wind power in the United States for 1980-2015*

Matthew R. Shaner[1], Steven J. Davis[1,2], Nathan S. Lewis[1,3], and Ken Caldeira[1]

[1] Carnegie Energy Innovation, Carnegie Institution for Science, Stanford, CA
[2] Department of Earth System Science, University of California, Irvine, Irvine, CA
[3] Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, CA

Ramification of Perfect Transmission and Energy Storage Assumptions

Transmission losses would increase the supply variability and would hence require additional storage. The details depend on the specific architecture of the grid being deployed, but localization would generally decrease the spatial decorrelation of the resource. In contrast, losses associated with charge/discharge energy storage cycling would increase the required storage capacity but would not affect the correlation of the resource.

Storage and Overcapacity

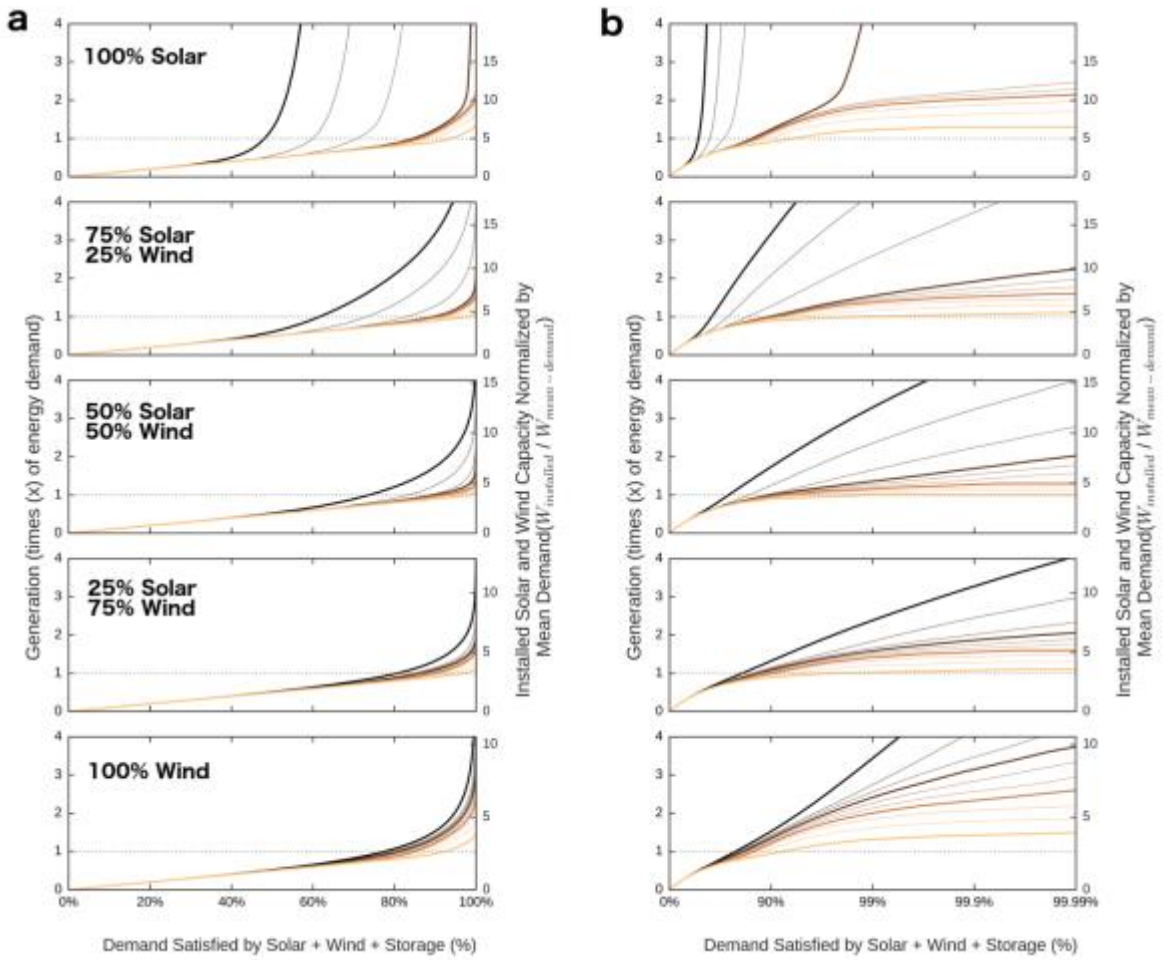

Figure S 1| Change in reliability as a function of energy storage capacity (1-30 days) and overcapacity. Lines in each panel show the reliability (% of demand met; x-axes, **a** linear scale, **b** log scale) as a function of solar and wind generation normalized by demand (y-axis) and energy storage capacity for the specified solar and wind mix aggregated over the contiguous U.S. The lines represent energy storage capacities of 0, 2, 4, 8, 12 and 24 hours and 2, 4, 8, 16 and 32 days; from top left to bottom right the storage size increases monotonically with 0 and 12 hours and 4 and 32 days shown as thicker lines. In all plots, the horizontal dashed line indicates a generation of 1x. The second y-axis on the right shows the installed capacity normalized by mean demand.

Characteristics of Unmet Demand

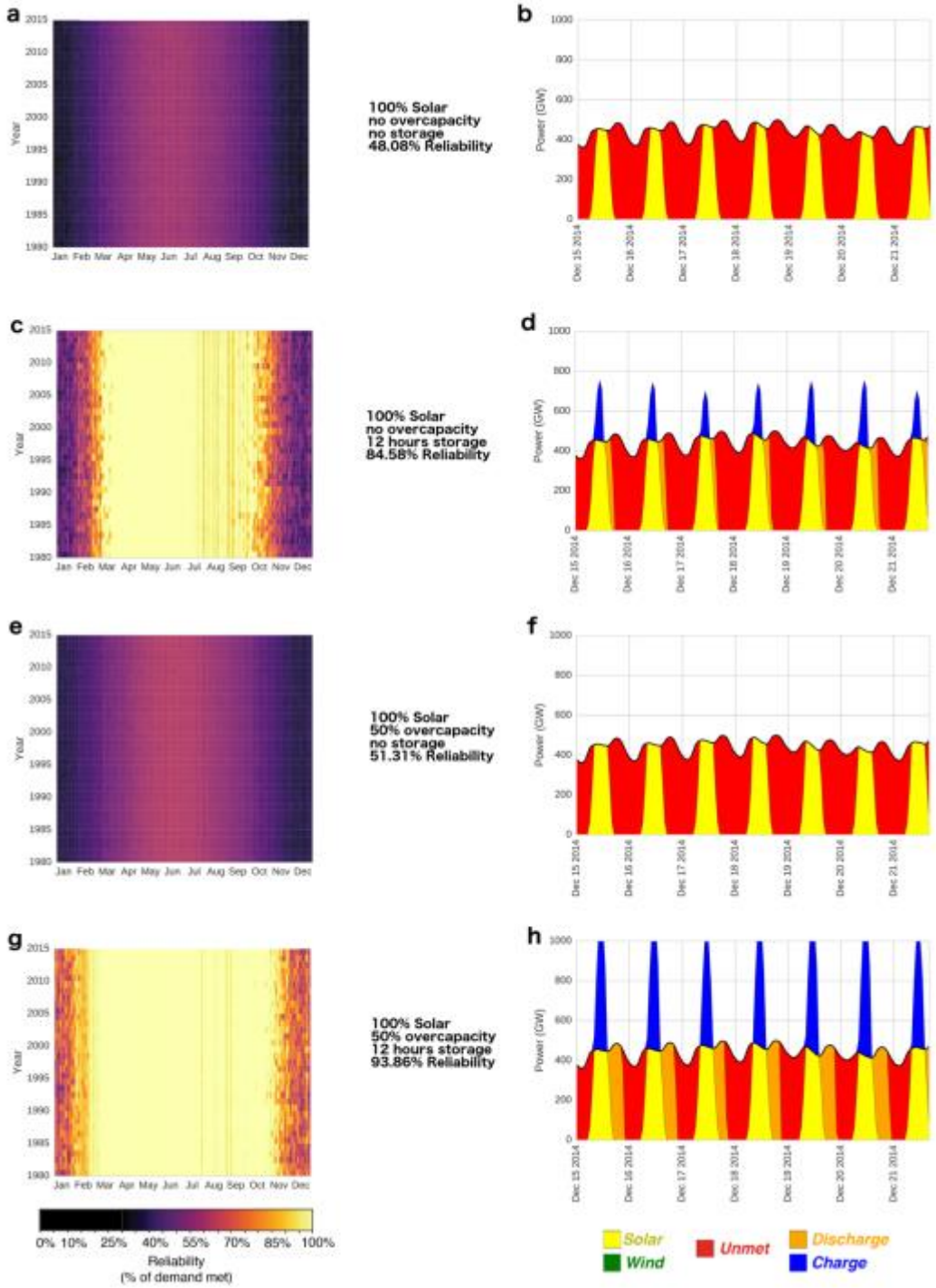

Figure S 2 | Daily demand met as a function of generation and energy storage capacity. Temporal characterization of the daily demand met by solar, wind and energy storage (if present) for every day in simulation period and aggregated over the contiguous U.S. under four different scenarios and with a resource mix of 100% solar: 1x generation and no energy storage (**a**, **b**); 1x generation and 12 hours of energy storage (**c**, **d**); 1.5x generation and no energy storage (**e**, **f**); 1.5x generation and 12 hours of energy storage (**g**, **h**). The right panel details the power profile over a weeklong period when meeting demand is particularly difficult.

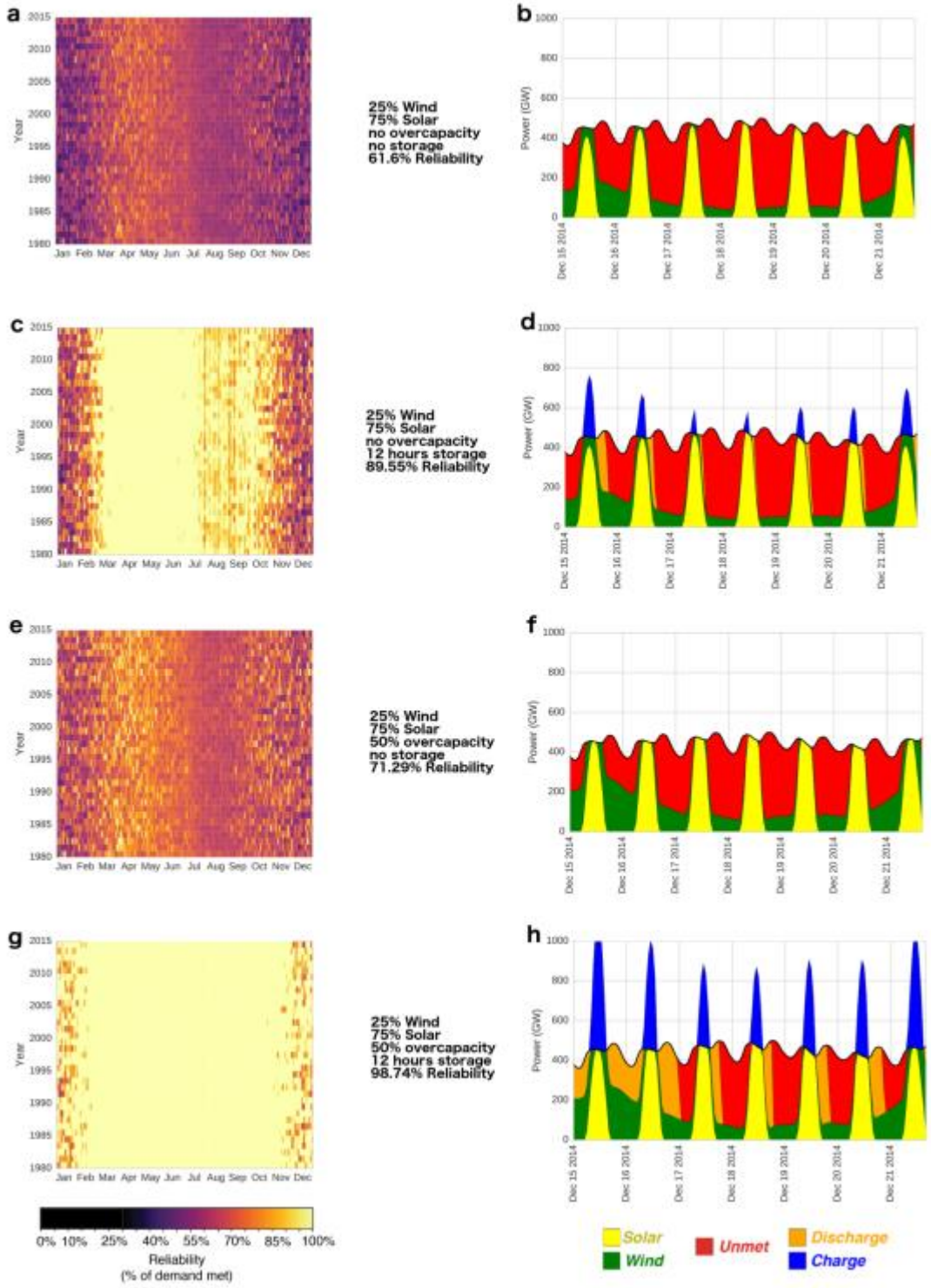

Figure S 3 | Daily demand met as a function of generation and energy storage capacity. Temporal characterization of the daily demand met by solar, wind and energy storage (if present) for every day in the simulation period and aggregated over the contiguous U.S. under four different scenarios and with a resource mix of 75% solar and 25% wind: 1x generation and no energy storage (**a**, **b**); 1x generation and 12 hours of energy storage (**c**, **d**); 1.5x generation and no energy storage (**e**, **f**); 1.5x generation and 12 hours of energy storage (**g**, **h**). The right panel details the power profile over a weeklong period when meeting demand is particularly difficult.

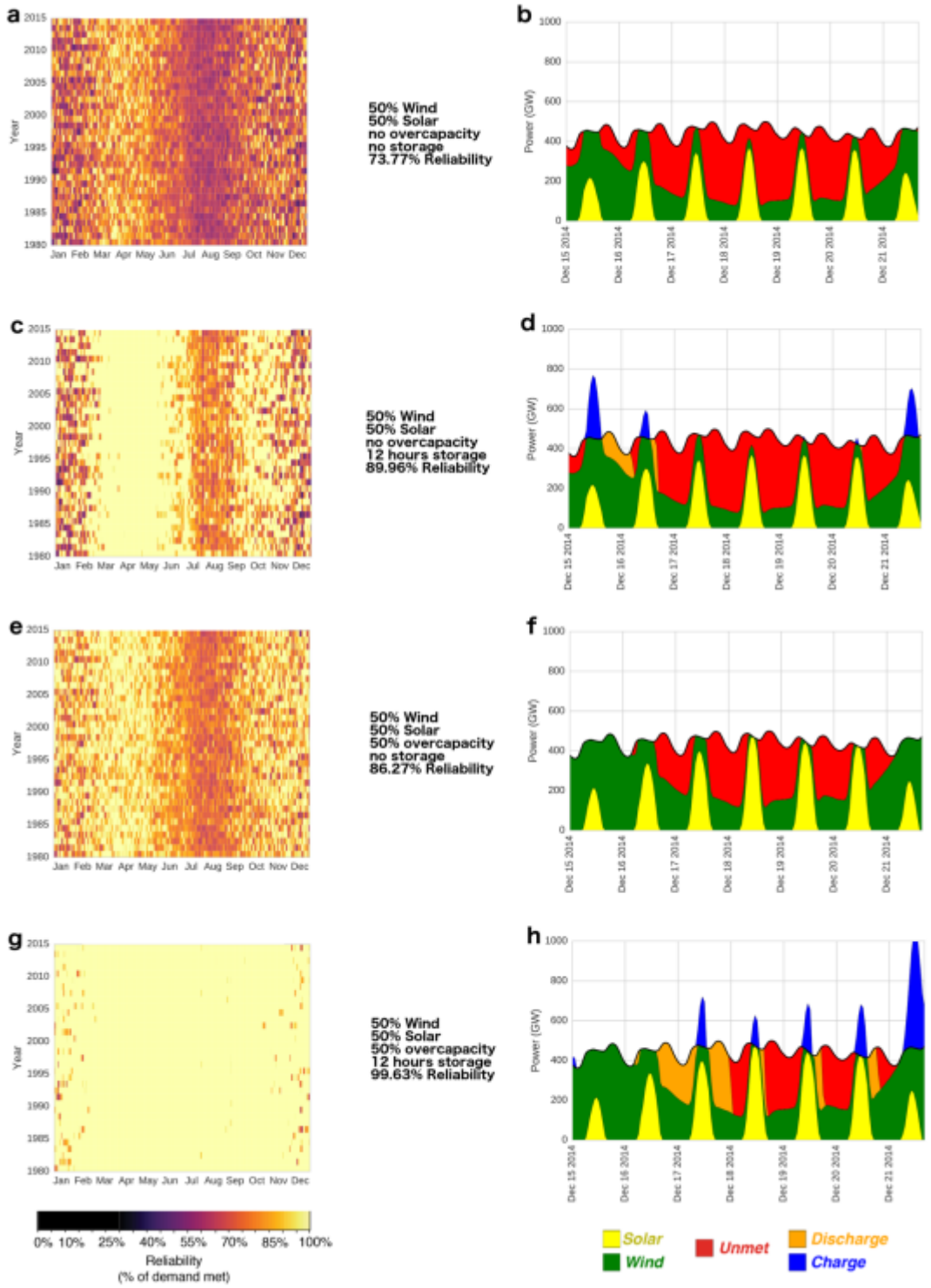

Figure S 4 | Daily demand met as a function of generation and energy storage capacity. Temporal characterization of the daily demand met by solar, wind and energy storage (if present) for every day in the simulation period and aggregated over the contiguous U.S. under four different scenarios and with a resource mix of 50% solar and 50% wind: 1x generation and no energy storage (**a**, **b**); 1x generation and 12 hours of energy storage (**c**, **d**); 1.5x generation and no energy storage (**e**, **f**); 1.5x generation and 12 hours of energy storage (**g**, **h**). The right panel details the power profile over a weeklong period when meeting demand is particularly difficult.

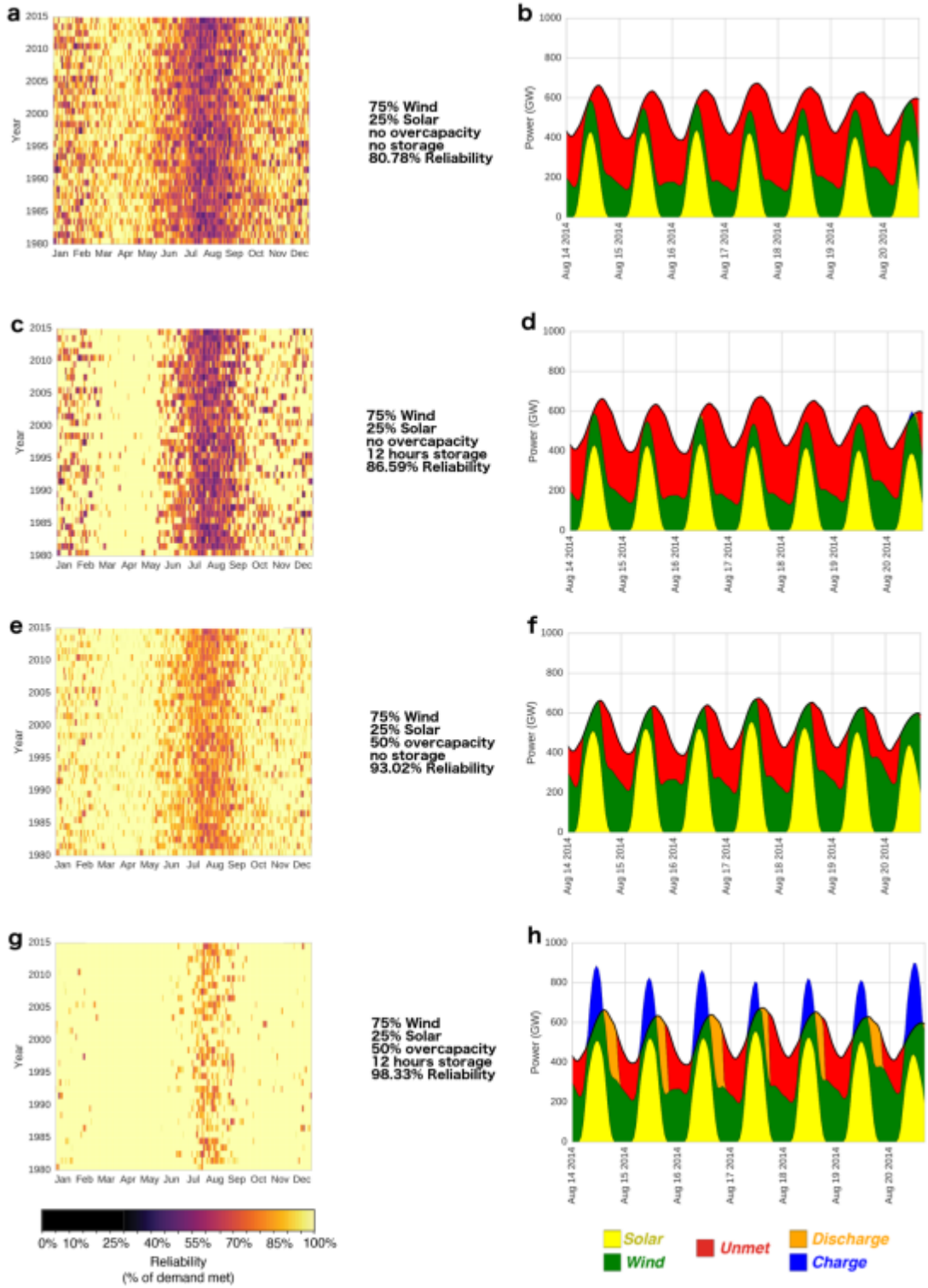

Figure S 5 | Daily demand met as a function of generation and energy storage capacity. Temporal characterization of the daily demand met by solar, wind and energy storage (if present) for every day in the simulation period and aggregated over the contiguous U.S. under four different scenarios and with a resource mix of 25% solar and 75% wind: 1x generation and no energy storage (**a**, **b**); 1x generation and 12 hours of energy storage (**c**, **d**); 1.5x generation and no energy storage (**e**, **f**); 1.5x generation and 12 hours of energy storage (**g**, **h**). The right panel details the power profile over a weeklong period when meeting demand is particularly difficult.

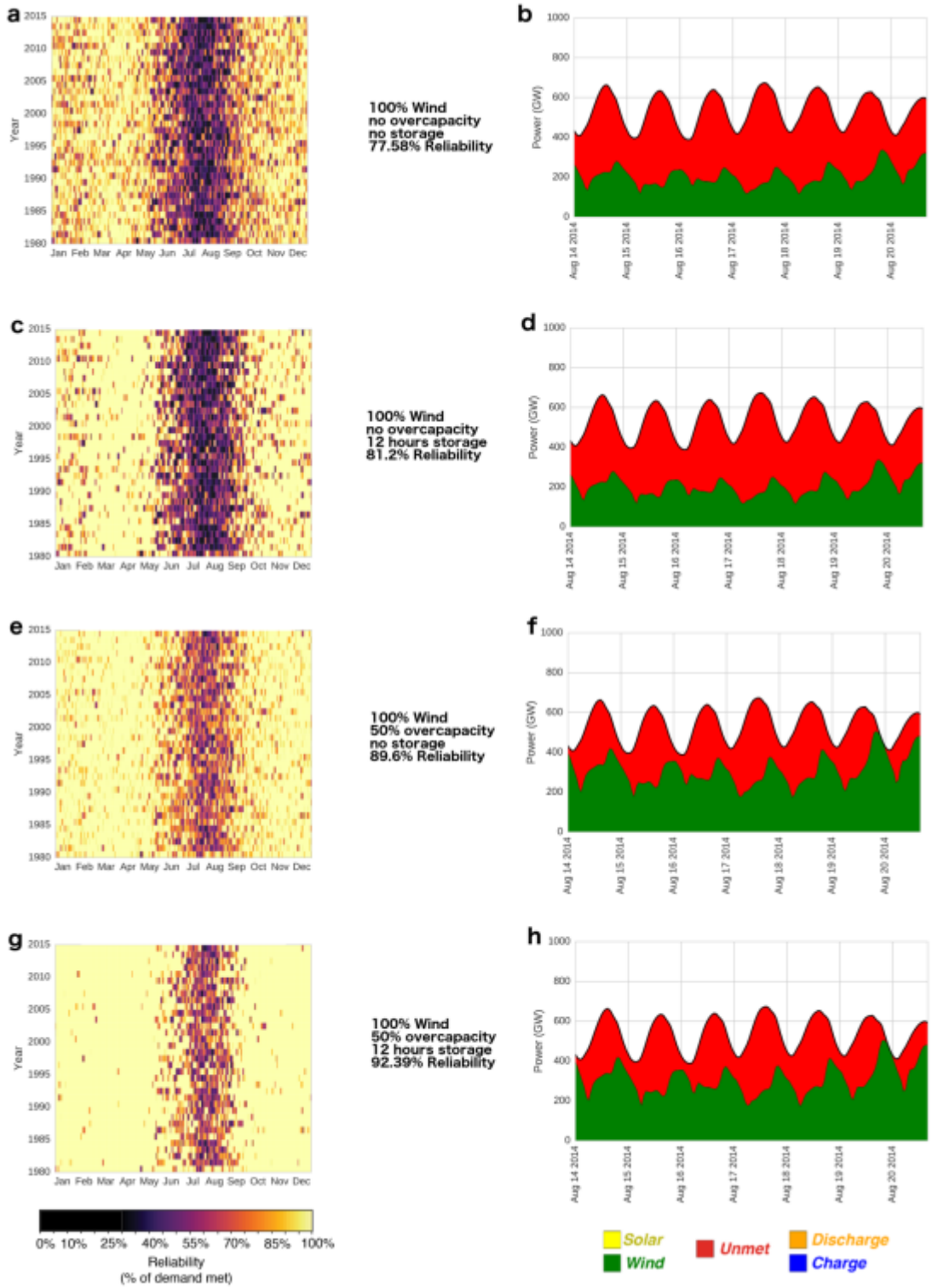

Figure S 6 | Daily demand met as a function of generation and energy storage capacity. Temporal characterization of the daily demand met by solar, wind and energy storage (if present) for every day in the simulation period and aggregated over the CONUS under four different scenarios and with a resource mix of 100% wind: 1x generation and no energy storage (**a**, **b**); 1x generation and 12 hours of energy storage (**c**, **d**); 1.5x generation and no energy storage (**e**, **f**); 1.5x generation and 12 hours of energy storage (**g**, **h**). The right panel details the power profile over a weeklong period when meeting demand is particularly difficult.

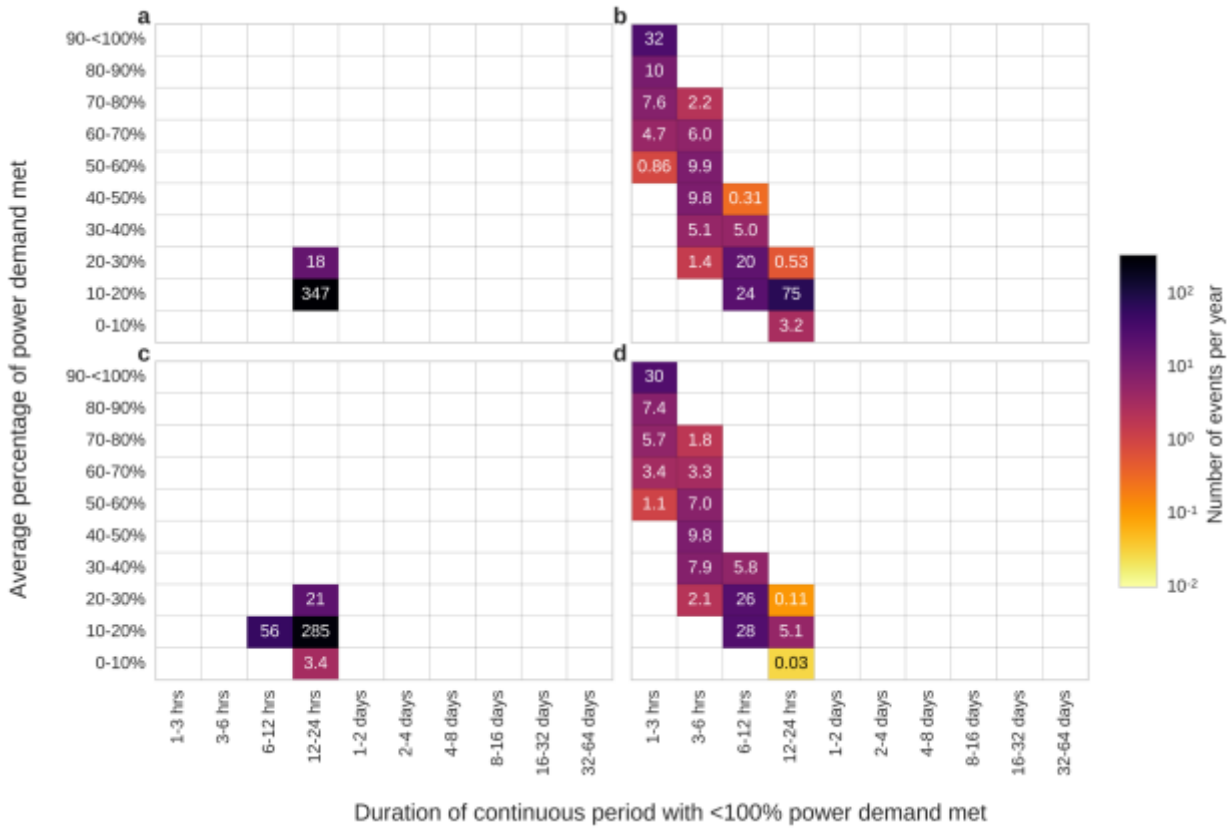

Figure S 7 | Heat maps of duration and size of continuous periods where < 100% of electricity demand is met. Four scenarios are shown for a resource mix of 100% solar: 1x generation and no energy storage (**a**); 1x generation and 12 hours of energy storage (**b**); 1.5x generation and no energy storage (**c**); 1.5x generation and 12 hours of energy storage (**d**).

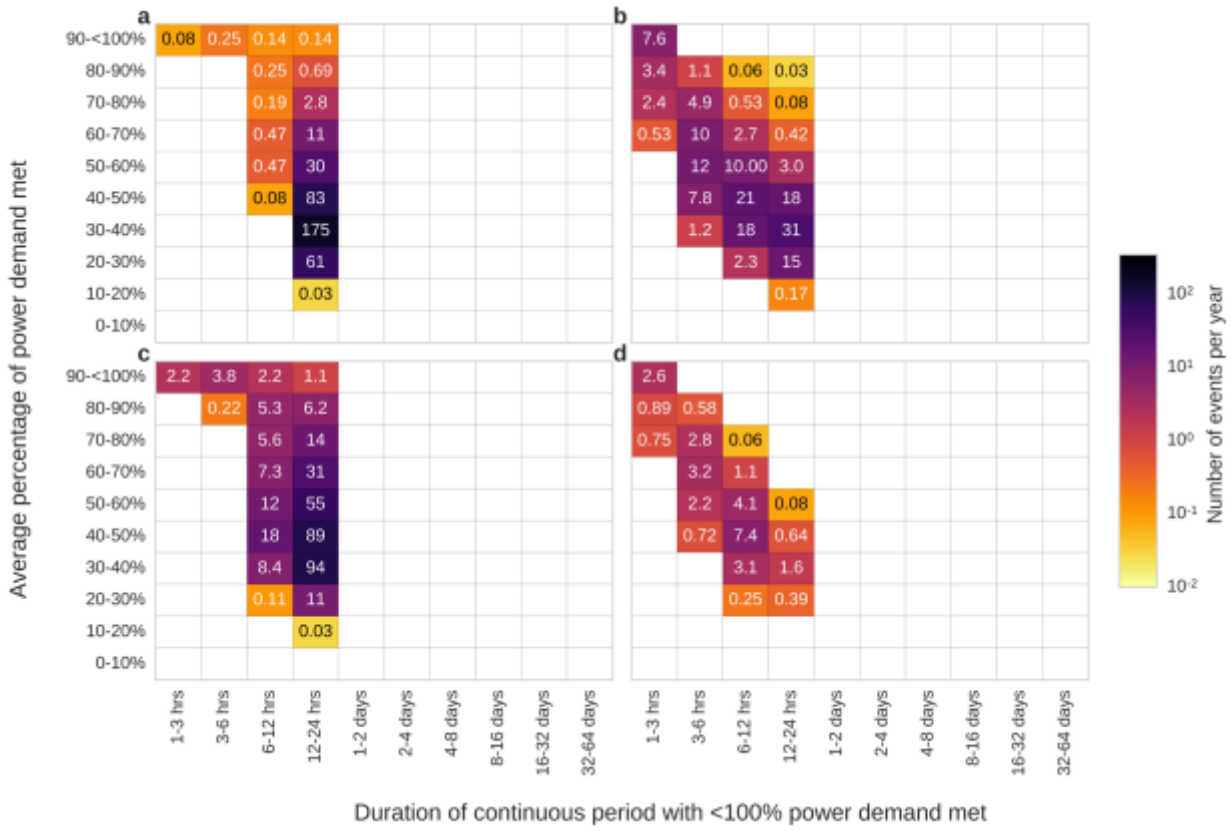

Figure S 8 | Heat maps of duration and size of continuous periods where < 100% of electricity demand is met. Four scenarios are shown for a resource mix of 75% solar, 25% wind: 1x generation and no energy storage (**a**); 1x generation and 12 hours of energy storage (**b**); 1.5x generation and no energy storage (**c**); 1.5x generation and 12 hours of energy storage (**d**).

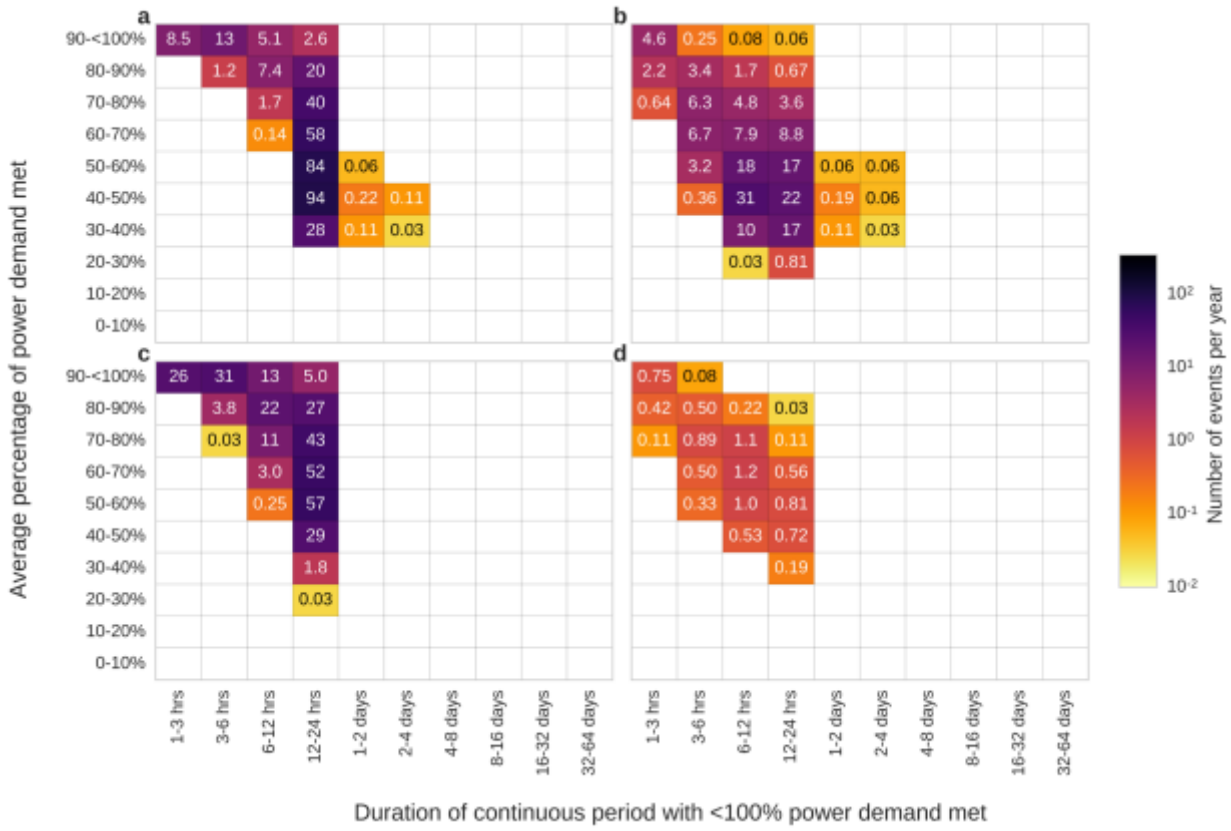

Figure S 9 | Heat maps of duration and size of continuous periods where < 100% of electricity demand is met. Four scenarios are shown for a resource mix of 50% solar, 50% wind: 1x generation and no energy storage (**a**); 1x generation and 12 hours of energy storage (**b**); 1.5x generation and no energy storage (**c**); 1.5x generation and 12 hours of energy storage (**d**).

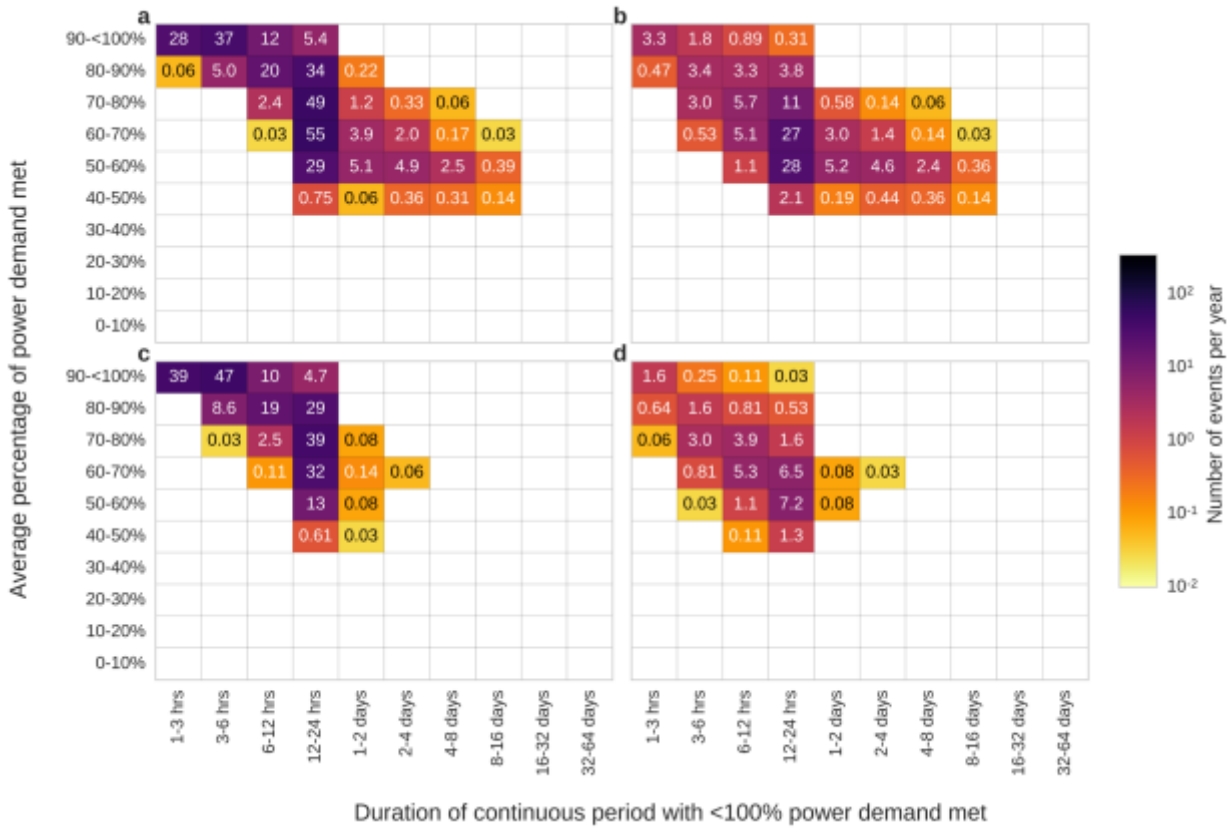

Figure S 10 | Heat maps of duration and size of continuous periods where < 100% of electricity demand is met. Four scenarios are shown for a resource mix of 25% solar, 75% wind: 1x generation and no energy storage (**a**); 1x generation and 12 hours of energy storage (**b**); 1.5x generation and no energy storage (**c**); 1.5x generation and 12 hours of energy storage (**d**).

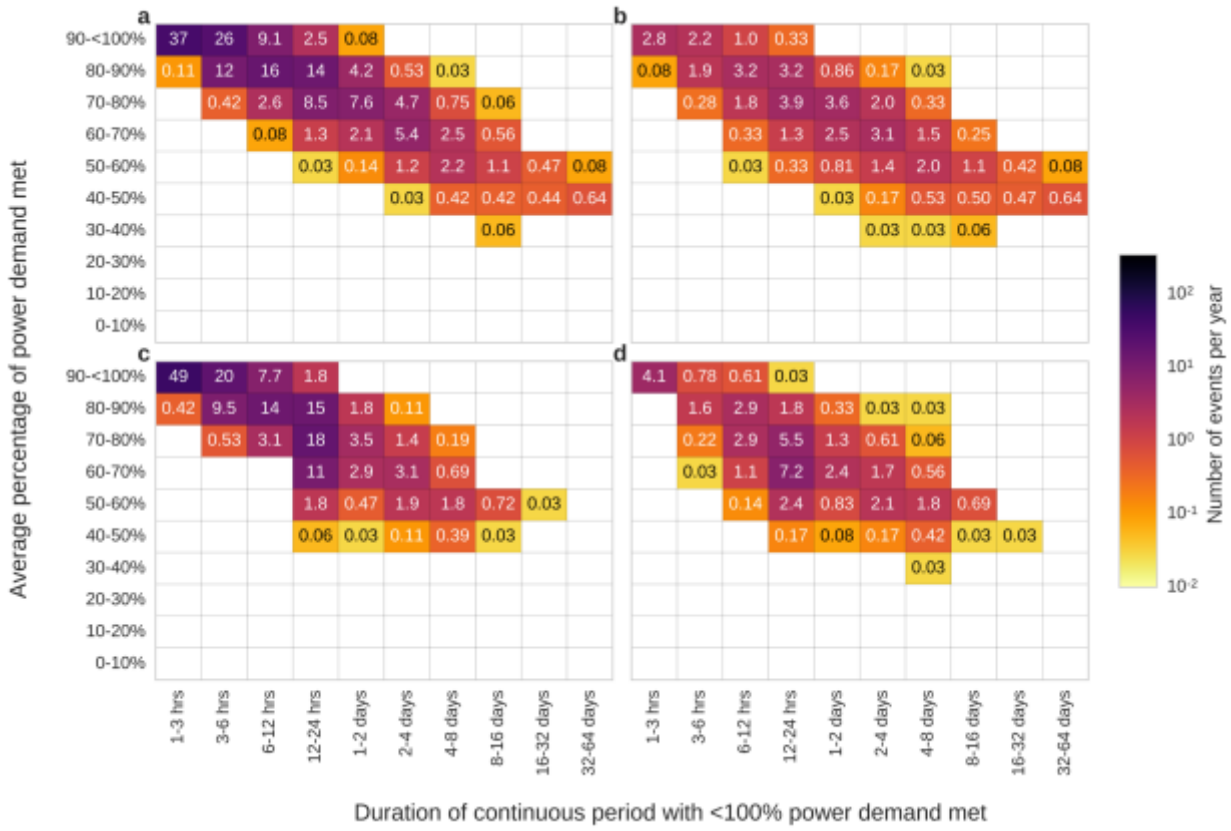

Figure S 11 | Heat maps of duration and size of continuous periods where < 100% of electricity demand is met. Four scenarios are shown for a resource mix of 100% wind: 1x generation and no energy storage (**a**); 1x generation and 12 hours of energy storage (**b**); 1.5x generation and no energy storage (**c**); 1.5x generation and 12 hours of energy storage (**d**).

Marginal Benefit of Additional Capacity

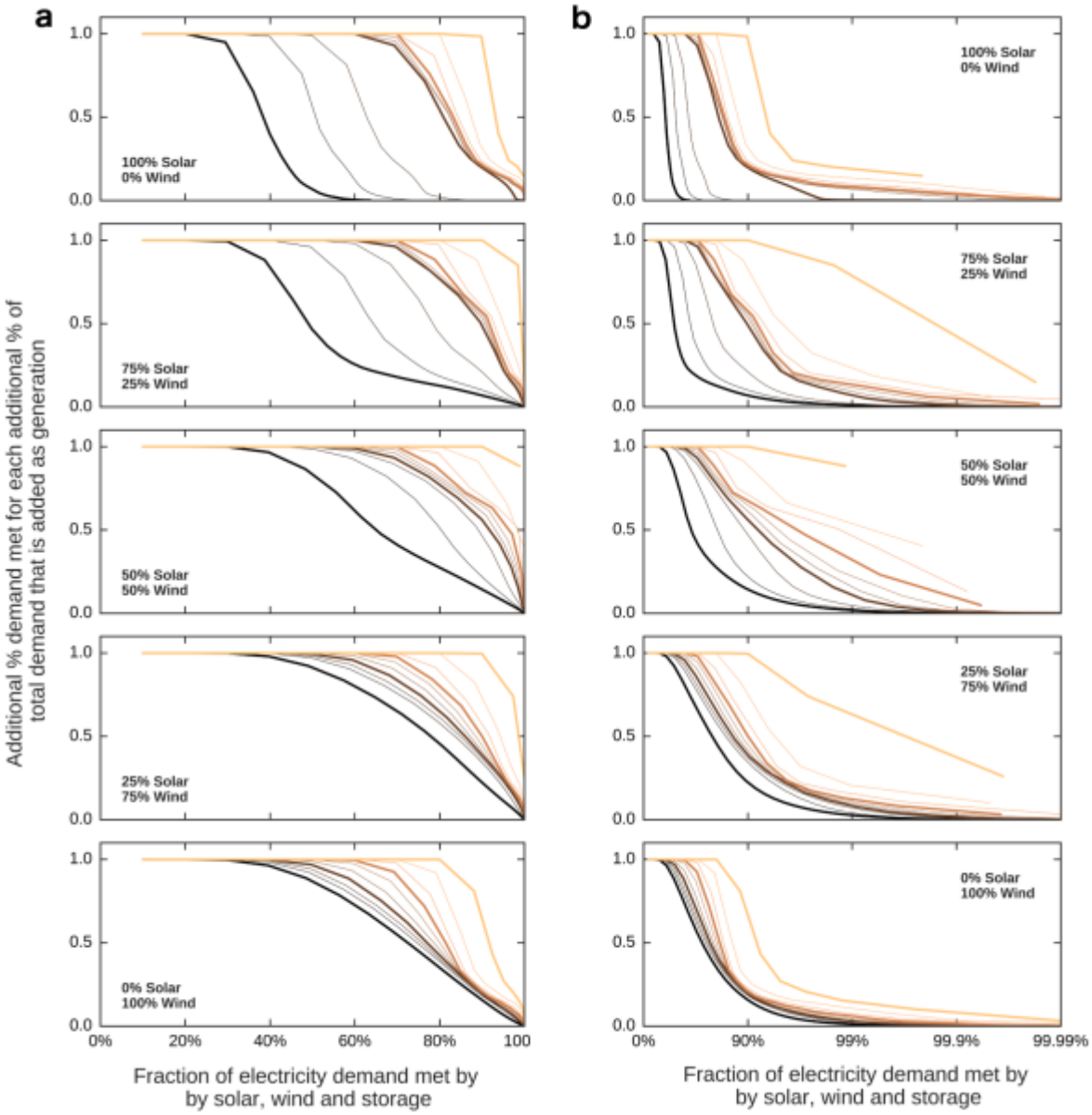

Figure S 12 | Marginal impact of additional generation as a function of energy storage capacity (0-24 hours), resource mix and reliability. Each line in each plot represents a different storage capacity. The storage capacities increase monotonically from bottom left to top right and represent 0, 3, 6, 12, 24 hours and 2, 4 ,8, 16, 32 days of mean demand storage. **a** (linear) and **b** (log) differ in the x-axis scaling of reliability with y-axis being shared for all plots. One way to interpret the y-scale is "if I installed solar and wind capacity such that they produce the equivalent of 1% of total demand, Y% will be used to meet demand".

Operational Characteristics of Storage

Discharge Rate

The discharge rate necessary for daily and seasonal variability is not expected a problem for batteries, but may be for technologies such as pumped hydro. Most batteries are rated to ~1C discharge or higher, which means they can discharge their full capacity in one single hour. If the battery capacity was 12 hours of mean demand, at most the batteries would discharge over ~6 hours (peak demand being < 2x mean demand). This is a rate of 1/6C which is far below the limit for most batteries. Pumped hydro facilities are limited by the power rating of the turbines and typically are built for full discharge over 10-12 hours. Higher powers and shorter discharge times could be obtained if designed for (higher power to energy ratios).

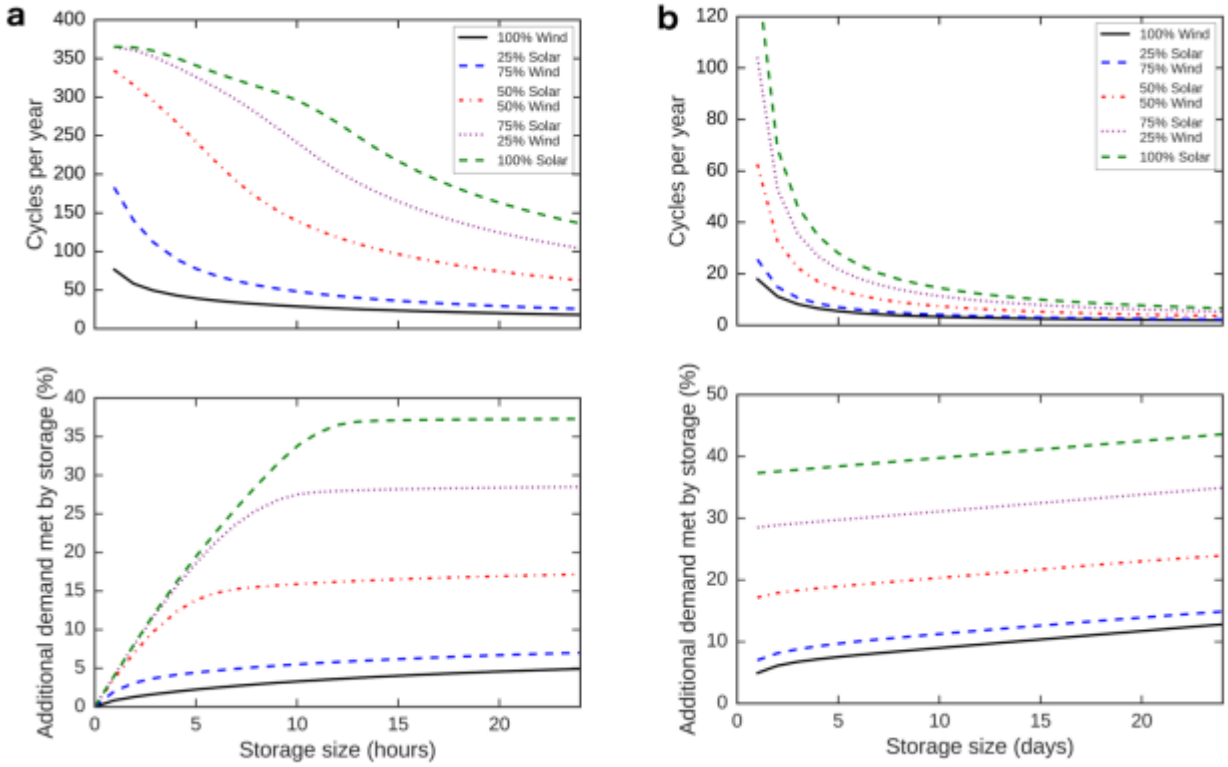

Figure S 13| Operational characteristics and reliability impact of energy storage. The number of cycles per year and increase in demand met as a function of storage size (**a** hours of storage, **b** days of storage) and resource mix. Storage provides the largest benefit, in terms of reliability increase, to 100% solar systems while providing less benefit as the wind proportion increases. The utilization (cycles per year) of storage decreases to very small numbers for days to weeks of storage and thus such a technical solution would have to be very inexpensive to compete.

Effect of Using 1 Year of Demand Data

It is possible that our results could be skewed by correlations of electricity demand with wind or solar generation. To test this possibility, we calculated the correlation coefficient between hourly wind and solar generation and hourly demand for the period of overlap (4 July to 31 Dec 2015). The fraction of variance explained by this correlation with hourly demand over this time period was 0.063 for wind and 0.275 for solar. If we take the other 35 half-years (from 4 July to 31 Dec of each year from 1980 to 2014), the mean and standard deviation of the R2 of the correlations was 0.067 ± 0.028 for wind and 0.275 ± 0.006 for solar. Thus, there is no evidence that correlations found in 2015 but not in other years would substantially affect our results.

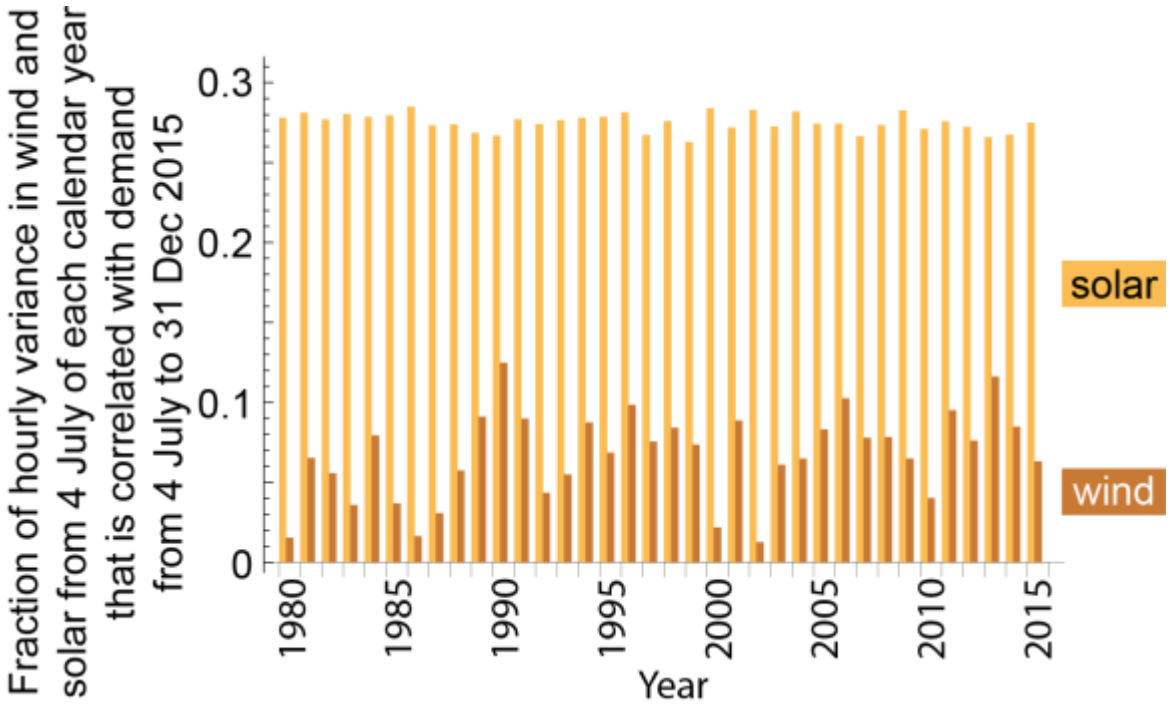

Figure S 14| R2 of correlations between hourly wind and demand, and solar and demand, data. In all cases, demand data is from 4 July to 31 December 2015. Correlations are calculated for wind and solar generation data for 4 July to 31 December of each listed year. Note that year 2015 is not an outlier, indicating that the differences in correlations coming from cycling of demand data are unlikely to substantially affect our results.

Corrections Made to Demand Data

- Adjustments made to hourly data from 7/1/2015 - 12/31/2015 dataset
  - When looking at the plot of the summed hourly data for the entire US some hours are clearly incorrect
  - A catalog of the changes made to the raw data are below
    - IPCO, August 24, 2015 @ 12am - raw value is 547897, this is three orders of magnitude larger than surround days at the same time (12am). This value as been changed to 2533 which is the value from the previous day (August 23, 2015). This number is in the noise of the total demand for the US so using the previous day's data should be ok.
    - IPCO, September (9) 28, 2015 @ 10pm (hour 22) - raw value is 1201314. Set to previous days value of 1817. This number is in the noise of the total demand for the US so using the previous day's data should be ok.

- PACE, December 29, 2015 @ 10pm - raw value was -12958873, replaced with previous days value at same time, 6081
- PACE, December 29, 2015 @ 11pm - raw value was -31990166, replaced with previous days value at same time, 5472
- IPCO, December 14, 2015 @ 11pm - raw value was -1096258, replaced with previous days value at same time, 1828
- IPCO, August 23, 2015 @ 9pm (hour 21) - raw value was -542965, replaced with previous days value at same time, 2378
- Errors addressed from Known Issues document
  - BPAT didn't report demand for 7/1/2016 (all 24 hours)
    - Copied data from 7/2 into those 24 hours
  - BPAT didn't report demand for 10/13/2016 (8am - midnight)
    - Copied data from 10/12/2016 (previous day)
  - PJM 11/1/2015 not reliable because of daylight savings time change
    - 11/1/2015 4am UTC was repeated twice and 11/2/2015 UTC hours 4am and 5am were missing. UTC time should be consistent and not change…it is the local time that should change. So I removed the extra 4am slot in the UTC column and added in the 4am and 5am hours for 11/2/2015. I then added in an extra 2am in the local time on 11/1/2015 as this is the appropriate time shift. No changes to the data were made, except for the addition of data to the now blank 11/2/15 5am (UTC) time slot. I used the previous days value of 67190 from 11/1/2015 5am (UTC) which was an hour later in local time.
  - CISO
    - 10/4/2015 no demand values from 7pm UTC to 2am 10/5/15 UTC. Used demand forecast for those periods as actual demand
    - 10/31/15 and 11/1/15 - duplicates in UTC times existed at 10pm, 10am, 1pm and 3pm and then it skipped 2 hours. Changed values such that time steps were 1 hour everywhere
    - 11/16/2015 3pm-4pm no demand values so used values from demand forecast
    - 11/18/2015 1am-2am UTC no demand values so used demand forecast values
    - 11/1/15 missing 2 hours worth of values. I don't know what to do here because the UTC time is duplicated at 9am but the local time isn't duplicated for daylight savings time change (Fall back). I think the local time is the one that should have a duplicate at 2am and the UTC time should be continuous without a duplicate. This would require shifting all the local times even those from 2016…
  - DOPD
    - Corrected UTC time mistakes on 11/1/2015. Many duplicate and associated missing hours.
  - TEPC
    - Didn't report demand values for 8/5/2015, replaced with forecast demand values
    - Didn't report demand values for 10/9/15, replaced with forecast demand values
  - PACE
    - 10/14/2015 4pm-5pm UTC, high demand values, replaced with forecast demand

        - 11/2/2015 3pm, 4pm, 7pm UTC, high demand values, replaced with forecast demand
        - 11/17/2015 6pm UTC, high demand value, replaced with forecast demand
        - 11/30/2015 11am, 1pm UTC, negative demand values, 12pm, 3pm, 6pm high demand values, replaced all with forecast demand
        - 11/22/2015 6-7pm UTC, high demand values, replaced with forecast demand
    - PACW
        - 10/30/2015 8pm negative demand value, 9pm high demand value, replaced all with forecast demand value
        - 11/3/2015 4pm negative demand value, 5pm high demand value, replaced all with forecast demand value
        - 12/8/2015 4am UTC, high demand value, replaced with forecast demand value
        - 12/9/2015 2am UTC, high demand value, replaced with forecast demand value
    - DUK
        - 11/1 and 11/2, adjusted UTC times to have 1 hour increments. 1 hour missing, 5am on 11/2
    - CPLE
        - Adjusted UTC times around on 11/1 and 11/2. 1 hour missing, 5am 11/2
    - CPLW
        - Adjusted UTC times around on 11/1 and 11/2. 1 hour missing, 6pm 11/1
    - FPC
        - Adjusted UTC times around on 11/1 and 11/2. 1 hour missing, 5am 11/2
    - TEC
        - Adjusted UTC times around on 11/1 and 11/2. 1 hour missing, 5am 11/2
    - LAWP
        - Adjusted UTC times around on 11/1 and 11/2. 1 hour missing, 6am 11/2
        - Missing 11am 11/1/15 to 7am 11/2/15, replaced with next days values for same hours…no forecast demand for these hours
    - AECI
        - Missing demand on 11/19/15 from 2am - 6am UTC. Replaced with demand forecast values
    - GVL
        - Missing demand on 12/31 all day (local time). Replaced with demand forecast values
        - Missing demand value on 11/2 at 11am utc, replaced with demand forecast value
    - ISNE
        - Missing demand values on 11/10 @ 1am-2am UTC. Replaced with demand forecast values
    - NYIS
        - Missing demand value on 11/4 @ 4pm UTC. Replaced with demand forecast value
    - PACE
        - 8/12/15 @ 7pm, large demand value, replaced with forecast demand value

- Adjustments made to hourly data from 1/1/2016 - 7/1/2016 dataset
    - A catalog of the changes made to the raw data are below

- IPCO, January 6, 2016 @ 8pm (hour 20) - raw value is 727601. Set to previous days value of 1899. This number is in the noise of the total demand for the US so using the previous day's data should be ok.
- IPCO, January 7, 2016 @ 5pm (hour 17) - raw value is 727875. Set to previous days value of 1728. This number is in the noise of the total demand for the US so using the previous day's data should be ok.
- PACW, January 11, 2016 @ 12am (hour 0) - raw value is 2530003. Set to previous days value of 2625.
- AZPS, February 25, 2016 @ 6am (hour 6) - raw value is 1671653. Set to previous days value of 2870.
- PACW, March 2, 2016 @ 9pm (hour 21) - raw value is 3104681. Set to previous days value of 2489.
- PACW, March 2, 2016 @ 10pm (hour 22) - raw value is 3104614. Set to previous days value of 2342.
- IPCO, March 17, 2016 @ 10pm (hour 22) - raw value is 8145294. Set to previous days value of 1663.
- PACE, April 22, 2016 @ 1am (hour 1) - raw value is 168005. Set to previous days value of 5064.
- SC, May 14, 2016 @ 4am (hour 4) - raw value is 9528282. Set to previous days value of 2536.
- PACE, February 2, 2016 @ 6pm (hour 18) - raw value is -243534. Set to previous days value of 5907.
- PJM, March 13, 2016 @ 6am (hour 6) - no raw value existed because of daylight savings change over. Took previous and next days values from 5am, 6am and 7am. Found ratio between 5am and 6am and 6am and 7am values. The ratio of the differences (5am to 6am / 6am to 7am) was ~1.26. I then used this ratio to find the value using the 5am and 7am values on March 13, 2016. The value input was 68541.
- PJM, March 13, 2016 @ 8am (hour 8) - no raw value due to daylight savings times. Used value from 9am on March 13 for this value. Looking at values from days after this the values at 8am and 9am were pretty close so this assumption seemed ok. Value entered was 65208.
- IPCO, March 17, 2016 @ 9pm (hour 21) - raw value is -8142034. Set to previous days value of 1676.
- PACE, April 2, 2016 @ 10pm (hour 22) - raw value is -176663. Set to previous days value of 5136.
- PACE, April 22, 2016 @ 12am (hour 0) - raw value is -97995. Set to previous days value of 5159.
- PACW, April 22, 2016 @ 12am (hour 0) - raw value is -80487. Set to previous days value of 2417.
- IPCO, May 18, 2016 @ 10pm (hour 22) - raw value is -851411. Set to previous days value of 1596.
- SC, June 12, 2016 @ 4am (hour 4) - raw value is -3747482. Set to previous days value of 2970.

Flow Diagram Algorithm of Simulation

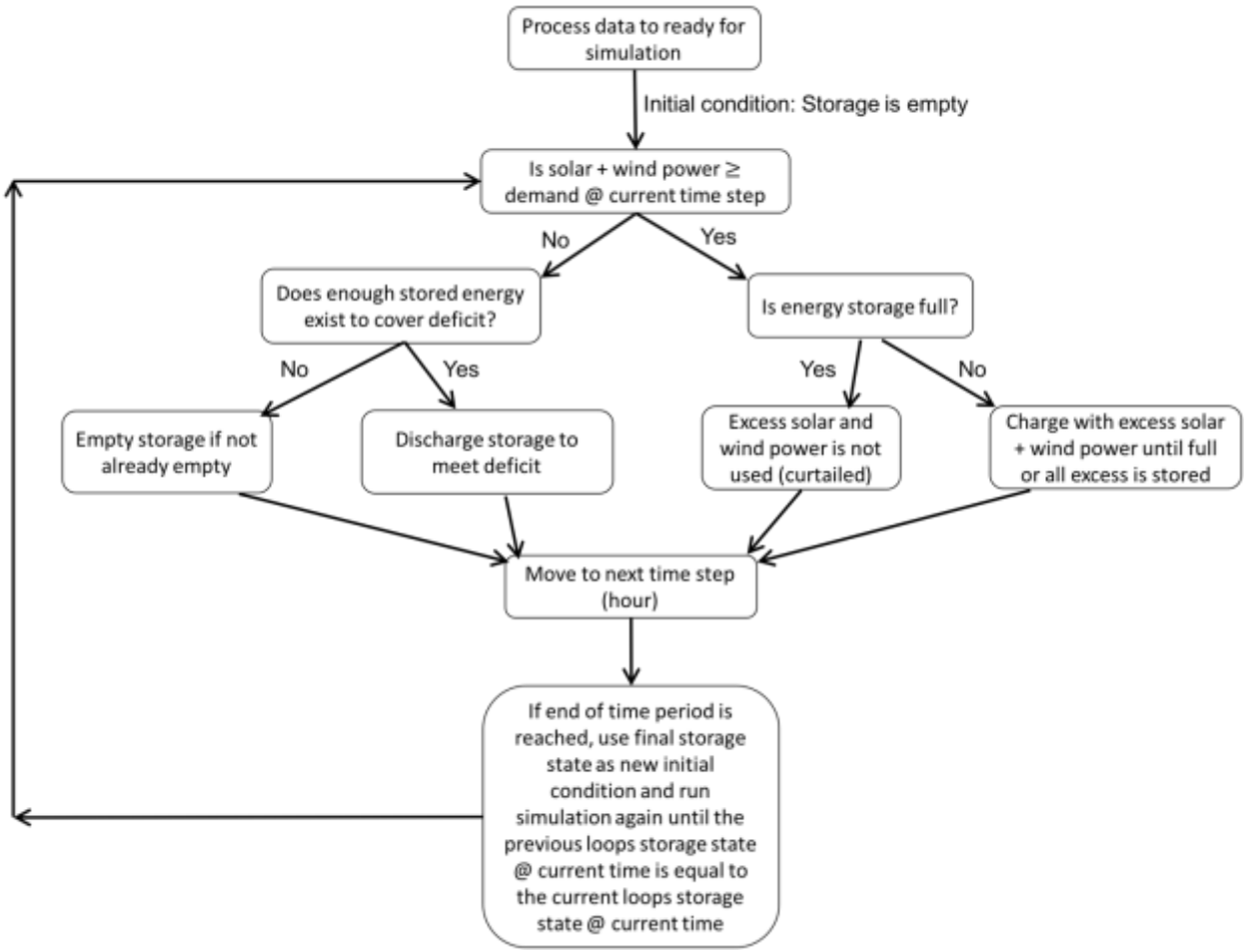

Figure S15| Flow diagram of simulation code

Python Code

Example Script to Run Core Simulation for Many Different Scenarios

```
import general_reliability_functions_real_demand as grfrd
import numpy as np

def main():

        # battery_size = np.array([0., round(1/3.,2), 1., 7., 30.]) # storage size in days of demand, later converted to hours, etc.
        battery_size = np.arange(1,31)

        SF = np.arange(0,1.01,0.05) # solar fraction of energy production
        demand = np.load('/lustre/data/mshaner/merra2/EIA_demand_files/conus_real_demand.npy') * 10**6 # W
        overbuild = [np.arange(0.1,5.1,0.1), np.arange(5.0, 30.1, 0.5)]
        overbuild = np.round(np.array([j for list in overbuild for j in list]), 2)
```

```
        path = '/lustre/data/mshaner/merra2/country_files' # file system common path
        country_folder = 'USA' # file system folder name for the country of interest
        country = 'United States of America'
        filename_start = '{}_area-weighted-mean_real-demand'.format(country) # start to the file
names

        # import solar and wind data
        CF_solar = np.load('{}/{}/{}_CFsolar_area-weighted-mean.npy'.format(path,
country_folder, country))

        CF_wind = np.load('{}/{}/{}_CFwind_area-weighted-mean.npy'.format(path,
country_folder, country))

        # run battery cycling for all permutations of cases of battery size, SF and overbuild

        total_hours = CF_wind.size # calculate total number of hours.  This should be identical
for wind and solar

        for ob in overbuild:

                pwr_avg = demand.mean() * ob # W, calculate the mean power production
                # based on the mean demand and overbuild

                # calculate wind (wc) and solar (sc) capacities and save
                wc, sc = grfrd.capacity_calc(CF_wind, CF_solar, SF, pwr_avg, total_hours)

                np.save('{}/{}/{}_wind_capacity_overbuild-{}'.format(path, country_folder,
                        filename_start, ob), wc)
                np.save('{}/{}/{}_solar_capacity_overbuild-{}'.format(path, country_folder,
                        filename_start, ob), sc)

                for bs in range(battery_size.size):

                        batsize = battery_size[bs] * demand.mean() * 24 # conversion to Wh

                        # run the system for one set of parameters and save the data
                        # br is binary reliability which counts yes or no whether demand was met
that hour
                        # brs is the binary reliability for all time; it is a single value that
represents all hours
                        # pr is power reliability which counts the amount of power unment for
each hour
                        # prs is power reliability for all time; it sums all power unmet and divides
by total power demanded
                        # bat_state is the hourly state of the battery
                        br, brs, pr, prs, bat_state = grfrd.reliability_calc(CF_wind, wc, CF_solar,
```

```
sc,
                                demand, batsize, SF, total_hours)

                        np.save('{}/{}/{}_binary_reliability_SF_batsize-{}-days_overbuild-
{}'.format(path,
                                country_folder, filename_start, battery_size[bs], ob), brs)
                        np.save('{}/{}/{}_hourly_binary_reliability_SF_batsize-{}-
days_overbuild-{}'.format(path,
                                country_folder, filename_start, battery_size[bs], ob), br)
                        np.save('{}/{}/{}_power_reliability_SF_batsize-{}-days_overbuild-
{}'.format(path,
                                country_folder, filename_start, battery_size[bs], ob), prs)
                        np.save('{}/{}/{}_hourly_power_reliability_SF_batsize-{}-
days_overbuild-{}'.format(path,
                                country_folder, filename_start, battery_size[bs], ob), pr)
                        np.save('{}/{}/{}_battery_state_SF_batsize-{}-days_overbuild-
{}'.format(path,
                                country_folder, filename_start, battery_size[bs], ob), bat_state)

        np.save('{}/{}/{}_battery_size'.format(path, country_folder, filename_start),
battery_size)


if __name__ == '__main__':
        main()
```

Core Simulation

```
from numba import jit
import numpy as np
import pandas as pd


#common function that calculates wind capacity factor given windspeeds
@jit
def windcf_calc(wind_speed): # calculate wind capacity factor

        u_ci = 3 # cut in speed in m/s
        u_r = 15 # rated speed in m/s
        u_co = 25 # cut out speed in m/s


        CF_wind = np.zeros(wind_speed.shape) # initialize array
        CF_wind[wind_speed < u_ci] = 0 # for all wind speed values less than cut in speed no
power is produced (CF=0)
```

```
        CF_wind[(u_ci <= wind_speed) & (wind_speed < u_r)] = wind_speed[(u_ci <=
wind_speed) & (wind_speed < u_r)]**3/u_r**3 # for all wind speed values between cut in and
rated speeds the CF goes by u cubed
        CF_wind[(u_r <= wind_speed) & (wind_speed <= u_co)] = 1 # for all wind speed values
above the rated speed CF = 1

        return CF_wind

# common function that calculates the wind and solar capacities
def capacity_calc(CF_wind, CF_solar, SF, pwr_avg, hours_in_period):

        wind_capacity = np.zeros(SF.size)
        solar_capacity = np.zeros(SF.size)

        counter = 0
        for sf in SF:
                wind_capacity[counter] = ((1-sf) * pwr_avg * hours_in_period) / CF_wind.sum()
                solar_capacity[counter] = sf * pwr_avg * hours_in_period / CF_solar.sum()

                counter += 1

        return wind_capacity, solar_capacity


# common function that takes in CF_wind, CF_solar data and user specified values and
# calculates the capacities and reliabilities for each case
def reliability_calc(CF_wind, wind_capacity, CF_solar, solar_capacity, demand, batsize, SF,
hours_in_period):

        solarfrac = range(SF.size)

        # initialize arrays for collection reliabilities
        binary_reliability = np.zeros((SF.size, hours_in_period)) # will collect hours unable to
met demand at each time step
        binary_reliability_sum = np.zeros(SF.size) # holder for biniary reliability over all time
for a given SF
        power_reliability = np.zeros((SF.size, hours_in_period)) # will collect amount of power
unable to be delivered at each time step
        power_reliability_sum = np.zeros(SF.size) # holder for reliability of power for all time
for a given SF
        battery_state = np.zeros((SF.size, hours_in_period + 1)) # initialize array; need +1
because first entry is initial state

        for sf in solarfrac: # loop through all solar fractions

                # supply - demand at each time step
```

```
                power = (np.squeeze(solar_capacity[sf]) * np.squeeze(CF_solar) + \
                        np.squeeze(wind_capacity[sf]) * np.squeeze(CF_wind)) - demand

                for time in range(0,hours_in_period): # loop through all times

                        bs_next_state = battery_state[sf, time] + power[time] # starting point for
next battery state; will be bounded by empty and full battery below

                        #discharging
                        if power[time] < 0:

                                if bs_next_state < 0: # if some power demand cannot be met, i.e.
energy stored in battery isn't enough
                                        binary_reliability[sf, time] = 1 # record that hour as
unreliabile
                                        power_reliability[sf, time] = np.absolute(bs_next_state) #
add amount of power that cannot be delivered to running sum
                                        # next battery state is 0, which is the default

                                else: # if enough energy left in battery to meet demand
                                        battery_state[sf, time+1] = bs_next_state # next battery
state is current state minus discharged amount

                        #charging
                        else:

                                battery_state[sf, time+1] = min(bs_next_state, batsize) # either
something less than full or full

                # now go back through battery state using last battery state to reloop through
years until the values don't change...stable solution reached
                time = 0
                num_iters = 0
                for anything in range(0,hours_in_period * 10): # the potential to loop through the
battery state many times

                        # calculate the potential next battery state
                        if time != 0:
                                possible_next_bs_state = battery_state[sf, time] + power[time]

                        else: # use end state of battery as initial state instead of original initial
state, this will be true least often so put as else statement
                                possible_next_bs_state = battery_state[sf, -1] + power[time]

                        # determine what poential next battery state is given battery size bounds of
full or empty
```

```
                if power[time] < 0:

                        if possible_next_bs_state < 0: # if some power demand cannot be
met, i.e. energy stored in battery isn't enough
                                binary_reliability[sf, time] = 1 # record that hour as
unreliabile
                                power_reliability[sf, time] =
np.absolute(possible_next_bs_state) # add amount of power that cannot be delivered to running
sum
                        else:
                                binary_reliability[sf, time] = 0 # record that hour as
reliable...may have been unreliable in previous run
                                power_reliability[sf, time] = 0 # record that hour as
reliable...may have been unreliable in previous run


                        possible_next_bs_state = max(possible_next_bs_state, 0)

                else:
                        possible_next_bs_state = min(possible_next_bs_state, batsize)

                # if steady solution reached break out of loop, otherwise give battery state
new value and go next time step
                if battery_state[sf, time+1] == possible_next_bs_state:
                        break

                else:
                        battery_state[sf, time+1] = possible_next_bs_state
                        time += 1

                        if time >= hours_in_period: # if end of battery state reached, start
over
                                time = 0

                num_iters += 1 #count number of iterations just to see how far this has to
go before steady solution is achieved

        # calculate fraction of power able to be delivered since recorded values are
power unable to be delivered
        binary_reliability_sum = 1 - binary_reliability.sum(axis=1) / hours_in_period
        power_reliability_sum = 1 - power_reliability.sum(axis=1) / (demand.sum())

        # return results
        return binary_reliability, binary_reliability_sum, power_reliability,
power_reliability_sum, battery_state
```