USENSYS - Electric Power Sector

https://github.com/olugovoy/usensys

Contents

Overview 1

Map data 2

Sub-annual time resolution (time-slices) 4
Commodities . . . 5
Demand . . . 5

Weather factors 10
Solar availability factors . . . 11
Wind availability factors . . . 16
Weather classes in the model . . . 24

Supply 27

Power generating technologies 28

Storage technologies 32

Interregional ELC electrical grid (simplified) 32

Interregional UHV electrical grid 39

Inverter and rectifier stations 42

Trade with the rest of the world (ROW) 44

Model #1 45

Model #2 47

Check the results 49

Shape of supply 52

Capacity maps 55

Trade flows maps 57

Overview

This is the first draft of electric power sector, the renewables balancing version. The output of the optimization is capacity of renewables, storages, and interregional UHVDC grid with inverter and rectifier station capacity. The main purpose of this stage is testing functionality of the framework, and sketching further steps of the model development.

The current features of the model:
* 49 regions (48 lower states and Disctict of Columbia);

* 1 year, 8760 hours (24x365);
* actual demand of electricity in 2017 (monthly date by states, desagregated by hours using national load curve - will be improved on further steps);
* Main technologies: - solar and wind power farms,
- electricity storage (intraday, and interday),
- endogenous interregional grid (UHV lines and coverter stations),
* MERRA-2 weather data (see bellow the aggregation procedure);

Technologies and features in progress (to be added on the next steps):
- static demand with fixed location,
- static demand with optimized location,
- demand-side management techs with fixed location,
- demand-side management techs with optimized location - …

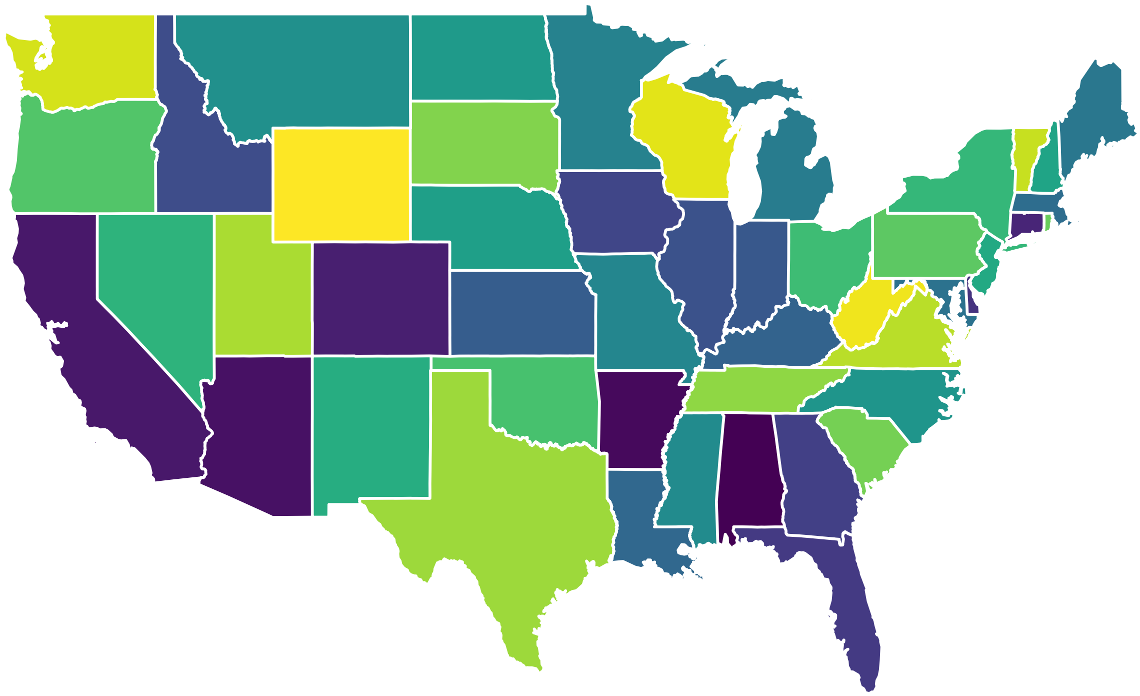
The model have been solved with GAMS/CPLEX. Time: >= 3 hours on 6-core, 5GHz Intel processor. The main hardware requirements is RAM >= 64Gb.

The results of the optimization are saved in " files.

Map data

GIS information will be used for graphical output and calibration of some parameters (such as estimation of distances between regions for required electric power grid). Map objects: **usa49reg** - lower 48 states + DC, spatial polygon data frame;
usa49r - lower 48 states + DC, data frame format.

```r
if (!file.exists("data/maps/usa49reg.RData")) {
  stop("US map data is not found. Follow the steps in 'usa_maps.R'")
} else {
  load("data/maps/usa49reg.RData")
}

ggplot(usa49r, aes(long,lat, group = group, fill = id)) +
  geom_polygon(aes(fill = id), colour = rgb(1,1,1,.2)) +
  # geom_polygon(aes(fill = id), colour = "white", size = .75) +
  labs(fill = "Region") +
  coord_fixed(1.45) +
  # scale_fill_brewer(palette = "Paired") +
  #coord_quickmap() +
  theme_void() +
  theme(legend.position="none")
```



```r
b <- last_plot()
# b + scale_fill_discrete()

(reg_names <- unique(as.character(usa49reg@data$region)))
```

```
##  [1] "WA" "MT" "ME" "ND" "SD" "WY" "WI" "ID" "VT" "MN" "OR" "NH" "IA" "MA"
## [15] "NE" "NY" "PA" "CT" "RI" "NJ" "IN" "NV" "UT" "CA" "OH" "IL" "DC" "DE"
## [29] "WV" "MD" "CO" "KY" "KS" "VA" "MO" "AZ" "OK" "NC" "TN" "TX" "NM" "AL"
## [43] "MS" "GA" "SC" "AR" "LA" "FL" "MI"
```

```r
(reg_names_in_gis <- as.character(usa49reg@data$region))
```

```
##  [1] "WA" "MT" "ME" "ND" "SD" "WY" "WI" "ID" "VT" "MN" "OR" "NH" "IA" "MA"
## [15] "NE" "NY" "PA" "CT" "RI" "NJ" "IN" "NV" "UT" "CA" "OH" "IL" "DC" "DE"
## [29] "WV" "MD" "CO" "KY" "KS" "VA" "MO" "AZ" "OK" "NC" "TN" "TX" "NM" "AL"
## [43] "MS" "GA" "SC" "AR" "LA" "FL" "MI"
```

```r
(nreg <- length(reg_names))
```

```
## [1] 49
```

```r
(nreg_in_gis <- length(usa49reg@data$region))
```

```
## [1] 49
```

```r
# Neighbor regions
if (!any((installed.packages())[, "Package"] == "spdep")) install.packages("spdep")
nbr <- spdep::poly2nb(usa49reg)
names(nbr) <- usa49reg@data$region
```

```r
# Centers of the regions
reg_centers <- get_labpt_spdf(usa49reg)
```

Sub-annual time resolution (time-slices)

Here we define two levels of time-slices:
- the day of the year (*YDAY*), from 1 to 365;
- the hour (1 to 24).
Therefore we have 8760 time slices, named according to the format *"dNNN_hNN"*, where *N* - numbers.

For convenience, let's define finctions to convert data-time into slices names and back.

```r
# A list with two levels slices
timeslices365 <- list(
  YDAY = paste0("d", formatC(1:365, width = 3, flag = "0")),
  HOUR = paste0("h", formatC(0:23, width = 2, flag = "0"))
)

# Function to convert data-time object into names of time-slices.
datetime2tsdh <- function(dt) {
  paste0("d", formatC(yday(dt), width = 3, flag = "0"), "_",
         "h", formatC(hour(dt), width = 2, flag = "0"))
}
# check
datetime2tsdh(today("EST"))
```

```
## [1] "d262_h00"
```

```r
# Function to coerse time-slices names into data-time format, for a given year and time-zone.
tsdh2datetime <- function(tslice, year = 2017, tz = "EST") {
  DAY <- as.integer(substr(tslice, 2, 4)) - 1
  HOUR <- as.integer(substr(tslice, 7, 8))
  lubridate::ymd_h(paste0(year, "-01-01 0"), tz = tz) + days(DAY) + hours(HOUR)
}
# check
tsdh2datetime("d365_h23")
```

```
## [1] "2017-12-31 23:00:00 EST"
```

```r
# data.frame object with names of the final time-slices in the model
#   and releted data-time information
slc365 <- tibble(
  slice = kronecker(timeslices365$YDAY, timeslices365$HOUR, FUN = "paste", sep = "_")
)
# add date-time info
slc365$syday <- substr(slc365$slice, 1, 4)
slc365$shour <- substr(slc365$slice, 6, 8)
slc365$yday <- as.integer(substr(slc365$slice, 2, 4))
slc365$hour <- as.integer(substr(slc365$slice, 7, 8))
slc365$datetime <- tsdh2datetime(slc365$slice)
slc365$month <- month(slc365$datetime)
slc365$week <- week(slc365$datetime)
head(slc365)
```

```
## # A tibble: 6 x 8
##   slice    syday shour  yday  hour datetime            month  week
##   <chr>    <chr> <chr> <int> <int> <dttm>              <dbl> <dbl>
## 1 d001_h00 d001  h00       1     0 2017-01-01 00:00:00     1     1
## 2 d001_h01 d001  h01       1     1 2017-01-01 01:00:00     1     1
## 3 d001_h02 d001  h02       1     2 2017-01-01 02:00:00     1     1
## 4 d001_h03 d001  h03       1     3 2017-01-01 03:00:00     1     1
## 5 d001_h04 d001  h04       1     4 2017-01-01 04:00:00     1     1
## 6 d001_h05 d001  h05       1     5 2017-01-01 05:00:00     1     1
```

```
tail(slc365)
```

```
## # A tibble: 6 x 8
##   slice    syday shour  yday  hour datetime            month  week
##   <chr>    <chr> <chr> <int> <int> <dttm>              <dbl> <dbl>
## 1 d365_h18 d365  h18     365    18 2017-12-31 18:00:00    12    53
## 2 d365_h19 d365  h19     365    19 2017-12-31 19:00:00    12    53
## 3 d365_h20 d365  h20     365    20 2017-12-31 20:00:00    12    53
## 4 d365_h21 d365  h21     365    21 2017-12-31 21:00:00    12    53
## 5 d365_h22 d365  h22     365    22 2017-12-31 22:00:00    12    53
## 6 d365_h23 d365  h23     365    23 2017-12-31 23:00:00    12    53
```

Commodities

```
ELC <- newCommodity('ELC', slice = "HOUR")
SOL <- newCommodity('SOL', slice = "HOUR")
WIN <- newCommodity('WIN', slice = "HOUR")
WFF <- newCommodity('WFF', slice = "HOUR")

UHV <- newCommodity(
  name = 'UHV',
  description = "Ultra High Voltage electricity",
  slice = "HOUR")
```

Demand

Hourly demand by states with aggregated load curve (has to be updated with actual data)

```
eiadir <- file.path(getwd(), "data/EIA")
load(file.path(eiadir, "eia_raw.RData"))

# load curve data from:
# https://www.eia.gov/opendata/qb.php?sdid=EBA.US48-ALL.D.H
# Demand for United States Lower 48 (region), hourly - UTC time
lc <- read_csv("data/EIA/Demand_for_United_States_Lower_48_(region)_Hourly.csv",
               skip = 5, col_names = c("datetime", "MWh"))
```

```
## Parsed with column specification:
## cols(
##   datetime = col_character(),
##   MWh = col_double()
## )
```

```r
dim(lc)
```

```
## [1] 33134     2
```

```r
head(lc)
```

```
## # A tibble: 6 x 2
##   datetime           MWh
##   <chr>            <dbl>
## 1 04/11/2019 18H 410282.
## 2 04/11/2019 17H 432074.
## 3 04/11/2019 16H 427778
## 4 04/11/2019 15H 423832.
## 5 04/11/2019 14H 417711.
## 6 04/11/2019 13H 410185.
```

```r
lc$datetime_EST <- mdy_h(lc$datetime, tz = "EST")
lc$year <- year(lc$datetime_EST)
lc <- lc[lc$year == 2017, ]
lc$slice <- datetime2tsdh(lc$datetime_EST)
lc <- lc[order(lc$datetime),]
lc$month <- month(lc$datetime_EST)
lc$hGWh <- lc$MWh/1e3

# Check
dim(lc)
```

```
## [1] 8760    7
```

```r
lc
```

```
## # A tibble: 8,760 x 7
##    datetime          MWh datetime_EST         year slice    month  hGWh
##    <chr>           <dbl> <dttm>              <dbl> <chr>    <dbl> <dbl>
##  1 01/1/2017 00H 459881 2017-01-01 00:00:00  2017 d001_h00     1  460.
##  2 01/1/2017 01H 460324 2017-01-01 01:00:00  2017 d001_h01     1  460.
##  3 01/1/2017 02H 453650 2017-01-01 02:00:00  2017 d001_h02     1  454.
##  4 01/1/2017 03H 441391 2017-01-01 03:00:00  2017 d001_h03     1  441.
##  5 01/1/2017 04H 426998 2017-01-01 04:00:00  2017 d001_h04     1  427.
##  6 01/1/2017 05H 413734 2017-01-01 05:00:00  2017 d001_h05     1  414.
##  7 01/1/2017 06H 401305 2017-01-01 06:00:00  2017 d001_h06     1  401.
##  8 01/1/2017 07H 391149 2017-01-01 07:00:00  2017 d001_h07     1  391.
##  9 01/1/2017 08H 381509 2017-01-01 08:00:00  2017 d001_h08     1  382.
## 10 01/1/2017 09H 374644 2017-01-01 09:00:00  2017 d001_h09     1  375.
## # ... with 8,750 more rows
```

```r
sum(lc$MWh) / 1e6
```

```
## [1] 3957.126
```

```r
summary(lc$hGWh)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   290.9   401.2   442.8   451.7   486.2   717.8
```

```r
mlc <- group_by(lc, month) %>%
  summarise(mGWh = sum(MWh)/1e3)
loadcurve <- full_join(select(lc, datetime_EST, slice, hGWh, month), mlc)
```

```
## Joining, by = "month"
```

```
loadcurve$mshare <- loadcurve$hGWh / loadcurve$mGWh

# Check
group_by(loadcurve, month) %>%
  summarise(msum = sum(mshare))
```

```
## # A tibble: 12 x 2
##    month  msum
##    <dbl> <dbl>
##  1     1 1
##  2     2 1
##  3     3 1
##  4     4 1
##  5     5 1
##  6     6 1.000
##  7     7 1
##  8     8 1
##  9     9 1
## 10    10 1
## 11    11 1.000
## 12    12 1
```

```
elc_gen_2017 <- elc_gen[elc_gen$YEAR == 2017 &
                          elc_gen$`TYPE OF PRODUCER` == "Total Electric Power Industry" &
                          elc_gen$`ENERGY SOURCE` == "Total" &
                          elc_gen$STATE != "US-Total" &
                          elc_gen$STATE != "AK" &
                          elc_gen$STATE != "HI",]
elc_gen_2017
```

```
## # A tibble: 588 x 6
##     YEAR MONTH STATE `TYPE OF PRODUCER`  `ENERGY SOURCE` `GENERATION\r\n(M~
##    <dbl> <dbl> <chr> <chr>               <chr>                        <dbl>
##  1  2017     1 AL    Total Electric Pow~ Total                     10940022
##  2  2017     1 AR    Total Electric Pow~ Total                      5413456
##  3  2017     1 AZ    Total Electric Pow~ Total                      7978430
##  4  2017     1 CA    Total Electric Pow~ Total                     17444705
##  5  2017     1 CO    Total Electric Pow~ Total                      4967360
##  6  2017     1 CT    Total Electric Pow~ Total                      3235848
##  7  2017     1 DC    Total Electric Pow~ Total                         9717
##  8  2017     1 DE    Total Electric Pow~ Total                       475758
##  9  2017     1 FL    Total Electric Pow~ Total                     17547018
## 10  2017     1 GA    Total Electric Pow~ Total                     10303007
## # ... with 578 more rows
```

```
sum(elc_gen_2017$`GENERATION\r\n(Megawatthours)`) / 1e6
```

```
## [1] 3998.985
```

```
sum(lc$MWh) / 1e6
```

```
## [1] 3957.126
```

```
length(unique(elc_gen_2017$STATE))
```

```
## [1] 49
```

```r
elc_gen_2017$GWh <- elc_gen_2017$`GENERATION\r\n(Megawatthours)`/1e3
elc_gen_2017$month <- as.integer(elc_gen_2017$MONTH)
elc_gen_2017$region <- elc_gen_2017$STATE
elc_gen_2017 <- select(elc_gen_2017, month, region, GWh)

elc_gen_mhr <- full_join(loadcurve, elc_gen_2017)
```

```
## Joining, by = "month"
```

```r
dim(elc_gen_mhr)[1]/49/365/24 == 1 # Check
```

```
## [1] TRUE
```

```r
elc_gen_mhr$GWh <- elc_gen_mhr$GWh * elc_gen_mhr$mshare
sum(elc_gen_mhr$GWh); sum(lc$hGWh); sum(elc_gen_2017$GWh)
```

```
## [1] 3998985
```

```
## [1] 3957126
```

```
## [1] 3998985
```

```r
# dim(elc_gen_mhr)
loadcurve <- select(elc_gen_mhr, -mGWh, -hGWh)
summary(loadcurve$GWh)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00249  3.81024  7.20024  9.31643 12.07852 84.28890
```

```r
# Demand class
DEM_ELC_DH <- newDemand(
  name = "DEM_ELC_DH",
  commodity = "ELC",
  dem = data.frame(
    year = 2017,
    region = c(
      loadcurve$region),
    slice = c(
      loadcurve$slice),
    dem = round(loadcurve$GWh, 7)
    )
  # slice = "HOUR"
)

# Check
dim(DEM_ELC_DH@dem)
```

```
## [1] 429240      4
```

```r
dim(DEM_ELC_DH@dem)[1] / 365 / 24
```

```
## [1] 49
```

```r
summary(DEM_ELC_DH@dem$dem)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00249  3.81024  7.20024  9.31643 12.07852 84.28890
```

```r
# Demand data for mapping
elc_gen_map <- elc_gen_2017 %>%
```

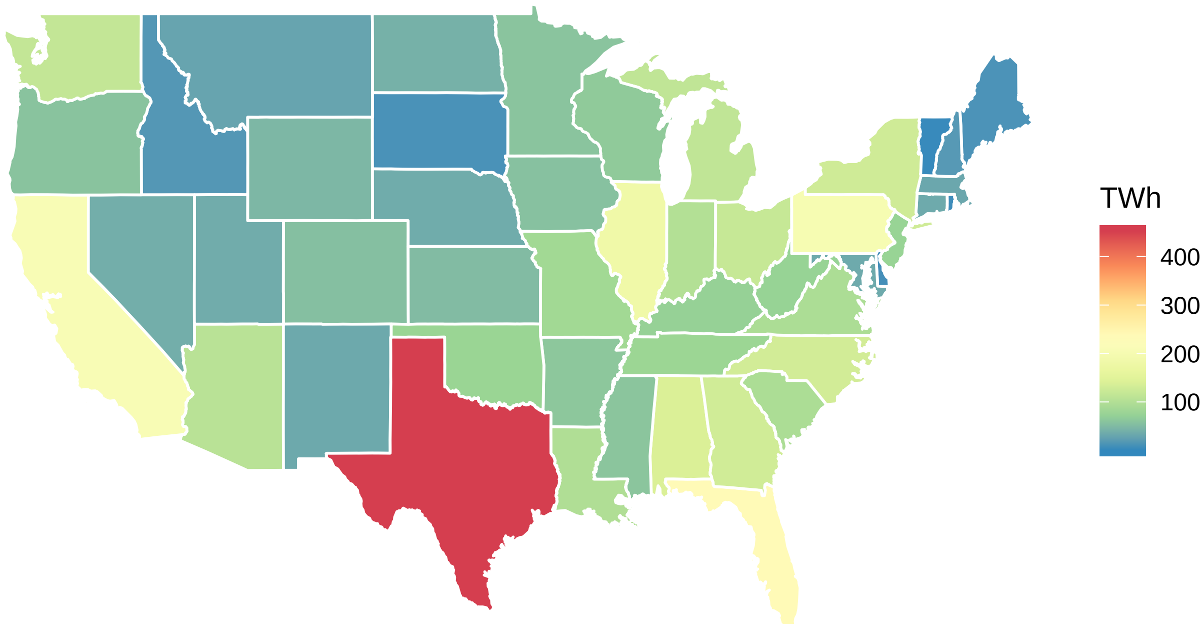
```r
  group_by(region) %>%
  summarise(TWh = sum(GWh)/1e3) %>%
  full_join(usa49r, by = c("region" = "id"))
```

```
## Warning: Column `region`/`id` joining character vector and factor, coercing
## into character vector
```

```r
elc_gen_map
```

```
## # A tibble: 11,481 x 8
##    region   TWh  long   lat order hole  piece group
##    <chr>  <dbl> <dbl> <dbl> <int> <lgl> <fct> <fct>
##  1 AL      139. -85.1  32.0     1 FALSE 1     AL.1
##  2 AL      139. -85.1  31.9     2 FALSE 1     AL.1
##  3 AL      139. -85.1  31.9     3 FALSE 1     AL.1
##  4 AL      139. -85.1  31.8     4 FALSE 1     AL.1
##  5 AL      139. -85.1  31.8     5 FALSE 1     AL.1
##  6 AL      139. -85.1  31.7     6 FALSE 1     AL.1
##  7 AL      139. -85.1  31.7     7 FALSE 1     AL.1
##  8 AL      139. -85.1  31.7     8 FALSE 1     AL.1
##  9 AL      139. -85.1  31.6     9 FALSE 1     AL.1
## 10 AL      139. -85.0  31.6    10 FALSE 1     AL.1
## # ... with 11,471 more rows
```

```r
ggplot(elc_gen_map) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = TWh), # fill = "wheat",
               colour = "white", alpha = 1, size = .5) +
  scale_fill_distiller(palette = "Spectral") +
  coord_fixed(1.45) +
  theme_void()
```




```r
ggsave("fig/elc_gen_map.png")
```

```
## Saving 6.5 x 4.5 in image
```

Weather factors

Hourly weather information is used to estimate output of intermittent renewables, solar PVs and wind tourbines. The weather information is supplied as multipliers to availability factors of the generating technologies. I.e. availability of solar energy can be estimated based on solar radiation (flux) data, or actual output of the technology (PV) expected under complex weather data (direct and indirect solar irradience, temperature, and/or clouds etc.). The estimation can be done on grid data, then aggregated using potentially perspective and available locations for the installation of the sollar arrays. Here, for simplicity, we estimate availability of the solar resource based on NASA's *"surface net downward shortwave flux" (SWGNT, Watts per square meter)*, assuming that the full capacity of PV arrays is achieved at 800 W/m2 level of the flux. This estimate is certainly not the best one, can be improved.
The estimated factors are aggregated for all availabel locations by states (which implies that allocation of PVs capacity is assumed to be evenly distributed across all the therritory in each state).

For wind energy, wind-power curves are used to estimate potential output of wind-mills across locations with the best potential of wind energy.

Solar availability factors

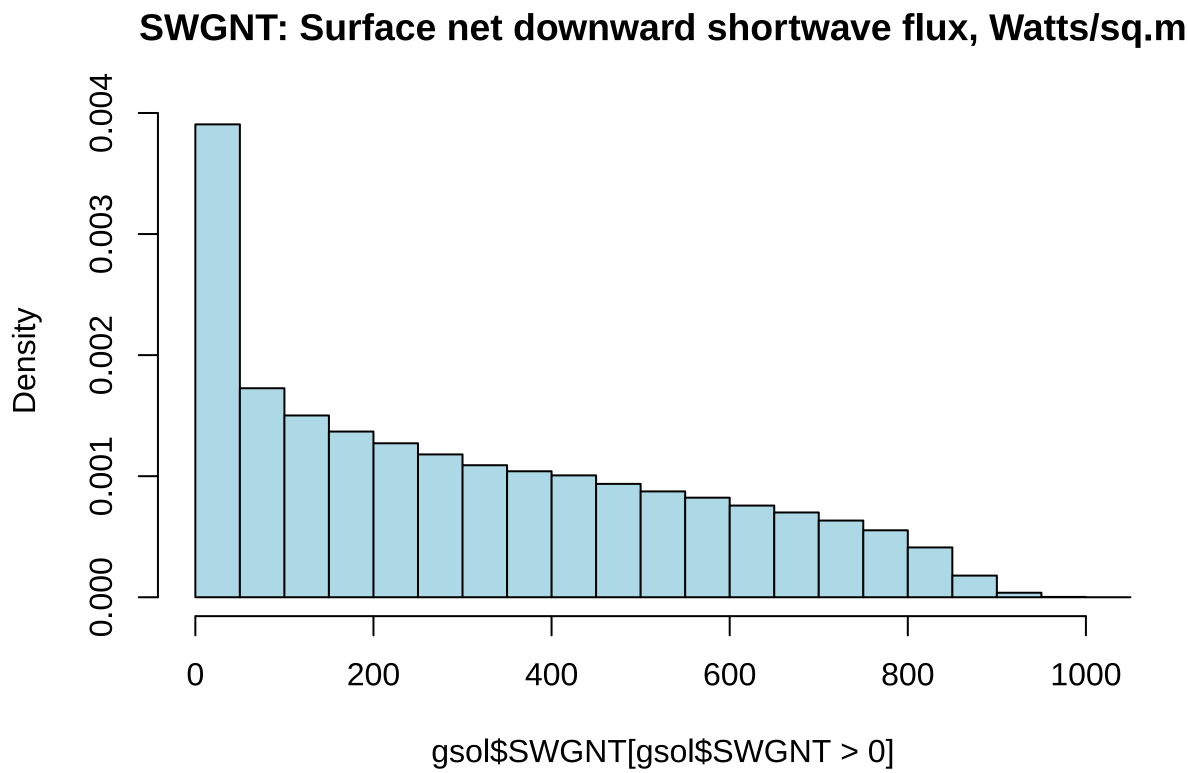
```r
# MERRA-2 data:
# https://gmao.gsfc.nasa.gov/reanalysis/MERRA-2/data_access/
# here we use data for one year only
load("data/MERRA2/nasa_sol_US49.RData")
gsol
```

```
## # A tibble: 22,916,160 x 11
##    datetime            loc_id SWGDN SWGNT   lon   lat region  hour month
##    <dttm>               <int> <dbl> <dbl> <dbl> <dbl> <fct>  <int> <fct>
##  1 2017-01-01 00:30:00 133216     0     0 -80.6  25.5 FL         0 Janu~
##  2 2017-01-01 00:30:00 133765     0     0 -97.5  26   TX         0 Janu~
##  3 2017-01-01 00:30:00 133791     0     0 -81.2  26   FL         0 Janu~
##  4 2017-01-01 00:30:00 133792     0     0 -80.6  26   FL         0 Janu~
##  5 2017-01-01 00:30:00 134339     0     0 -98.8  26.5 TX         0 Janu~
##  6 2017-01-01 00:30:00 134340     0     0 -98.1  26.5 TX         0 Janu~
##  7 2017-01-01 00:30:00 134341     0     0 -97.5  26.5 TX         0 Janu~
##  8 2017-01-01 00:30:00 134366     0     0 -81.9  26.5 FL         0 Janu~
##  9 2017-01-01 00:30:00 134367     0     0 -81.2  26.5 FL         0 Janu~
## 10 2017-01-01 00:30:00 134368     0     0 -80.6  26.5 FL         0 Janu~
## # ... with 22,916,150 more rows, and 2 more variables: mdays <int>,
## #   yday <dbl>
```

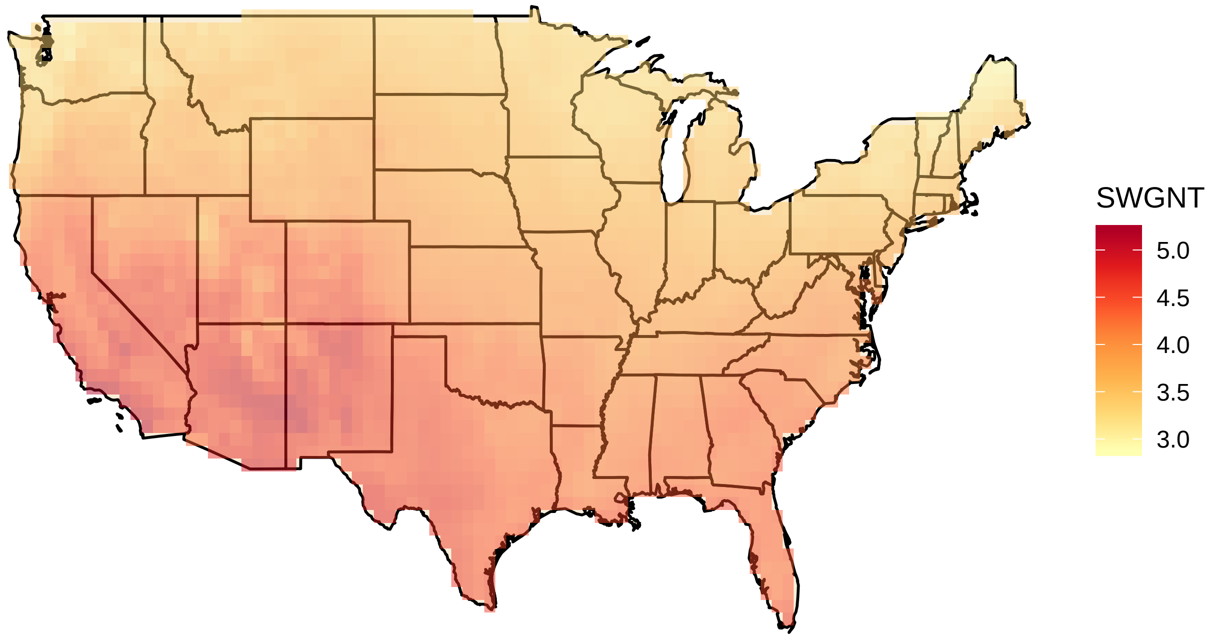
```r
summary(gsol$SWGNT)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##    0.000    0.000    7.051  167.446  294.875 1011.000
```

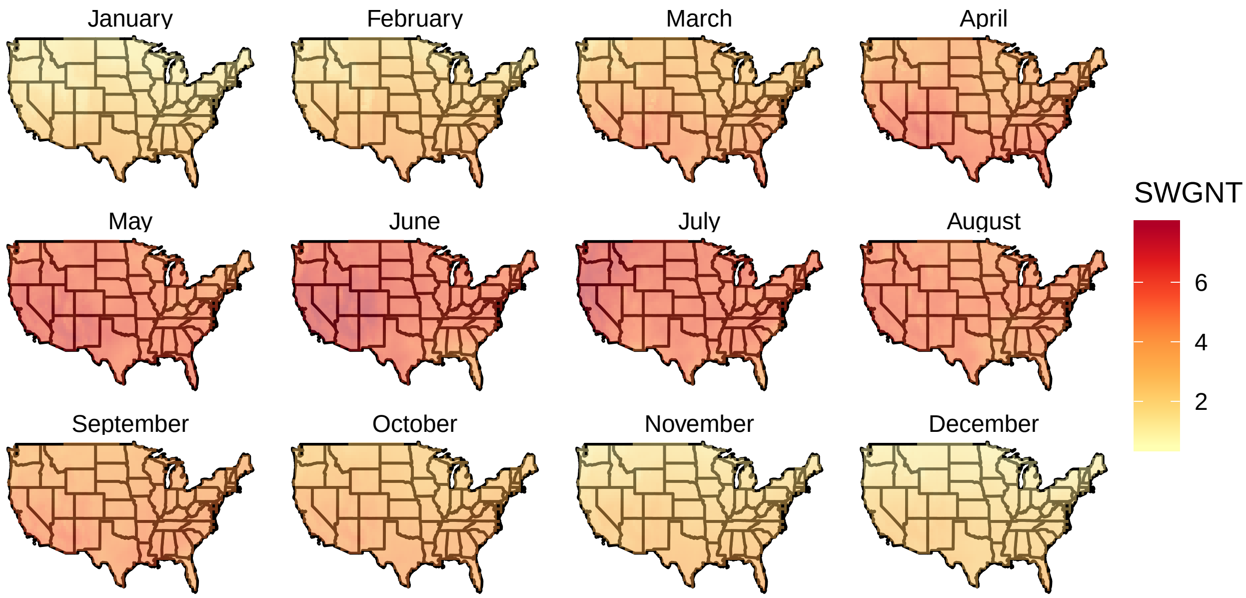
```r
hist(gsol$SWGNT[gsol$SWGNT > 0], col = "lightblue", probability = T,
     main = "SWGNT: Surface net downward shortwave flux, Watts/sq.m")
```




```r
# Annual aggregate, kWh/day
ysol <- group_by(gsol, loc_id, lat, lon, region) %>%
  summarise(SWGNT = sum(SWGNT)/365/1e3)

ggplot(usa49r, aes(long, lat, group = group)) +
  geom_polygon(colour = "black", fill = "wheat", alpha = .5) +
  coord_fixed(1.45) +
  theme_void() +
  geom_raster(data = ysol, aes(lon, lat, fill = SWGNT), interpolate = F, inherit.aes = F, alpha = .95) +
  scale_fill_distiller(palette = "YlOrRd", direction = 1)
```




```r
# By months, kWh/day
msol <- group_by(gsol, loc_id, lat, lon, region, month) %>%
  summarise(SWGNT = sum(SWGNT/mdays)/1e3)

ggplot(usa49r, aes(long, lat, group = group)) +
  geom_polygon(colour = "black", fill = "wheat", alpha = .5) +
  coord_fixed(1.45) +
  theme_void() +
  geom_raster(data = msol, aes(lon, lat, fill = SWGNT), interpolate = F, inherit.aes = F, alpha = .95) +
  scale_fill_distiller(palette = "YlOrRd", direction = 1) +
  facet_wrap(.~month)
```




```r
# Aggregation regions
dhsol <- group_by(gsol, region, yday, hour) %>% # lat, lon,
  summarise(SWGNT = mean(SWGNT))
summary(dhsol$SWGNT)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   7.611 162.951 291.846 937.517
```

```r
# Estimated weather factors
dhsol$WF <- dhsol$SWGNT / 800
dhsol$WF[dhsol$WF > 1] <- 1
dhsol$WF[dhsol$WF < .03] <- 0 # kick-starting irradiance
summary(dhsol$WF)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.2025  0.3648  1.0000
```

```r
dhsol
```

```
## # A tibble: 420,480 x 5
## # Groups:   region, yday [17,520]
##    region  yday  hour SWGNT    WF
##    <fct>  <dbl> <int> <dbl> <dbl>
##  1 AL         1     0 0     0
##  2 AL         1     1 0     0
##  3 AL         1     2 0     0
##  4 AL         1     3 0     0
##  5 AL         1     4 0     0
```

```
##   6 AL          1     5  0      0
##   7 AL          1     6  0      0
##   8 AL          1     7  0.281  0
##   9 AL          1     8 23.9    0
##  10 AL          1     9 75.4    0.0942
## # ... with 420,470 more rows
```

```
# Add slice-names
dhsol$slice <- paste0("d", formatC(dhsol$yday, width = 3, flag = "0"),
                      "_h", formatC(dhsol$hour, width = 2, flag = "0"))
dhsol
```

```
## # A tibble: 420,480 x 6
## # Groups:   region, yday [17,520]
##    region  yday  hour  SWGNT     WF slice
##    <fct>  <dbl> <int>  <dbl>  <dbl> <chr>
##  1 AL         1     0  0      0      d001_h00
##  2 AL         1     1  0      0      d001_h01
##  3 AL         1     2  0      0      d001_h02
##  4 AL         1     3  0      0      d001_h03
##  5 AL         1     4  0      0      d001_h04
##  6 AL         1     5  0      0      d001_h05
##  7 AL         1     6  0      0      d001_h06
##  8 AL         1     7  0.281  0      d001_h07
##  9 AL         1     8 23.9    0      d001_h08
## 10 AL         1     9 75.4    0.0942 d001_h09
## # ... with 420,470 more rows
```

```
dim(dhsol)[1] / 365/24 # 48, i.e. one region is missing - DC
```

```
## [1] 48
```

```
dhsol$region <- as.character(dhsol$region)
reg_names[!(reg_names %in% unique(dhsol$region))]
```

```
## [1] "DC"
```

```
# Use MD weather data for DC
dhsol_DC <- dhsol[dhsol$region == "MD",]
dim(dhsol_DC)
```

```
## [1] 8760    6
```

```
dhsol_DC$region <- "DC"
dhsol_DC
```

```
## # A tibble: 8,760 x 6
## # Groups:   region, yday [365]
##    region  yday  hour SWGNT    WF slice
##    <chr>  <dbl> <int> <dbl> <dbl> <chr>
##  1 DC         1     0   0    0     d001_h00
##  2 DC         1     1   0    0     d001_h01
##  3 DC         1     2   0    0     d001_h02
##  4 DC         1     3   0    0     d001_h03
##  5 DC         1     4   0    0     d001_h04
##  6 DC         1     5   0    0     d001_h05
##  7 DC         1     6   0    0     d001_h06
##  8 DC         1     7  10.8  0     d001_h07
```

```
##   9 DC          1     8  96.1 0.120 d001_h08
## 10 DC          1     9 202.  0.253 d001_h09
## # ... with 8,750 more rows
```

```r
# Add DC
dhsol <- bind_rows(dhsol, dhsol_DC)
dim(dhsol)[1] / 365/24
```

```
## [1] 49
```

```r
length(unique(dhsol$region)) == nreg # double-check
```

```
## [1] TRUE
```

```r
size(gsol); rm(gsol)
```

```
## [1] "1.5 Gb"
```

Wind availability factors

```r
# Hourly wind speed data at 50 meters height for US, 2017
# (source: NASA/MERRA2, preprocessed)
load("data/merra2/nasa_wnd_US49.RData")
gwnd
```

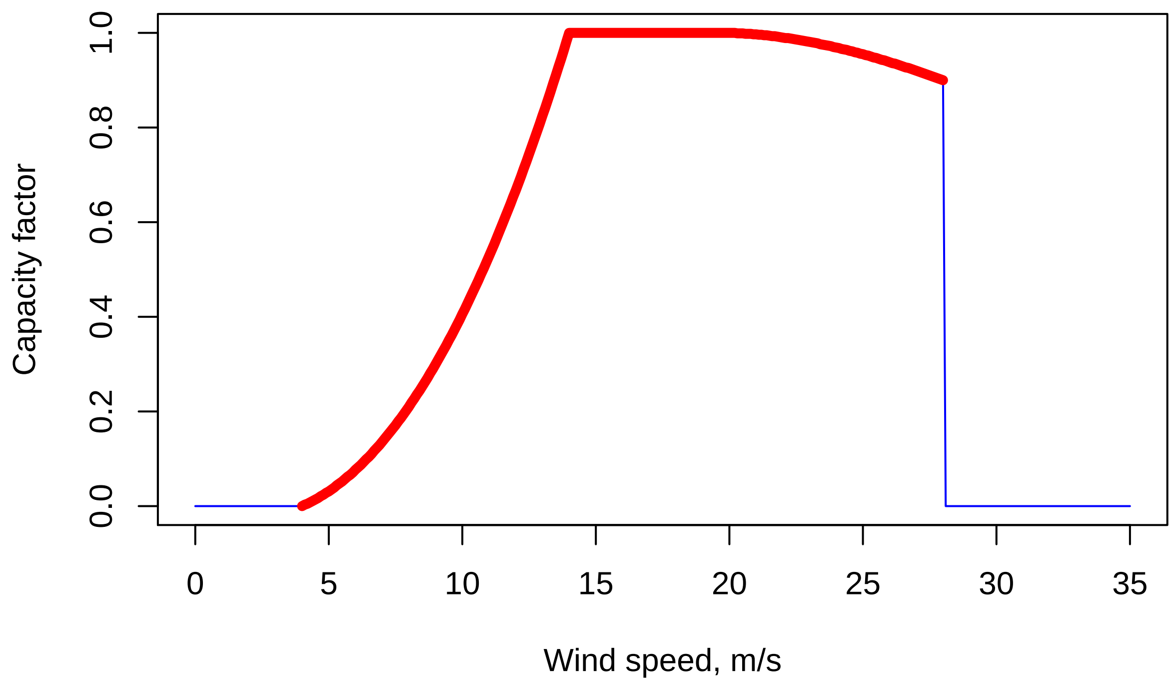
```
## # A tibble: 26,481,480 x 19
##    datetime            loc_id WS50M   lon   lat US_100km_buffer
##    <dttm>               <int> <dbl> <dbl> <dbl> <lgl>
##  1 2017-01-01 00:30:00 132062  7.46 -81.9  24.5 TRUE
##  2 2017-01-01 00:30:00 132063  7.67 -81.2  24.5 TRUE
##  3 2017-01-01 00:30:00 132064  7.78 -80.6  24.5 TRUE
##  4 2017-01-01 00:30:00 132065  7.92 -80    24.5 TRUE
##  5 2017-01-01 00:30:00 132638  6.98 -81.9  25   TRUE
##  6 2017-01-01 00:30:00 132639  6.93 -81.2  25   TRUE
##  7 2017-01-01 00:30:00 132640  7.43 -80.6  25   TRUE
##  8 2017-01-01 00:30:00 132641  7.82 -80    25   TRUE
##  9 2017-01-01 00:30:00 132642  7.98 -79.4  25   TRUE
## 10 2017-01-01 00:30:00 133213  6.70 -82.5  25.5 TRUE
## # ... with 26,481,470 more rows, and 13 more variables:
## #   US_10km_buffer <lgl>, non_US <lgl>, drop <lgl>, US_land <lgl>,
## #   offshore <lgl>, nearest_neighbor <chr>, region <fct>, buff_10km <chr>,
## #   offshore_names <chr>, hour <int>, month <fct>, mdays <int>, yday <int>
```

```r
if (is.factor(gwnd$region)) gwnd$region <- as.character(gwnd$region)

# Simplified aggregated wind-power curve as a function of wind speed.
# (assumed, should be replaced by read)
WindPowerCurve <- function(x = NULL,
                           xcutin = 4, xpeak = 14, xpeak2 = 20, xcutoff = 28,
                           ycutoff = .9, round = 3) {
  # x - wind speed in m/s
  # xcutin, xcutoff - operational speed of wind (m/s, min and max respectively)
  # xpeak, xpeak2 - the range of speed of with peak (nameplate) power
  # ycutoff - output factor at cut off (max) wind speed
  ff1 <- function(x1) {
    xx <- c(xcutin, 0.5 * (xpeak + xcutin), xpeak)
```

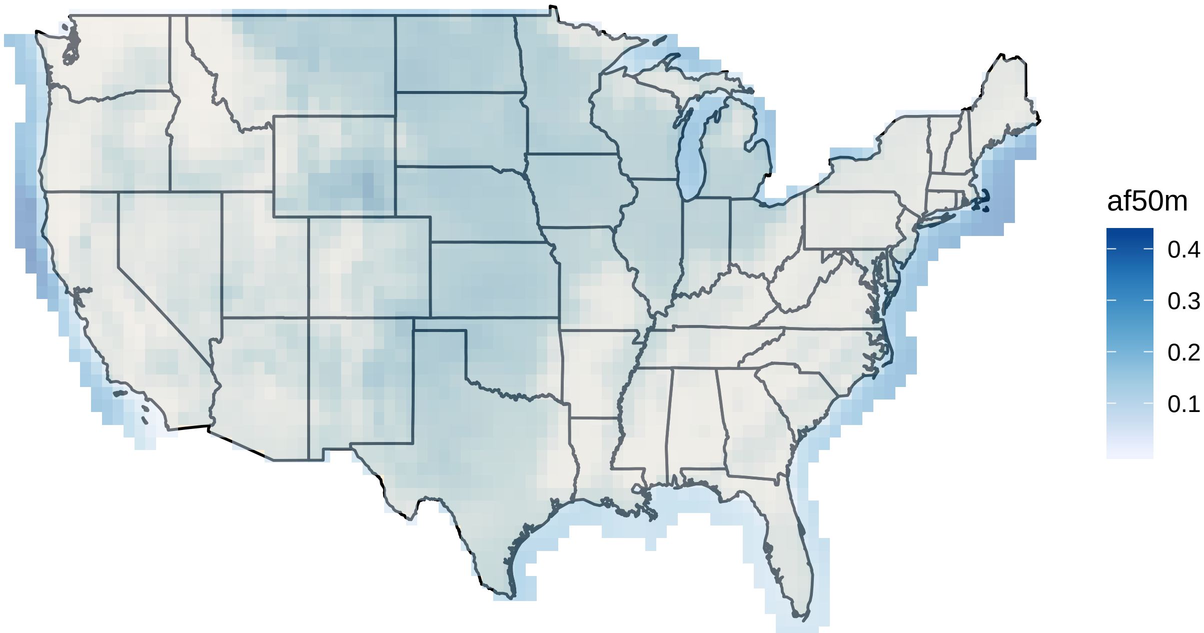
```
    xx2 <- xx * xx
    xx3 <- xx * xx2
    yy <- c(0, .3, 1)
    ff <- lm(yy ~ 0 + xx + xx + xx2 + xx3)
    predict(ff, data.frame(
      xx = x1,
      xx2 = x1 * x1,
      xx3 = x1 * x1 * x1
    ))
  }
  ff2 <- function(x2) {
    xx <- c(xpeak2, 0.5 * (xcutoff + xpeak2), xcutoff)
    xx2 <- xx * xx
    xx3 <- xx * xx2
    yy <- c(1, .51 * (ycutoff + 1), ycutoff)
    ff <- lm(yy ~ 0 + xx + xx + xx2 + xx3)
    predict(ff, data.frame(
      xx = x2,
      xx2 = x2 * x2,
      xx3 = x2 * x2 * x2
    ))
  }
  y <- rep(0., length(x))
  ii <- x <= xcutin
  y[ii] <- 0
  ii <- x > xcutoff
  y[ii] <- 0
  ii <- x >= xpeak & x <= xpeak2
  y[ii] <- 1
  ii <- x > xcutin & x < xpeak
  y[ii] <- ff1(x[ii])
  ii <- x > xpeak2 & x <= xcutoff
  y[ii] <- ff2(x[ii])
  return(round(y, round))
  # splinefun(xx, yy, method = "natural")
}

# Check
WindPowerCurve(0:30)
```

```
##  [1] 0.000 0.000 0.000 0.000 0.000 0.031 0.077 0.136 0.210 0.300 0.406
## [12] 0.528 0.667 0.825 1.000 1.000 1.000 1.000 1.000 1.000 1.000 0.997
## [23] 0.990 0.981 0.969 0.955 0.938 0.920 0.900 0.000 0.000
```

```
x <- seq(0, 35, by = .1)
plot(x, WindPowerCurve(x), type = "l", col = "blue", lwd = 1,
     xlab = "Wind speed, m/s", ylab = "Capacity factor")
x1 <- seq(4, 28, by = .1)
points(x1, WindPowerCurve(x1), type = "l", col = "red", lwd = 5)
```




```r
# Availability factor for wind-energy technologies,
# estimated based on the wind-power curve
gwnd$af50m <- WindPowerCurve(gwnd$WS50M)

# Annual availability factor of wind turbines by location
ywnd <- group_by(gwnd, loc_id, lat, lon, region) %>%
  summarise(af50m = sum(af50m)/365/24)

ggywnd <- function(ii = rep(T, length(ywnd$loc_id))) {
  ggplot(usa49r, aes(long,lat, group = group)) +
    geom_polygon(colour = "black", fill = "wheat", alpha = .5) +
    coord_fixed(1.45) +
    theme_void() +
    geom_raster(data = ywnd[ii,], aes(lon, lat, fill = af50m),
                interpolate = F, inherit.aes = F, alpha = .95) +
    scale_fill_distiller(palette = "Blues", direction = 1)
}
ggywnd()
```

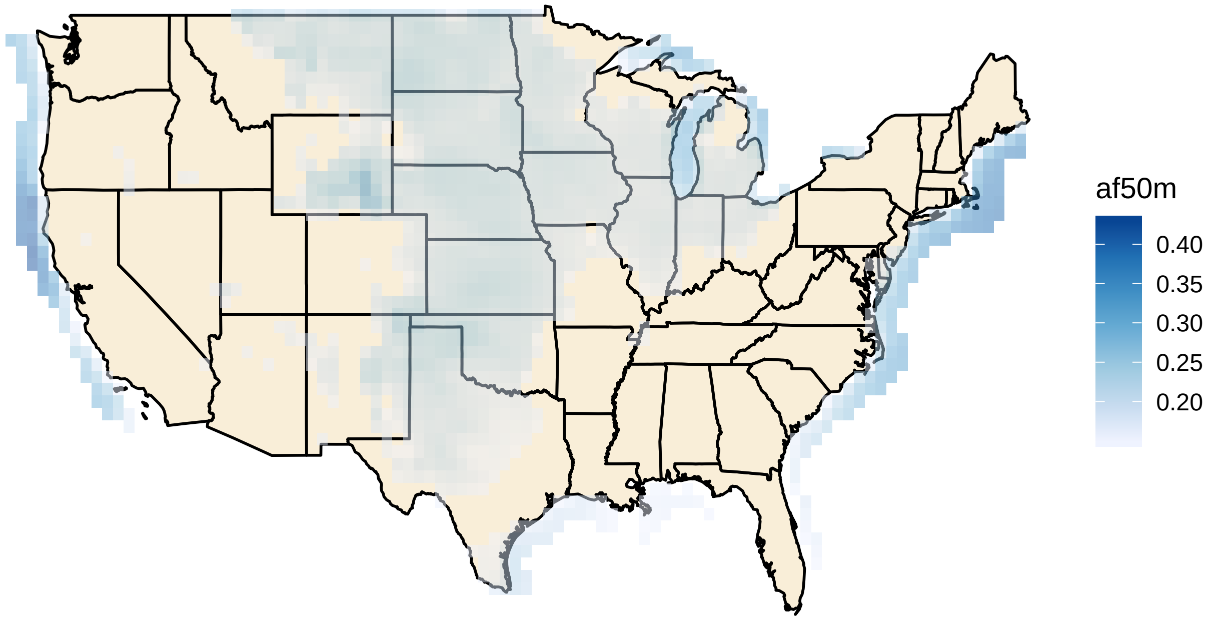



```r
# filtered
ii <- ywnd$af50m >= .15; summary(ii)
```

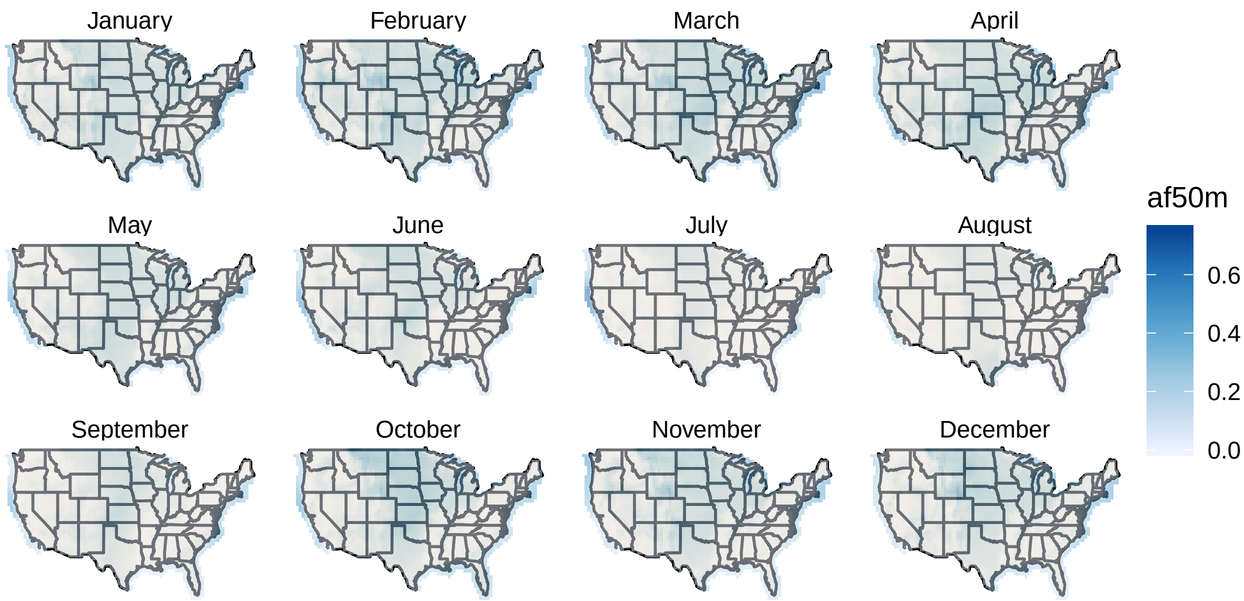
```
##    Mode   FALSE    TRUE
## logical    1723    1300
```

```r
ggywnd(ii)
```

```
## Warning in f(...): Raster pixels are placed at uneven horizontal intervals
## and will be shifted. Consider using geom_tile() instead.
```



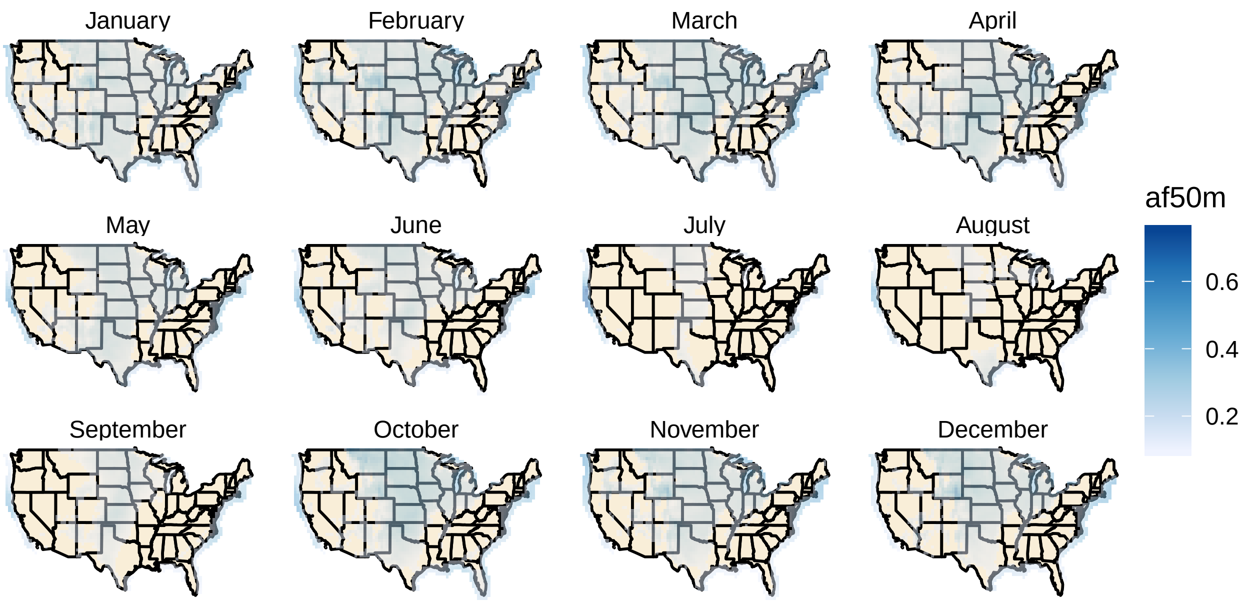

```
# Aggregation by months
mwnd <- group_by(gwnd, loc_id, lat, lon, region, month) %>%
  summarise(af50m = sum(af50m/mdays)/24)

ggmwnd <- function(mm = rep(T, length(mwnd$loc_id))) {
  ggplot(usa49r, aes(long,lat, group = group)) +
    geom_polygon(colour = "black", fill = "wheat", alpha = .5) +
    coord_fixed(1.45) +
    theme_void() +
    geom_raster(data = mwnd[mm,], aes(lon, lat, fill = af50m),
                interpolate = F, inherit.aes = F, alpha = .95) +
    scale_fill_distiller(palette = "Blues", direction = 1) + # , trans = "log10"
    facet_wrap(.~month)
}
ggmwnd()
```




```r
mm <- mwnd$af50m >= .1; summary(mm)
```

```
##     Mode   FALSE    TRUE
## logical   16570   19706
```

```r
ggmwnd(mm)
```




```r
# Averaging by annual potential
# `loc_id_??` is locations with the ranged potential
yy_20 <- ywnd$af50m >= .20; summary(yy_20); loc_id_20 <- ywnd$loc_id[yy_20]
```

```
##    Mode   FALSE    TRUE
## logical    2303     720
```

```r
yy_15 <- ywnd$af50m >= .15 & ywnd$af50m < .20; summary(yy_15); loc_id_15 <- ywnd$loc_id[yy_15]
```

```
##    Mode   FALSE    TRUE
## logical    2443     580
```

```r
yy_10 <- ywnd$af50m >= .10 & ywnd$af50m < .15; summary(yy_10); loc_id_10 <- ywnd$loc_id[yy_10]
```

```
##    Mode   FALSE    TRUE
## logical    2553     470
```

```r
# Check for overlay
any(loc_id_20 %in% loc_id_15) | any(loc_id_10 %in% loc_id_20)
```

```
## [1] FALSE
```

```r
# Assuming 4 megawatts per square kilometer (about 10 megawatts per square mile, NREL)
# for offshore: 3-5 megawatts (MW) per square kilometer
# Taking one grid cell is about ~50x60 km, or ~3000 sq.km,
# one grid cell can accomodate up to 12GW (3000 * 4 / 1e3) onshore or 9GW offshore.
# For simplicity let's assume up to 5GW can be used for both
# on- and off-shore wind power capacity per one cell of the grid.
# hist(gwnd$af50m)
```

```r
gwnd$max_GWh <- 5 * gwnd$af50m # X GW for hourly data

# Onshore wind resource
# Filter and aggregate by regions locations
# with potential >20% availability factor of wind
ii <- gwnd$loc_id %in% loc_id_20 & gwnd$US_10km_buffer
summary(ii)
```

```
##    Mode    FALSE     TRUE
## logical 21707280  4774200
```

```r
wnd_af20 <- group_by(gwnd[ii,], datetime, region) %>%
  summarise(max_GWh = sum(max_GWh, na.rm = T),
            af50m = mean(af50m))
wnd_af20
```

```
## # A tibble: 210,240 x 4
## # Groups:   datetime [8,760]
##    datetime            region max_GWh  af50m
##    <dttm>              <chr>    <dbl>  <dbl>
##  1 2017-01-01 00:30:00 CA        2.7  0.54
##  2 2017-01-01 00:30:00 CO        9.74 0.244
##  3 2017-01-01 00:30:00 DE        5    1
##  4 2017-01-01 00:30:00 IA       41.6  0.308
##  5 2017-01-01 00:30:00 IL        0.04 0.004
##  6 2017-01-01 00:30:00 KS       24.7  0.0771
##  7 2017-01-01 00:30:00 MA        9.48 0.948
##  8 2017-01-01 00:30:00 MD        3.1  0.62
##  9 2017-01-01 00:30:00 MI       25.1  0.295
## 10 2017-01-01 00:30:00 MN       78.7  0.375
## # ... with 210,230 more rows
```

```r
sum(wnd_af20$max_GWh)/1e3 # Total estimated onshore potential
```

```
## [1] 5288.6
```

```r
unique(wnd_af20$region) # regions with onshore wind potential
```

```
##  [1] "CA" "CO" "DE" "IA" "IL" "KS" "MA" "MD" "MI" "MN" "MT" "NC" "ND" "NE"
## [15] "NJ" "NM" "NY" "OH" "OK" "SD" "TX" "VA" "WI" "WY"
```

```r
# Offshore wind resource
ii <- gwnd$loc_id %in% loc_id_20 & gwnd$offshore
summary(ii)
```

```
##    Mode    FALSE     TRUE
## logical 24948480  1533000
```

```r
wndf_af20 <- group_by(gwnd[ii,], datetime, region) %>%
  summarise(max_GWh = sum(max_GWh, na.rm = T),
            af50m = mean(af50m))
wndf_af20
```

```
## # A tibble: 166,440 x 4
## # Groups:   datetime [8,760]
##    datetime            region max_GWh af50m
##    <dttm>              <chr>    <dbl> <dbl>
```

```
##   1 2017-01-01 00:30:00 CA      76.0   0.460
##   2 2017-01-01 00:30:00 DE       5     1
##   3 2017-01-01 00:30:00 IL       0.265 0.053
##   4 2017-01-01 00:30:00 MA      72.9   0.972
##   5 2017-01-01 00:30:00 MD      10     1
##   6 2017-01-01 00:30:00 ME      47.8   0.956
##   7 2017-01-01 00:30:00 MI      55.6   0.463
##   8 2017-01-01 00:30:00 MN       5     1
##   9 2017-01-01 00:30:00 NC      59.4   0.517
##  10 2017-01-01 00:30:00 NJ      29.1   0.972
## # ... with 166,430 more rows
```

```r
sum(wndf_af20$max_GWh)/1e3 # Total offshore wind potential
```

```
## [1] 2250.915
```

```r
unique(wndf_af20$region) # regions with offshore wind potential
```

```
##  [1] "CA" "DE" "IL" "MA" "MD" "ME" "MI" "MN" "NC" "NJ" "NY" "OH" "OR" "RI"
## [15] "SC" "TX" "VA" "WA" "WI"
```

Weather classes in the model

Weather classes (in the energyRt package) are used to store/add weather factors to the model data.

```r
# Solar AF
# dim(dhsol)
# head(dhsol)
ii <- rep(T, dim(dhsol)[1]) # filter if needed

SOLAR_AF <- newWeather("SOLAR_AF",
                       description = "Ground level insolation AF",
                       # unit = "kWh/kWh_max",
                       slice = "HOUR",
                       weather = data.frame(
                         region = as.character(dhsol$region[ii]),
                         # year = 2017,
                         slice = dhsol$slice[ii],
                         wval = dhsol$WF[ii]
                       ))

head(SOLAR_AF@weather)
```

```
##   region year    slice wval
## 1     AL   NA d001_h00    0
## 2     AL   NA d001_h01    0
## 3     AL   NA d001_h02    0
## 4     AL   NA d001_h03    0
## 5     AL   NA d001_h04    0
## 6     AL   NA d001_h05    0
```

```r
dim(SOLAR_AF@weather)[1] / 49 / 365 / 24 # Check: must be == 1
```

```
## [1] 1
```

```r
# Wind AF (onshore)
wnd_af20$slice <- datetime2tsdh(wnd_af20$datetime)
wnd_af20
```

```
## # A tibble: 210,240 x 5
## # Groups:   datetime [8,760]
##    datetime            region max_GWh  af50m slice
##    <dttm>              <chr>    <dbl>  <dbl> <chr>
##  1 2017-01-01 00:30:00 CA        2.7  0.54   d001_h00
##  2 2017-01-01 00:30:00 CO        9.74 0.244  d001_h00
##  3 2017-01-01 00:30:00 DE        5    1      d001_h00
##  4 2017-01-01 00:30:00 IA       41.6  0.308  d001_h00
##  5 2017-01-01 00:30:00 IL        0.04 0.004  d001_h00
##  6 2017-01-01 00:30:00 KS       24.7  0.0771 d001_h00
##  7 2017-01-01 00:30:00 MA        9.48 0.948  d001_h00
##  8 2017-01-01 00:30:00 MD        3.1  0.62   d001_h00
##  9 2017-01-01 00:30:00 MI       25.1  0.295  d001_h00
## 10 2017-01-01 00:30:00 MN       78.7  0.375  d001_h00
## # ... with 210,230 more rows
```

```r
dim(wnd_af20)[1] / 365/24 # number of regions with the onshore potential
```

```
## [1] 24
```

```r
ii <- rep(T, dim(wnd_af20)[1]) #
summary(wnd_af20$af50m)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0494  0.1493  0.2253  0.3231  1.0000
```

```r
# wnd_af20$region <- as.character(wnd_af20$region)
WINON20_AF <- newWeather("WINON20_AF",
                         description = "Onshore wind potential af20",
                         region = unique(wnd_af20$region[ii]),
                         # unit = "kWh/kWh_max",
                         slice = "HOUR",
                         weather = data.frame(
                           region = as.character(wnd_af20$region[ii]),
                           # year = 2010,
                           slice = wnd_af20$slice[ii],
                           wval = wnd_af20$af50m[ii]
                         ))
head(WINON20_AF@weather)
```

```
##   region year    slice       wval
## 1     CA   NA d001_h00 0.54000000
## 2     CO   NA d001_h00 0.24350000
## 3     DE   NA d001_h00 1.00000000
## 4     IA   NA d001_h00 0.30848148
## 5     IL   NA d001_h00 0.00400000
## 6     KS   NA d001_h00 0.07714063
```

```r
dim(WINON20_AF@weather)[1] / 365 / 24 # Check - number of regions with the data
```

```
## [1] 24
```

```r
# Wind AF (offshore)
wndf_af20$slice <- datetime2tsdh(wndf_af20$datetime)
wndf_af20
```

```
## # A tibble: 166,440 x 5
## # Groups:   datetime [8,760]
##    datetime            region max_GWh af50m slice
##    <dttm>              <chr>    <dbl> <dbl> <chr>
##  1 2017-01-01 00:30:00 CA      76.0   0.460 d001_h00
##  2 2017-01-01 00:30:00 DE       5     1     d001_h00
##  3 2017-01-01 00:30:00 IL       0.265 0.053 d001_h00
##  4 2017-01-01 00:30:00 MA      72.9   0.972 d001_h00
##  5 2017-01-01 00:30:00 MD      10     1     d001_h00
##  6 2017-01-01 00:30:00 ME      47.8   0.956 d001_h00
##  7 2017-01-01 00:30:00 MI      55.6   0.463 d001_h00
##  8 2017-01-01 00:30:00 MN       5     1     d001_h00
##  9 2017-01-01 00:30:00 NC      59.4   0.517 d001_h00
## 10 2017-01-01 00:30:00 NJ      29.1   0.972 d001_h00
## # ... with 166,430 more rows
```

```r
dim(wndf_af20)[1] / 365/24
```

```
## [1] 19
```

```r
ii <- rep(T, dim(wndf_af20)[1]) #
summary(wndf_af20$af50m)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.04948 0.18450 0.27774 0.42129 1.00000
```

```r
wndf_af20$region <- as.character(wndf_af20$region)
WINOF20_AF <- newWeather("WINOF20_AF",
                        description = "Offshore wind potential af20",
                        # unit = "kWh/kWh_max",
                        region = unique(wndf_af20$region),
                        slice = "HOUR",
                        weather = data.frame(
                          region = as.character(wndf_af20$region[ii]),
                          # year = 2010,
                          slice = wndf_af20$slice[ii],
                          wval = wndf_af20$af50m[ii]
                        ))
head(WINOF20_AF@weather)
```

```
##   region year    slice      wval
## 1     CA   NA d001_h00 0.4603636
## 2     DE   NA d001_h00 1.0000000
## 3     IL   NA d001_h00 0.0530000
## 4     MA   NA d001_h00 0.9717333
## 5     MD   NA d001_h00 1.0000000
## 6     ME   NA d001_h00 0.9564000
```

```r
dim(WINOF20_AF@weather)[1] / 365 / 24
```

```
## [1] 19
```

Supply

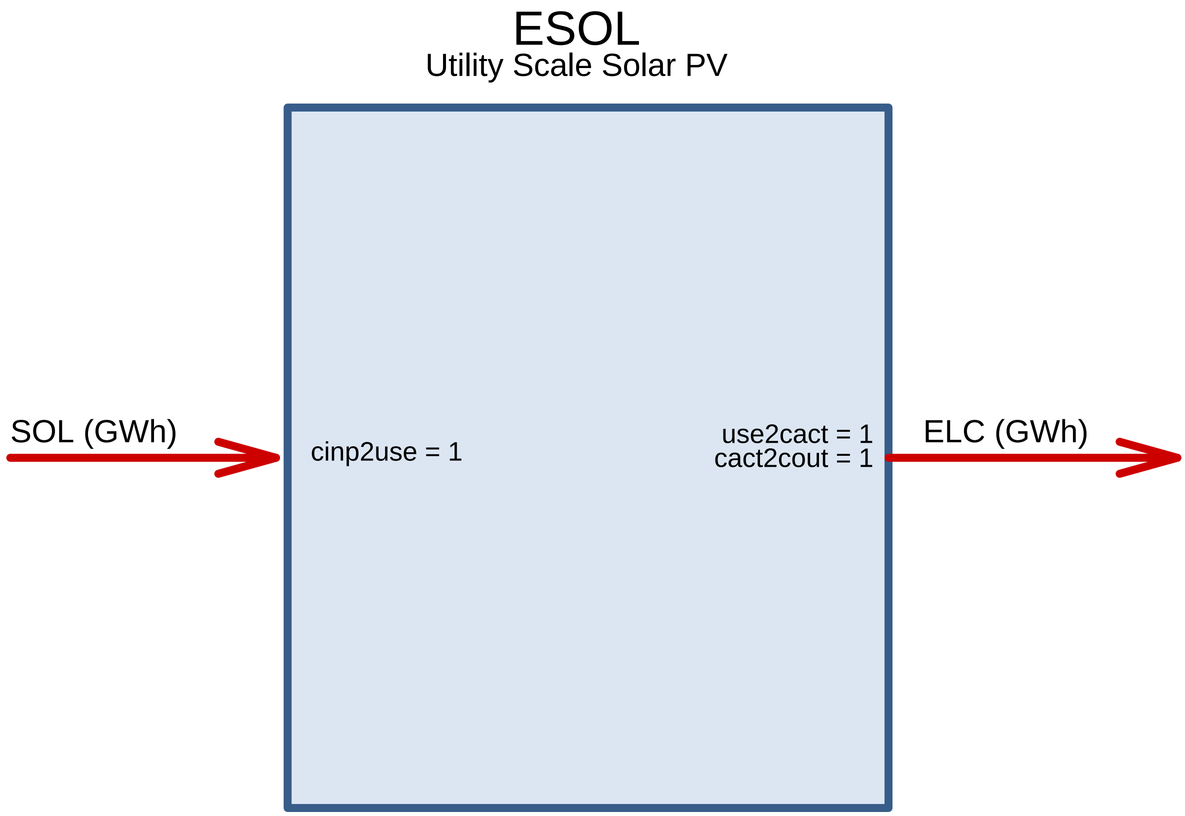
Declaration of resources (upstream technologies) in the model.
Here we use only solar and wind energy.

```
RES_SOL <- newSupply(
  name = "RES_SOL",
  description = "Terrestrial solar radiation - maximum potential",
  commodity = "SOL",
  unit = "GWh",
# Weather factors could be used to regulate hourly supply of the resources.
# Though to reduce the model dimension, it is enough to use
# weather factors in technologies.
  # weather = data.frame(
  #   weather = c("SOLAR_AF"),
  #   wava.up =  c(1)
  #   ),
  availability = list(
    # region = dh$region,
    # slice = dh$slice365,
    ava.up = 1e10 # Max available resource in hour, i.e. no limit by now
  ),
  slice = "HOUR"
)

RES_WIN <- newSupply(
  name = "RES_WIN",
  description = "Onshore wind - maximum potential",
  commodity = "WIN",
  region = unique(wnd_af20$region),
  unit = "GWh",
  # weather = data.frame(
  #   weather = c("SOLAR_AF"),
  #   wava.up =  c(1)
  #   ),
  availability = list(
    region = wnd_af20$region,
    slice = wnd_af20$slice,
# here is an alternative (equivalent) way to use weather factors in supply
    ava.up = wnd_af20$max_GWh # Max available resource in hour
  ),
  slice = "HOUR"
)

RES_WFF <- newSupply(
  name = "RES_WFF",
  description = "Offshore wind - maximum potential",
  commodity = "WFF",
  region = unique(wndf_af20$region),
  unit = "GWh",
  # weather = data.frame(
  #   weather = c("SOLAR_AF"),
  #   wava.up =  c(1)
  #   ),
```

```
  availability = list(
    region = wndf_af20$region,
    slice = wndf_af20$slice,
    ava.up = wndf_af20$max_GWh # Max available resource in hour
  ),
  slice = "HOUR"
)
```

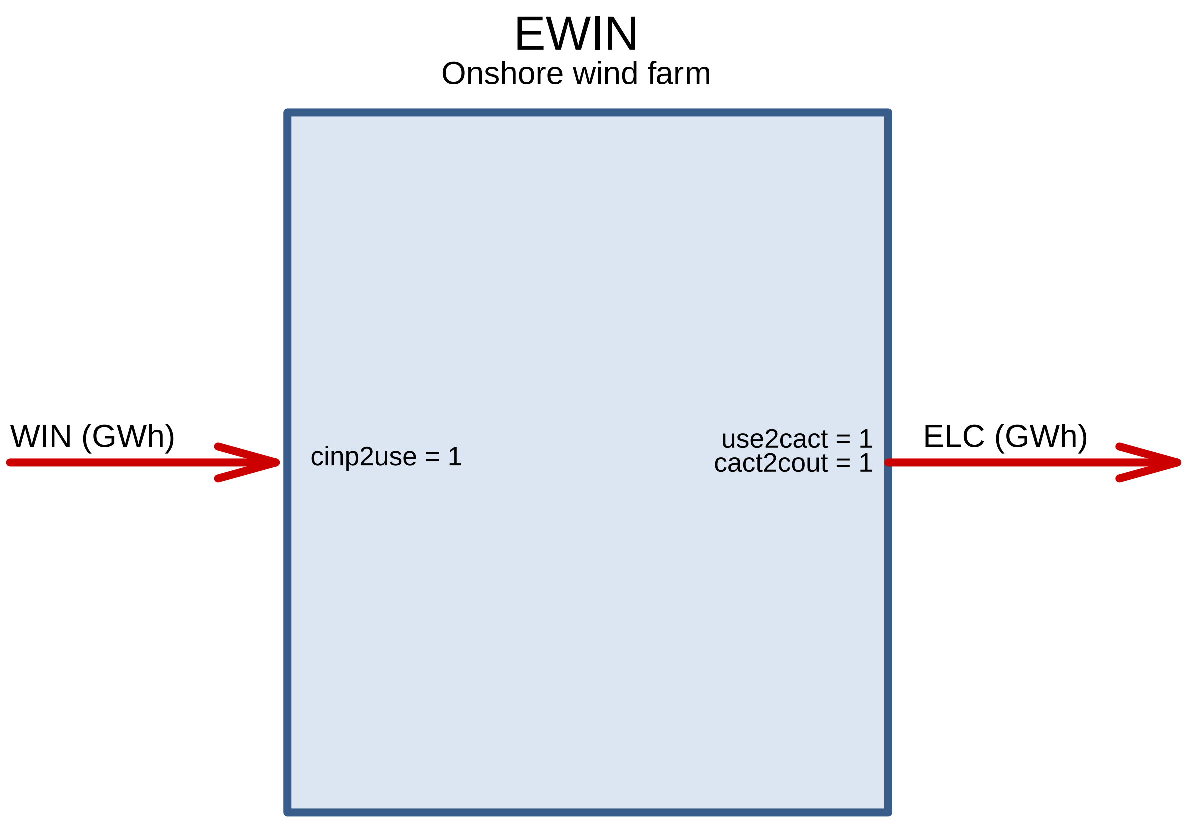
Power generating technologies

```
ESOL <- newTechnology(
  name = "ESOL",
  description = "Utility Scale Solar PV",
  # region = "AZ",
  input = list(
    comm = "SOL",
    unit = "GWh"
  ),
  output = list(
    comm = "ELC",
    unit = "GWh"
  ),
  cap2act = 365*24,
  af = list(
    af.fx = 1 # forcing output when resource is available
  ),
  weather = list(
    weather = c("SOLAR_AF"),
    waf.fx = 1 # weather factor (multiplier) will be applied to af.fx
  ),
  fixom = list(
    fixom = 10 # assumed, 1% of investment costs a year
  ),
  invcost = list(
    # Assuming 1$/Watt
    # https://www.nrel.gov/news/press/2017/nrel-report-utility-scale-solar-pv-system-cost-fell-last-yea
    invcost = 1000 # convert("USD/W", "MUSD/GW", 1)
  ),
  start = list(
    start = 2017
  ),
  olife = list(
    olife = 25
  )
)
draw(ESOL)
```



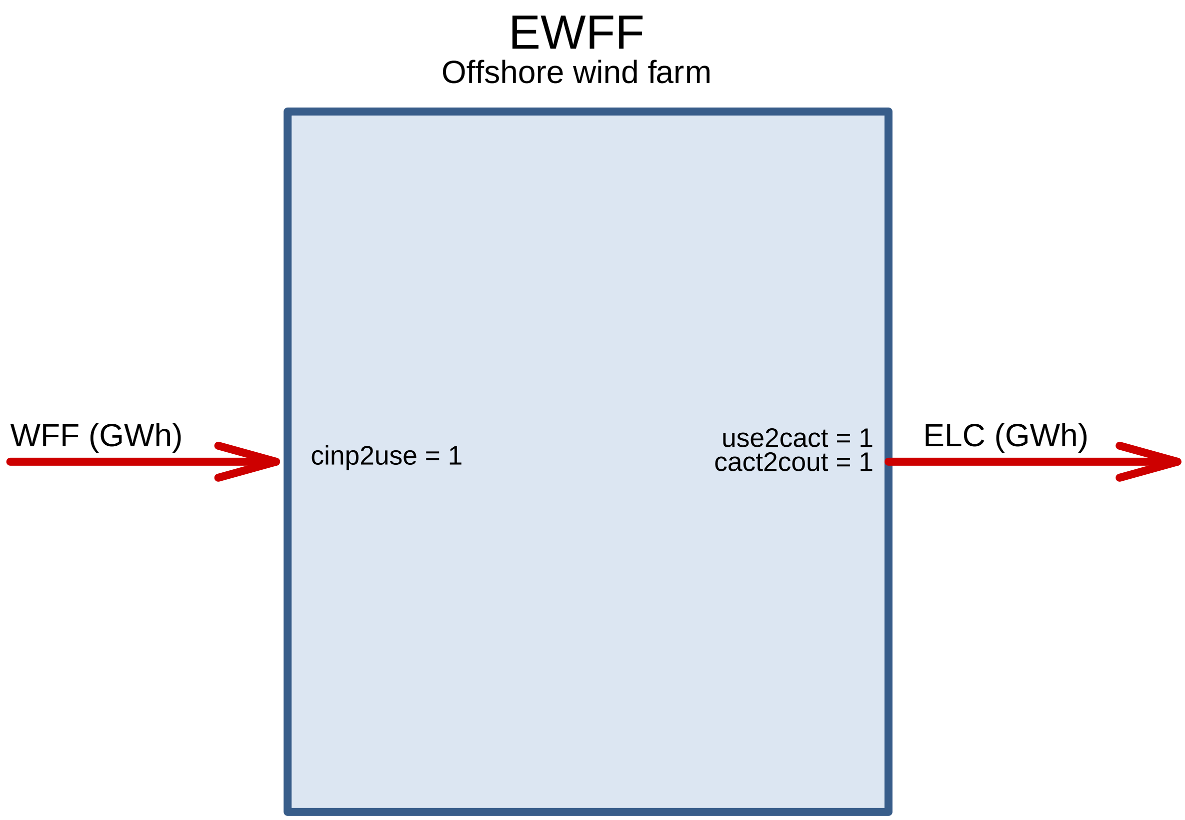

```
EWIN <- newTechnology(
  name = "EWIN",
  description = "Onshore wind farm",
  input = list(
    comm = "WIN",
    unit = "GWh"
  ),
  output = list(
    comm = "ELC",
    unit = "GWh"
  ),
  cap2act = 365*24,
  af = list(
    af.fx = 1 # forcing output when resource is available
  ),
  weather = list(
    weather = c("WINON20_AF"),
    waf.fx = c(1)
    # waf.up = c(1) #
  ),
  fixom = list(
    fixom = 15 # Assumed, 1% a year
  ),
  invcost = list(
    # Assuming 1.5$/Watt
    # https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2018/Jan/IRENA_2017_Power_Costs_2018
```

```
    invcost = 1500 #
  ),
  start = list(
    start = 2017
  ),
  olife = list(
    olife = 25
  )
)
draw(EWIN)
```




```
EWFF <- newTechnology(
  name = "EWFF",
  description = "Offshore wind farm",
  # region = "AZ",
  input = list(
    comm = "WFF",
    unit = "GWh"
  ),
  output = list(
    comm = "ELC",
    unit = "GWh"
  ),
  cap2act = 365*24,
  af = list(
    af.fx = 1 # forcing output when resource is available
```

```
  ),
  weather = list(
    weather = c("WINOF20_AF"),
    waf.up = c(1) #
  ),
  fixom = list(
    fixom = 45 # Assumed, 1% a year
  ),
  invcost = list(
    # Assuming 4.5$/Watt
    # https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2018/Jan/IRENA_2017_Power_Costs_2018
    invcost = 4500 # convert("USD/W", "MUSD/GW", 1)
  ),
  start = list(
    start = 2017
  ),
  olife = list(
    olife = 25
  )
)
draw(EWFF)
```



Storage technologies

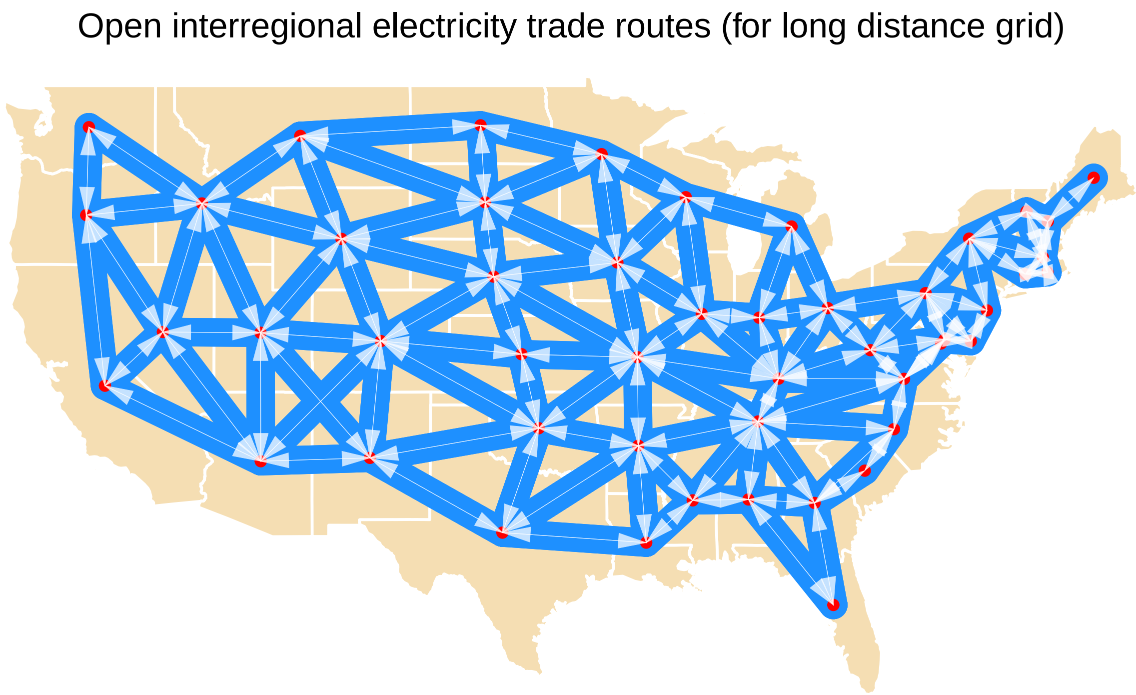
```r
# STGELCH Hourly - within 24h ####
STGELCH <- newStorage(
  name = 'STGELCH',
  commodity = 'ELC',
  description = "Generic grid-integrated intraday storage (battery)",
  slice = 'HOUR',
  cap2stg = 1, #
  olife = list(olife = 10),
  invcost = list(invcost = convert("USD/kWh", "MUSD/GWh", 200)), # See IRENA 2030 (from 77 to 574, p.77
  seff = data.frame(
    inpeff = 0.8, # assumed efficiency of charging
    stgeff = 0.9 # assumed efficiency of storing energy (annual)
    # outeff = 1 # discharge efficiency
    )
)

# STGELCD Daily - within 365 days ####
STGELCD <- newStorage(
  name = 'STGELCD',
  commodity = 'ELC',
  description = "Long term energy storage systems (ala P2X)",
  # cap2act = ECOA@cap2act,
  slice = 'YDAY',
  cap2stg = 1, # if in PJ, convert("GWh", "PJ"),
  olife = list(olife = 25),
  # invcost = list(invcost = convert("USD/MWh", "MRMB/PJ", 180)), # Assumption
  invcost = 50, # USD/kWh, assumption
  seff = data.frame(
    inpeff = 0.7 # overall (roundtrip) efficiency
    # stgeff = .9
    ),
  varom = list(
    # Assuming high operational costs, adding ~ 5 cents/kWh
    inpcost = convert("USD/kWh", "MUSD/GWh", .05)
    )
)
```

Interregional ELC electrical grid (simplified)

```r
# Trade matrix
{
if (F) { # Load trade matrix from file
  trd_xl <- readxl::read_excel("data/trade_matrix01.xlsx",
                               range = "A1:AX50")

  trd_nbr <- as.matrix(trd_xl[,-1])
  rownames(trd_nbr) <- trd_xl$region
} else {
  # Creating neighbors-trade matrix
```

```r
  trd_nbr <- matrix(rep(NA, nreg_in_gis*nreg_in_gis),
                    nrow = nreg_in_gis,
                    dimnames = list(reg_names_in_gis, reg_names_in_gis))
  head(trd_nbr)

  for (i in 1:length(reg_names)) {
    trd_nbr[i, nbr[[i]]] <- 1
    trd_nbr[nbr[[i]], i] <- 1
  }
  dim(trd_nbr)
  ii <- reg_names_in_gis %in% reg_names
  trd_nbr <- trd_nbr[ii, ii]
  dim(trd_nbr)
}
head(trd_nbr)
# write.csv(trd_nbr, file = "data/trade_matrix.csv")
trd_nbr[trd_nbr == 0] <- NA

trd_dt <- as.data.frame.table(trd_nbr, stringsAsFactors = F)
trd_dt <- trd_dt[!is.na(trd_dt$Freq),]
head(trd_dt)
dim(trd_dt)
trd_dt <- dplyr::distinct(trd_dt)
dim(trd_dt)
names(trd_dt) <- c("src", "dst", "trd")
head(trd_dt)

# Map region flows
trd_map <- left_join(trd_dt, reg_centers[,1:3], by = c("src" = "region"))
trd_map <- left_join(trd_map, reg_centers[,1:3], by = c("dst" = "region"))
trd_map <- trd_map %>%
  rename(xsrc = x.x, ysrc = y.x,
         xdst = x.y, ydst = y.y)
trd_map <- as_tibble(trd_map)

# a <- value
trd_flows_map <-
ggplot(data = usa49r) +
    geom_polygon(aes(x = long, y = lat, group = group), fill = "wheat",
                 colour = "white", alpha = 1, size = .5) + # aes fill = id,
    coord_fixed(1.3) +
    guides(fill=FALSE) +  # do this to leave off the color legend
    theme_void() + labs(title = "Open interregional electricity trade routes (for long distance grid)")
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5)) +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst),
                 data = trd_map, inherit.aes = FALSE, size = 5,
                 alpha = 1, colour = "dodgerblue", lineend = "round", show.legend = T) +
    geom_point(data = reg_centers, aes(x, y), colour = "red") +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst),
                 data = trd_map, inherit.aes = FALSE, size = .1,
             arrow = arrow(type = "closed", angle = 15,
                           length = unit(0.15, "inches")),
```

```
             colour = "white", alpha = 0.75,
             lineend = "butt", linejoin = "mitre", show.legend = T) # , name = "Trade, PJ"
trd_flows_map
# ggsave("fig/trd_flows_map.pdf", trd_flows_map, device = "pdf")
}
```




```
# Calculate distance between regions centers:
labpt <- as.data.frame(get_labpt_spdf(usa49reg))
rownames(labpt) <- labpt$region

trd_dt$distance_km <- 0.

for (i in 1:dim(trd_dt)[1]) {
  rg_dst <- trd_dt$dst[i]
  rg_src <- trd_dt$src[i]
  lab_dst <- labpt[rg_dst, 2:3]
  lab_src <- labpt[rg_src, 2:3]
  trd_dt$distance_km[i] <- raster::pointDistance(
    lab_src, lab_dst, T)/1e3
}

# Assume losses 2% per 1000 km
# trd_dt$losses <- round(trd_dt$distance_km / 1e3 * 0.02, 4)
# Assume losses 1% per 1000 km
trd_dt$losses <- round(trd_dt$distance_km / 1e3 * 0.01, 4)
# Converting losses to costs, based on 6 UScent/kWh
```

```r
# trd_dt$cost <- round(trd_dt$losses * convert("USD/kWh", "MRMB/PJ", .06), 3)
# Assumption - additional to ELC losses costs per 1000 km
# trd_dt$cost <- round(trd_dt$losses * convert("USD/kWh", "MRMB/PJ", .06), 3) # Assuming 6 US cents per
# trd_dt$cost <- round(trd_dt$distance_km / 1e3 *
#                        convert("USD/kWh", "MUSD/GWh", .012), 3) # Assuming 1.2 US cents per kWh for t
trd_dt$cost <- round(trd_dt$distance_km / 1e3 *
                       convert("USD/kWh", "MUSD/GWh", .001), 5) # Assuming 0.1 US cents per kWh per 100
trd_dt <- as_tibble(trd_dt)

#Repository for trades, related technologies and commodities
TRD_ELC_ALL <- newRepository(name = "TRD_ELC_ALL")
# Select regions with non-zero number of neighbours
# nbr0 <- sapply(nbr, function(x) length(x) > 0 && x > 0)
#reg_names0 <- reg_names
# reg_id <- 1:length(reg_names)
# for (i in reg_id) {
for (src_nm in unique(trd_dt$src)) {
# The loop creates trade-objects for every region with neighbors,
# as well as required commodities, and technologies to model the trade
  # if (nbr0[i]) {
    # Names
    # src_nm <- reg_names[i]    # Source region name
    # dst_nms <- reg_names[nbr[[i]]]  # Destination region name
  ii <- trd_dt$src %in% src_nm
  dst_nms <- trd_dt$dst[ii]
    # n_dst <- length(nbr[[i]]) # Number of destimation regions
    # n_src <- n_dst  # Assuming simmetric trade oppennes with neighbor regions
    trd_nm <- paste0("TRD_ELC_", src_nm)  # Trade object name
    # cmd_nm <- paste0("ELC_", src_nm)  # Commodity name, exporting to other regions
    # cmd_nm_nbr <- paste0("ELC_", dst_nms) # Commodities name, importing from other regions
    cmd_nm <- "ELC"
    cmd_nm_nbr <- "ELC"
    # tch_src_nm <- paste0("TELC_", src_nm, "_SRC") # Technology name - transportation to source region
    # tch_dst_nm <- paste0("TELC_", src_nm, "_DST") # Technology name - transportation from other regio

    trd <- newTrade(trd_nm,
                    commodity = cmd_nm,
                    source = src_nm,
                    destination = dst_nms,
                    trade = data.frame(
                      src = src_nm,
                      dst = dst_nms,
                      ava.up = convert("GWh", "GWh", 100), # Maximum capacity per route in GW
                      # cost = rep(convert("USD/kWh", "MRMB/PJ", 0.015), n_dst)
                      cost = trd_dt$cost[ii]
                    )#,
                    # aux = list(acomm = "ELC"),
                    # aeff = data.frame(
                    #   acomm = "ELC",
                    #   csrc2ainp = trd_dt$losses[ii]) # 1% dispatch losses per 1K km
                    )
    TRD_ELC_ALL <- add(TRD_ELC_ALL, trd)
  # }
```

```r
}

names(TRD_ELC_ALL@data)
```

```
##  [1] "TRD_ELC_ID" "TRD_ELC_OR" "TRD_ELC_ND" "TRD_ELC_SD" "TRD_ELC_WY"
##  [6] "TRD_ELC_NH" "TRD_ELC_MT" "TRD_ELC_MN" "TRD_ELC_IA" "TRD_ELC_NE"
## [11] "TRD_ELC_UT" "TRD_ELC_CO" "TRD_ELC_IL" "TRD_ELC_MI" "TRD_ELC_WA"
## [16] "TRD_ELC_NV" "TRD_ELC_MA" "TRD_ELC_NY" "TRD_ELC_WI" "TRD_ELC_CA"
## [21] "TRD_ELC_ME" "TRD_ELC_VT" "TRD_ELC_MO" "TRD_ELC_CT" "TRD_ELC_RI"
## [26] "TRD_ELC_KS" "TRD_ELC_PA" "TRD_ELC_NJ" "TRD_ELC_OH" "TRD_ELC_DE"
## [31] "TRD_ELC_WV" "TRD_ELC_MD" "TRD_ELC_KY" "TRD_ELC_AZ" "TRD_ELC_NM"
## [36] "TRD_ELC_IN" "TRD_ELC_VA" "TRD_ELC_DC" "TRD_ELC_OK" "TRD_ELC_TN"
## [41] "TRD_ELC_NC" "TRD_ELC_AR" "TRD_ELC_TX" "TRD_ELC_GA" "TRD_ELC_SC"
## [46] "TRD_ELC_AL" "TRD_ELC_MS" "TRD_ELC_LA" "TRD_ELC_FL"
```

```r
TRD_ELC_ALL@data$TRD_ELC_ID
```

```
## An object of class "trade"
## Slot "name":
## [1] "TRD_ELC_ID"
##
## Slot "description":
## [1] ""
##
## Slot "commodity":
## [1] "ELC"
##
## Slot "source":
## [1] "ID"
##
## Slot "destination":
## [1] "WA" "MT" "WY" "OR" "NV" "UT"
##
## Slot "trade":
##   src dst year slice ava.up ava.fx ava.lo    cost markup teff
## 1  ID  WA   NA  <NA>    100     NA     NA 0.00056     NA   NA
## 2  ID  MT   NA  <NA>    100     NA     NA 0.00049     NA   NA
## 3  ID  WY   NA  <NA>    100     NA     NA 0.00059     NA   NA
## 4  ID  OR   NA  <NA>    100     NA     NA 0.00047     NA   NA
## 5  ID  NV   NA  <NA>    100     NA     NA 0.00058     NA   NA
## 6  ID  UT   NA  <NA>    100     NA     NA 0.00061     NA   NA
##
## Slot "aux":
## [1] acomm unit
## <0 rows> (or 0-length row.names)
##
## Slot "aeff":
## [1] acomm     src       dst       year      slice     csrc2aout csrc2ainp
## [8] cdst2aout cdst2ainp
## <0 rows> (or 0-length row.names)
##
## Slot "invcost":
## [1] src     dst     year    invcost
## <0 rows> (or 0-length row.names)
```

```
##
## Slot "olife":
## [1] src   dst   olife
## <0 rows> (or 0-length row.names)
##
## Slot "start":
## [1] src   dst   start
## <0 rows> (or 0-length row.names)
##
## Slot "end":
## [1] src dst end
## <0 rows> (or 0-length row.names)
##
## Slot "stock":
## [1] src   dst   year  stock
## <0 rows> (or 0-length row.names)
##
## Slot "capacityVariable":
## [1] FALSE
##
## Slot "GIS":
## NULL
##
## Slot "cap2act":
## [1] 1
##
## Slot "bidirectional":
## [1] FALSE
##
## Slot "misc":
## list()
##
## Slot ".S3Class":
## [1] "trade"
```

```
# Do the same for UHV commodity, simply copying and replacing
TRD_UHV_ALL <- TRD_ELC_ALL
TRD_UHV_ALL@data <- lapply(TRD_UHV_ALL@data, function(x) {
  x@name <- sub("_ELC_", "_UHV_", x@name)
  x@commodity <- "UHV"
  x
})
names(TRD_UHV_ALL@data) <- sub("_ELC_", "_UHV_", names(TRD_UHV_ALL@data))
names(TRD_ELC_ALL@data)
```

```
##  [1] "TRD_ELC_ID" "TRD_ELC_OR" "TRD_ELC_ND" "TRD_ELC_SD" "TRD_ELC_WY"
##  [6] "TRD_ELC_NH" "TRD_ELC_MT" "TRD_ELC_MN" "TRD_ELC_IA" "TRD_ELC_NE"
## [11] "TRD_ELC_UT" "TRD_ELC_CO" "TRD_ELC_IL" "TRD_ELC_MI" "TRD_ELC_WA"
## [16] "TRD_ELC_NV" "TRD_ELC_MA" "TRD_ELC_NY" "TRD_ELC_WI" "TRD_ELC_CA"
## [21] "TRD_ELC_ME" "TRD_ELC_VT" "TRD_ELC_MO" "TRD_ELC_CT" "TRD_ELC_RI"
## [26] "TRD_ELC_KS" "TRD_ELC_PA" "TRD_ELC_NJ" "TRD_ELC_OH" "TRD_ELC_DE"
## [31] "TRD_ELC_WV" "TRD_ELC_MD" "TRD_ELC_KY" "TRD_ELC_AZ" "TRD_ELC_NM"
## [36] "TRD_ELC_IN" "TRD_ELC_VA" "TRD_ELC_DC" "TRD_ELC_OK" "TRD_ELC_TN"
## [41] "TRD_ELC_NC" "TRD_ELC_AR" "TRD_ELC_TX" "TRD_ELC_GA" "TRD_ELC_SC"
## [46] "TRD_ELC_AL" "TRD_ELC_MS" "TRD_ELC_LA" "TRD_ELC_FL"
```

```r
TRD_UHV_ALL@data$TRD_UHV_ID
```

```
## An object of class "trade"
## Slot "name":
## [1] "TRD_UHV_ID"
##
## Slot "description":
## [1] ""
##
## Slot "commodity":
## [1] "UHV"
##
## Slot "source":
## [1] "ID"
##
## Slot "destination":
## [1] "WA" "MT" "WY" "OR" "NV" "UT"
##
## Slot "trade":
##   src dst year slice ava.up ava.fx ava.lo    cost markup teff
## 1  ID  WA   NA  <NA>    100     NA     NA 0.00056     NA   NA
## 2  ID  MT   NA  <NA>    100     NA     NA 0.00049     NA   NA
## 3  ID  WY   NA  <NA>    100     NA     NA 0.00059     NA   NA
## 4  ID  OR   NA  <NA>    100     NA     NA 0.00047     NA   NA
## 5  ID  NV   NA  <NA>    100     NA     NA 0.00058     NA   NA
## 6  ID  UT   NA  <NA>    100     NA     NA 0.00061     NA   NA
##
## Slot "aux":
## [1] acomm unit
## <0 rows> (or 0-length row.names)
##
## Slot "aeff":
## [1] acomm     src       dst       year      slice     csrc2aout csrc2ainp
## [8] cdst2aout cdst2ainp
## <0 rows> (or 0-length row.names)
##
## Slot "invcost":
## [1] src     dst     year    invcost
## <0 rows> (or 0-length row.names)
##
## Slot "olife":
## [1] src   dst   olife
## <0 rows> (or 0-length row.names)
##
## Slot "start":
## [1] src   dst   start
## <0 rows> (or 0-length row.names)
##
## Slot "end":
## [1] src dst end
## <0 rows> (or 0-length row.names)
##
## Slot "stock":
## [1] src   dst   year  stock
```

```
## <0 rows> (or 0-length row.names)
##
## Slot "capacityVariable":
## [1] FALSE
##
## Slot "GIS":
## NULL
##
## Slot "cap2act":
## [1] 1
##
## Slot "bidirectional":
## [1] FALSE
##
## Slot "misc":
## list()
##
## Slot ".S3Class":
## [1] "trade"
```

Interregional UHV electrical grid

(not tested yet)

```
# Propose trade matrix between regions
{
if (T) {
  # Load trade matrix from your file
  trd_xl <- readxl::read_excel("data/trade_matrix01.xlsx",
                               range = "A1:AX50")

  trd_nbr <- as.matrix(trd_xl[,-1])
  trd_nbr[lower.tri(trd_nbr)] <- NA
  rownames(trd_nbr) <- trd_xl$region
} else {
  # Or create trade matrix for all neighbour regions
  trd_nbr <- matrix(rep(NA, nreg_in_gis*nreg_in_gis),
                    nrow = nreg_in_gis,
                    dimnames = list(reg_names_in_gis, reg_names_in_gis))
  head(trd_nbr)

  for (i in 1:length(reg_names)) {
    trd_nbr[i, nbr[[i]]] <- 1
    trd_nbr[nbr[[i]], i] <- 1
  }
  dim(trd_nbr)
  summary(!is.na(c(trd_nbr)))
  ii <- reg_names_in_gis %in% reg_names
  trd_nbr <- trd_nbr[ii, ii]
  dim(trd_nbr)
  # For bidirectional trade, keep only one direction
  trd_nbr[lower.tri(trd_nbr)] <- NA
  summary(!is.na(c(trd_nbr)))
```

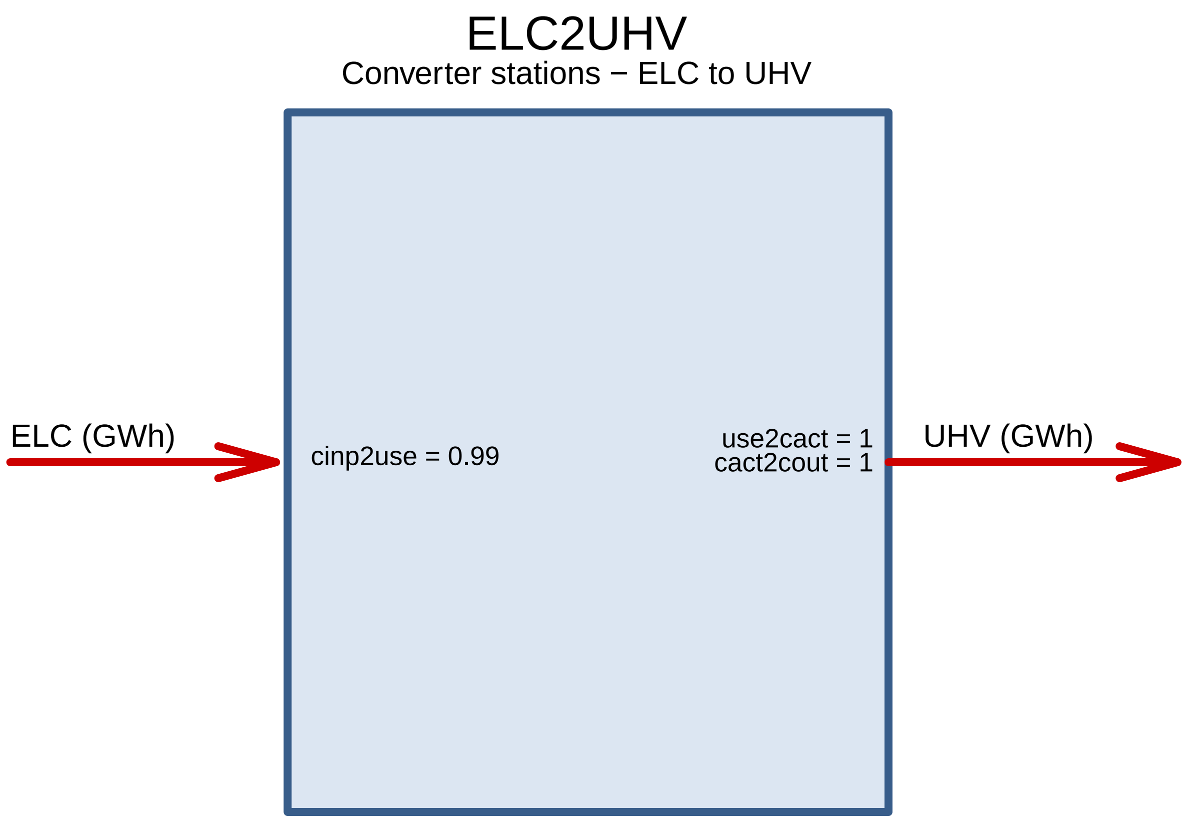
```r
}
head(trd_nbr)
# write.csv(trd_nbr, file = "data/trade_matrix.csv")
trd_nbr[trd_nbr == 0] <- NA

# Convert the matrix to data.frame (table) format
trd_dt <- as.data.frame.table(trd_nbr, stringsAsFactors = F)
trd_dt <- trd_dt[!is.na(trd_dt$Freq),]
head(trd_dt)
dim(trd_dt)
trd_dt <- dplyr::distinct(trd_dt)
dim(trd_dt)
names(trd_dt) <- c("src", "dst", "trd")
head(trd_dt)

# Map region flows
trd_map <- left_join(trd_dt, reg_centers[,1:3], by = c("src" = "region"))
trd_map <- left_join(trd_map, reg_centers[,1:3], by = c("dst" = "region"))
trd_map <- trd_map %>%
  rename(xsrc = x.x, ysrc = y.x,
         xdst = x.y, ydst = y.y)
trd_map <- as_tibble(trd_map)

# a <- value
trd_flows_map <-
ggplot(data = usa49r) +
    geom_polygon(aes(x = long, y = lat, group = group), fill = "wheat",
                 colour = "white", alpha = 1, size = .5) + # aes fill = id,
    coord_fixed(1.3) +
    guides(fill=FALSE) +  # do this to leave off the color legend
    theme_void() + labs(title = "Open interregional electricity trade routes (long distance grid)")  +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5)) +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst),
                 data = trd_map, inherit.aes = FALSE, size = 5,
                 alpha = 1, colour = "grey", lineend = "round", show.legend = T) +
    geom_point(data = reg_centers, aes(x, y), colour = "red") +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst),
                 data = trd_map, inherit.aes = FALSE, size = .1,
             # arrow = arrow(type = "closed", angle = 15,
             #               length = unit(0.15, "inches")),
             colour = "darkgrey", alpha = 1,
             lineend = "butt", linejoin = "mitre", show.legend = T) # , name = "Trade, PJ"
trd_flows_map
# ggsave("fig/trd_flows_map.pdf", trd_flows_map, device = "pdf")
}
# Calculate distance between regions centers:
labpt <- get_labpt_spdf(usa49reg)
rownames(labpt) <- labpt$region

# Estimate grid length, losses, costs
trd_dt$distance_km <- 0.
for (i in 1:dim(trd_dt)[1]) {
```

```r
  rg_dst <- trd_dt$dst[i]
  rg_src <- trd_dt$src[i]
  lab_dst <- labpt[rg_dst, 2:3]
  lab_src <- labpt[rg_src, 2:3]
  trd_dt$distance_km[i] <- raster::pointDistance(
    lab_src, lab_dst, T)/1e3
}

# Assume losses 2% per 1000 km
# trd_dt$losses <- round(trd_dt$distance_km / 1e3 * 0.02, 4)
# Assume losses 1% per 1000 km (UHVDC)
trd_dt$losses <- round(trd_dt$distance_km / 1e3 * 0.01, 4)
trd_dt$teff <- 1 - trd_dt$losses
# Assumption based on ABB's 4000-8000 MUSD per 12GW UHVDC, 2000km, 1-5% system losses
# i.e. ~$160-333 MUSD/1000km per 1GW of the total system
# assuming ~$200 MUSD/1000km per 1GW of power line,
# and $50 MUSD/GW for converter stations on each end
trd_dt$invcost <- round(trd_dt$distance_km / 1e3 * 200) # MUSD/GW of 1000km UHVDC
trd_dt <- as_tibble(trd_dt)

# Define trade object for every route,
# store in a repository object
TRBD_UHV_NEI <- newRepository(name = "TRBD_UHV_NEI")
for (i in 1:dim(trd_dt)[1]) {
  src <- trd_dt$src[i]
  dst <- trd_dt$dst[i]
  trd_nm <- paste0("TRBD_UHV_", src, "_", dst)  # Trade object name
  cmd_nm <- "UHV"
  # Trade class for every route
  trd <- newTrade(trd_nm,
                  commodity = cmd_nm,
                  source = c(src, dst),
                  destination = c(src, dst),
                  trade = data.frame(
                    src = src,
                    dst = dst,
                    # Maximum capacity per route in GW
                    ava.up = convert("GWh", "GWh", 120),
                    teff = trd_dt$teff[i] # trade losses
                    # cost = trd_dt$cost[i]  # trade costs
                    # markup = trd_dt$cost[i] # and/or markup
                    ),
                  #!!! New stuff - testing
                  capacityVariable = TRUE, # The trade route has capacity (not just flow) and can be en
                  bidirectional = TRUE, #
                  invcost = data.frame(
                    src = src,
                    dst = dst,
                    year = 2010,
                    invcost = trd_dt$distance_km[i] / 1e3 * 250 #
                  ),
                  olife = data.frame(
                    olife = 80
```

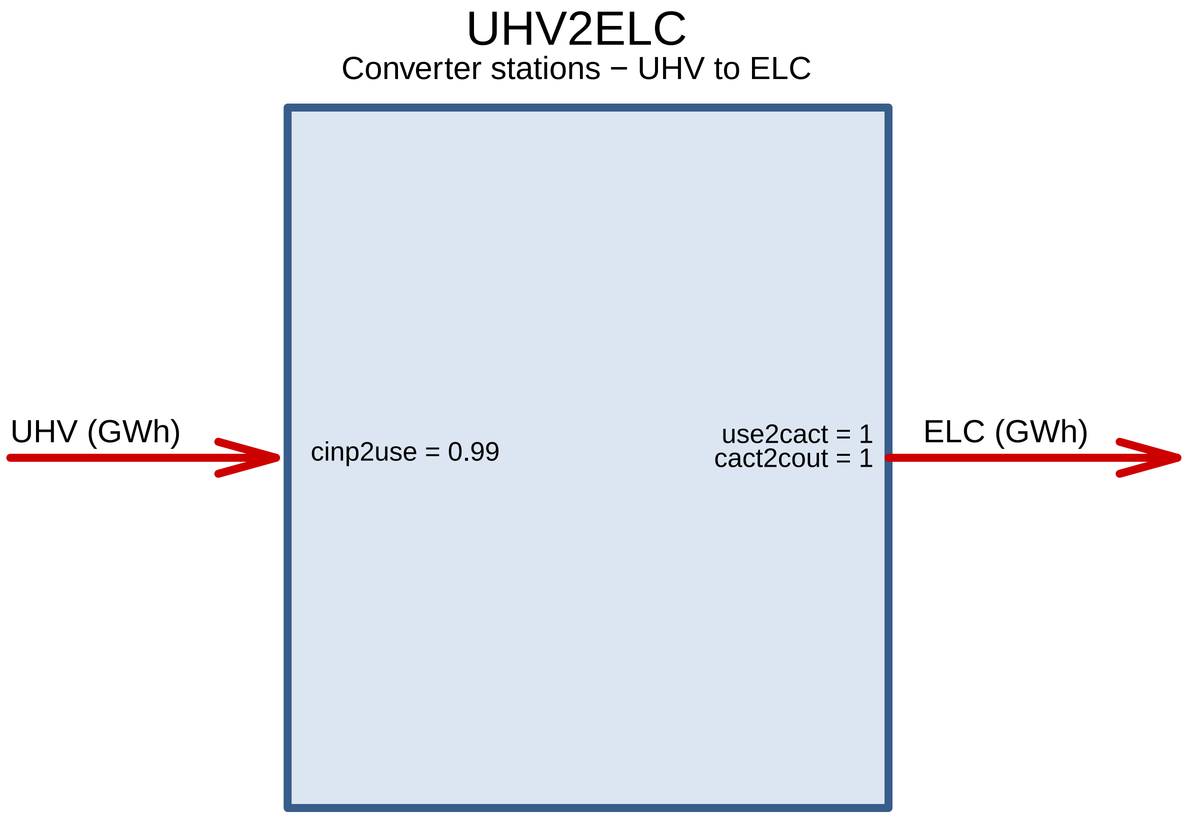
```
                  ),
                  cap2act = convert("GWh", "GWh", 24*365)
                  )

  TRBD_UHV_NEI <- add(TRBD_UHV_NEI, trd)
}
names(TRBD_UHV_NEI@data)
# TRBD_UHV_NEI@data[[1]]
```

Inverter and rectifier stations

```
# Parameters assumed or calibrated based on
# ABB's 4000-8000 MUSD per 12GW, 2000km, 1-5%
# and other sources

ELC2UHV <- newTechnology(
  name = "ELC2UHV",
  description = "Converter stations - ELC to UHV",
  input = list(
    comm = "ELC",
    unit = "GWh"
    # unit = "PJ"
  ),
  output = list(
    comm = "UHV",
    unit = "GWh"
    # unit = "PJ"
  ),
  # cap2act = 31.536,
  cap2act = 24*365,
  ceff = list(
    comm = "ELC",
    cinp2use = .99 # see also Siemens -- 3% total losses for HVDC/1000km
  ),
  slice = "HOUR",
  invcost = list(
    invcost = 50  #
  ),
  olife = 20 # assumed
)
draw(ELC2UHV)
```




```r
UHV2ELC <- newTechnology(
  name = "UHV2ELC",
  description = "Converter stations - UHV to ELC",
  input = list(
    comm = "UHV",
    unit = "GWh"
    # unit = "PJ"
  ),
  output = list(
    comm = "ELC",
    unit = "GWh"
    # unit = "PJ"
  ),
  # cap2act = 31.536,
  cap2act = 24*365,
  ceff = list(
    comm = "UHV",
    cinp2use = .99
  ),
  slice = "HOUR",
  invcost = list(
    invcost = 50 # Assumed, based on ABB data
  ),
  olife = 20 # assumed
)
draw(UHV2ELC)
```



Trade with the rest of the world (ROW)

Currently used to estimate curtailments and the system failure to meet the demand.

```
EEXP <- newExport(
  name = "EEXP",
  description = "Supply curtalments (artificial export to capture excessive ELC production by renewables
  commodity = "ELC",
  exp = list(
    price = convert("USD/kWh", "MUSD/GWh", .01/1000) # 1/1000 cents per kWh
  )
)

EIMP <- newImport(
  name = "EIMP",
  description = "Demand curtailments, electricity import at high price (to identify shortages)",
  commodity = "ELC",
  imp = list(
    price = convert("USD/kWh", "MUSD/GWh", 10) # USD per kWh, marginal price
  )
)
```

Model #1

```r
# Repository with all the data-objects
reps <- add(
  newRepository('main_repository'),
  # Commodities
  ELC, SOL, WIN, WFF, # UHV,
  # Resources (supply)
  RES_SOL, RES_WIN, RES_WFF,
  # Weather factors
  SOLAR_AF, WINOF20_AF, WINON20_AF,
  # Generating technologies
  ESOL, EWIN, EWFF,
  # Storage
  STGELCH, STGELCD,
  # Simplified interregional ELC trade
  TRD_ELC_ALL,
  # UHV grid
  # TRBD_UHV_NEI,
  # ELC2UHV, UHV2ELC,
  # Export/Import,
  EEXP, EIMP,
  # ELC demand with load curve (24 hours x 365 days)
  DEM_ELC_DH # static
)
length(reps@data)
names(reps@data)

mdl <- newModel(
  name = 'RENBAL',
  description = "Renewables balancing model",
  # debug = data.frame(#comm = "ELC",
  #                    dummyImport = 1e6,
  #                    dummyExport = 1e6),
  region = reg_names,
  discount = 0.05,
  slice = list(ANNUAL = "ANNUAL",
               # MONTH = timeslices$MONTH,
               # HOUR = timeslices$HOUR
               YDAY = timeslices365$YDAY,
               HOUR = timeslices365$HOUR
               ),
  repository = reps,
  GIS = usa49reg,
  early.retirement = F)

head(mdl@sysInfo@slice@levels)
# mdl <- setMilestoneYears(mdl, start = 2015, interval = c(1, 2, 5, 6, 7, rep(10, 3)))
mdl <- setMilestoneYears(mdl, start = 2017, interval = c(1))
mdl@sysInfo@milestone

# (optional) GAMS and CPLEX options
{mdl@misc$includeBeforeSolve <- "
```

```
option iterlim = 3e7;
option reslim = 3e4;
option LP = CPLEX;
option threads=12;
option savepoint=1;
*option bRatio = 0;
$onecho > cplex.opt
freegamsmodel 1
parallelmode -1
lpmethod 6
*advind 1
printoptions 0
names no
*memoryemphasis 1
$offecho
*$exit
energyRt.OptFile = 1;
*execute_loadpoint 'energyRt_p';"
} # GAMS & CPLEX parameters

# Adding the scenario info
{mdl@description <- "
USENSYS, power sector, balancing version,
49 regions, 1 hour resolution,
3 types of renewables,
no initial stock,
exogenous ELC grid."}

# (optional) save GDX file with the solution
{mdl@misc$includeAfterSolve <-
"parameter zModelStat;
zModelStat = energyRt.ModelStat;
execute_unload 'USENSYS_ELC_RENBAL.gdx';"
}

# Workflow to solve the model (recommended for large models):
# 1. Interpolation of parameters
scen_REN <- interpolate(mdl, name = "RENBAL")
# 2. Writing the model code and data in folder
write_model(scen_REN, tmp.dir = "solwork/RENBAL", solver = "GAMS")
# save(scen_REN, file = "solwork/renbal/scen_REN.RData")
# 3. Solving the model
# Alterntively, run GAMS directly in the model folder: 'gams mdl'
solve_model(tmp.dir = "solwork/RENBAL1")
# 4. Read the solution
scen_REN <- read_solution(scen_REN, dir.result = "solwork/RENBAL")
summary(scen_REN)

save(scen_REN, file = "scenarios/scen_RENBAL.RData")
```

Model #2

```
# Repository with all the data-objects
reps <- add(
  newRepository('main_repository'),
  # Commodities
  ELC, SOL, WIN, WFF, UHV,
  # Resources (supply)
  RES_SOL, RES_WIN, RES_WFF,
  # Weather factors
  SOLAR_AF, WINOF20_AF, WINON20_AF,
  # Generating technologies
  ESOL, EWIN, EWFF,
  # Storage
  STGELCH, STGELCD,
  # Simplified interregional ELC trade
  # TRD_ELC_ALL,
  TRD_UHV_ALL,
  # UHV grid
  # TRBD_UHV_NEI,
  ELC2UHV, UHV2ELC,
  # Export/Import,
  EEXP, EIMP,
  # ELC demand with load curve (24 hours x 365 days)
  DEM_ELC_DH # static
)
length(reps@data)
names(reps@data)

mdl <- newModel(
  name = 'RENBAL2',
  description = "Renewables balancing model",
  # debug = data.frame(#comm = "ELC",
  #                    dummyImport = 1e6,
  #                    dummyExport = 1e6),
  region = reg_names,
  discount = 0.05,
  slice = list(ANNUAL = "ANNUAL",
             # MONTH = timeslices$MONTH,
             # HOUR = timeslices$HOUR
             YDAY = timeslices365$YDAY,
             HOUR = timeslices365$HOUR
             ),
  repository = reps,
  GIS = usa49reg,
  early.retirement = F)

head(mdl@sysInfo@slice@levels)
# mdl <- setMilestoneYears(mdl, start = 2015, interval = c(1, 2, 5, 6, 7, rep(10, 3)))
mdl <- setMilestoneYears(mdl, start = 2017, interval = c(1))
mdl@sysInfo@milestone

# (optional) GAMS and CPLEX options
```

```
{mdl@misc$includeBeforeSolve <- "
option iterlim = 3e7;
option reslim = 3e4;
option LP = CPLEX;
option threads=12;
option savepoint=1;
*option bRatio = 0;
$onecho > cplex.opt
freegamsmodel 1
parallelmode -1
lpmethod 6
*advind 1
printoptions 0
names no
*memoryemphasis 1
$offecho
*$exit
energyRt.OptFile = 1;
*execute_loadpoint 'energyRt_p';"
} # GAMS & CPLEX parameters

# Adding the scenario info
{mdl@description <- "
USENSYS, power sector, balancing version,
49 regions, 1 hour resolution,
3 types of renewables,
no initial stock,
exogenous UHV grid,
converter stations."}

# (optional) save GDX file with the solution
{mdl@misc$includeAfterSolve <-
"parameter zModelStat;
zModelStat = energyRt.ModelStat;
execute_unload 'USENSYS_ELC_RENBAL2.gdx';"
}

# Workflow to solve the model (recommended for large models):
# 1. Interpolation of parameters
scen_REN2 <- interpolate(mdl, name = "RENBAL2")
# 2. Writing the model code and data in folder
write_model(scen_REN2, tmp.dir = "solwork/RENBAL2", solver = "GAMS")
# save(scen_REN2, file = "solwork/renbal/scen_REN2.RData")
# 3. Solving the model
# Alterntively, run GAMS directly in the model folder: 'gams mdl'
solve_model(tmp.dir = "solwork/RENBAL2")
# 4. Read the solution
scen_REN2 <- read_solution(scen_REN2, dir.result = "solwork/RENBAL2")
summary(scen_REN2)

save(scen_REN2, file = "scenarios/scen_RENBAL2.RData")
```

Check the results

```r
if (!exists("scen")) {
  if(!exists("scen_REN")) load("scenarios/scen_RENBAL.RData")
  if(!exists("scen_REN2")) load("scenarios/scen_RENBAL2.RData")
  # Select scenario for analysis
  scen <- scen_REN
  scen <- scen_REN2
}

# Objective (system costs)
(vObjective <- getData(scen, name = "vObjective", merge = T))
```

```
## # A tibble: 1 x 3
##   scenario name        value
##   <chr>    <chr>       <dbl>
## 1 RENBAL2  vObjective 381374.
```

```r
# Total electricity output from generating technologies
(vTechOut_ELC <- getData(scen, name = "vTechOut", comm = "ELC", tech_ = "^E",
                         merge = T))
```

```
## # A tibble: 221,598 x 8
##    scenario name     tech  comm  region  year slice    value
##    <chr>    <chr>    <chr> <chr> <chr>  <int> <chr>    <dbl>
##  1 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h10 0.559
##  2 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h11 2.30
##  3 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h12 4.01
##  4 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h13 5.22
##  5 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h14 5.76
##  6 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h15 5.56
##  7 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h16 4.67
##  8 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h17 3.21
##  9 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h18 1.43
## 10 RENBAL2  vTechOut ESOL  ELC   NV      2017 d002_h11 1.51
## # ... with 221,588 more rows
```

```r
unique(vTechOut_ELC$tech)
```

```
## [1] "ESOL" "EWIN" "EWFF"
```

```r
# Total demand
(pDemand_ELC <- getData(scen, name = "pDemand", comm = "ELC", merge = T))
```

```
## # A tibble: 429,240 x 8
##    scenario name    dem        comm  region  year slice     value
##    <chr>    <chr>   <chr>      <chr> <chr>  <dbl> <chr>     <dbl>
##  1 RENBAL2  pDemand DEM_ELC_DH ELC   WA      2017 d001_h00 14.7
##  2 RENBAL2  pDemand DEM_ELC_DH ELC   MT      2017 d001_h00  3.74
##  3 RENBAL2  pDemand DEM_ELC_DH ELC   ME      2017 d001_h00  1.26
##  4 RENBAL2  pDemand DEM_ELC_DH ELC   ND      2017 d001_h00  4.83
##  5 RENBAL2  pDemand DEM_ELC_DH ELC   SD      2017 d001_h00  1.22
##  6 RENBAL2  pDemand DEM_ELC_DH ELC   WY      2017 d001_h00  6.08
##  7 RENBAL2  pDemand DEM_ELC_DH ELC   WI      2017 d001_h00  7.80
##  8 RENBAL2  pDemand DEM_ELC_DH ELC   ID      2017 d001_h00  1.90
##  9 RENBAL2  pDemand DEM_ELC_DH ELC   VT      2017 d001_h00  0.237
```

```
## 10 RENBAL2  pDemand DEM_ELC_DH ELC   MN      2017 d001_h00  7.50
## # ... with 429,230 more rows
```

```r
# Total ELC Output vs Demand
sum(vTechOut_ELC$value) / sum(pDemand_ELC$value)
```

```
## [1] 1.637351
```

```r
# System costs USD/kWh (== MUSD/GWh)
## for electricity consumed
vObjective$value / sum(pDemand_ELC$value)
```

```
## [1] 0.09536775
```

```r
## for electricity produced
vObjective$value / sum(vTechOut_ELC$value)
```

```
## [1] 0.05824516
```

```r
# Supply curtailments
(vExportROW_ELC <- getData(scen, name = "vExportRow", merge = T))
```

```
## # A tibble: 18,619 x 8
##    scenario name       expp  comm  region  year slice    value
##    <chr>    <chr>      <chr> <chr> <chr>  <int> <chr>    <dbl>
##  1 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h08  9.27
##  2 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h09 27.6
##  3 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h10 34.6
##  4 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h11 43.3
##  5 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h12 55.6
##  6 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h13 74.8
##  7 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h14 85.8
##  8 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h15 90.1
##  9 RENBAL2  vExportRow EEXP  ELC   ND      2017 d008_h16 78.8
## 10 RENBAL2  vExportRow EEXP  ELC   ND      2017 d009_h09 12.7
## # ... with 18,609 more rows
```

```r
# Demand curtailments
(vImportROW_ELC <- getData(scen, name = "vImportRow", merge = T))
```

```
## NULL
```

```r
# Generating capacity
(vTechCap <- getData(scen, name = "vTechCap", merge = T))
```

```
## # A tibble: 97 x 6
##    scenario name     tech  region  year  value
##    <chr>    <chr>    <chr> <chr>  <int>  <dbl>
##  1 RENBAL2  vTechCap ESOL  NV      2017  12.6
##  2 RENBAL2  vTechCap ESOL  CA      2017 487.
##  3 RENBAL2  vTechCap ESOL  AZ      2017  64.2
##  4 RENBAL2  vTechCap ESOL  NC      2017   2.60
##  5 RENBAL2  vTechCap ESOL  TX      2017  62.8
##  6 RENBAL2  vTechCap ESOL  NM      2017 418.
##  7 RENBAL2  vTechCap ESOL  SC      2017  25.3
##  8 RENBAL2  vTechCap ESOL  FL      2017 726.
##  9 RENBAL2  vTechCap EWIN  ND      2017 307.
## 10 RENBAL2  vTechCap EWIN  SD      2017   2.06
## # ... with 87 more rows
```

```r
vTechCap %>%
  group_by(scenario, tech, year) %>%
  summarise(GW = sum(value))
```

```
## # A tibble: 5 x 4
## # Groups:   scenario, tech [5]
##   scenario tech     year     GW
##   <chr>    <chr>   <int>  <dbl>
## 1 RENBAL2  ELC2UHV  2017 1274.
## 2 RENBAL2  ESOL     2017 1798.
## 3 RENBAL2  EWFF     2017   21.0
## 4 RENBAL2  EWIN     2017 1406.
## 5 RENBAL2  UHV2ELC  2017  559.
```

```r
# Storage capacity
getData(scen, name = "vStorageCap", merge = T)
```

```
## # A tibble: 91 x 6
##    scenario name        stg     region  year   value
##    <chr>    <chr>       <chr>   <chr>  <int>   <dbl>
##  1 RENBAL2  vStorageCap STGELCH WA      2017 0.366
##  2 RENBAL2  vStorageCap STGELCH MT      2017 0.168
##  3 RENBAL2  vStorageCap STGELCH ME      2017 0.0215
##  4 RENBAL2  vStorageCap STGELCH ND      2017 0.964
##  5 RENBAL2  vStorageCap STGELCH SD      2017 0.119
##  6 RENBAL2  vStorageCap STGELCH WY      2017 0.00257
##  7 RENBAL2  vStorageCap STGELCH WI      2017 0.186
##  8 RENBAL2  vStorageCap STGELCH ID      2017 0.0259
##  9 RENBAL2  vStorageCap STGELCH IA      2017 0.00331
## 10 RENBAL2  vStorageCap STGELCH MA      2017 0.0269
## # ... with 81 more rows
```

```r
sum(getData(scen, name = "vStorageCap", merge = T)$value)
```

```
## [1] 11074.62
```

```r
getData(scen, name = "vStorageCap", merge = T) %>%
  group_by(scenario, stg, region, year) %>%
  summarize(GW = sum(value))
```

```
## # A tibble: 91 x 5
## # Groups:   scenario, stg, region [91]
##    scenario stg     region  year        GW
##    <chr>    <chr>   <chr>  <int>     <dbl>
##  1 RENBAL2  STGELCD AL      2017  168.
##  2 RENBAL2  STGELCD AR      2017   65.8
##  3 RENBAL2  STGELCD AZ      2017  310.
##  4 RENBAL2  STGELCD CA      2017 1432.
##  5 RENBAL2  STGELCD CO      2017  163.
##  6 RENBAL2  STGELCD CT      2017   31.7
##  7 RENBAL2  STGELCD DC      2017    0.0916
##  8 RENBAL2  STGELCD DE      2017    9.31
##  9 RENBAL2  STGELCD FL      2017 3007.
## 10 RENBAL2  STGELCD GA      2017  163.
## # ... with 81 more rows
```

```r
(vStorageCap_STGELCD <- getData(scen, name = "vStorageCap", merge = T, stg = "STGELCD"))
```

```
## # A tibble: 49 x 6
##    scenario name        stg     region  year  value
##    <chr>    <chr>       <chr>   <chr>  <int>  <dbl>
##  1 RENBAL2  vStorageCap STGELCD WA      2017  76.7
##  2 RENBAL2  vStorageCap STGELCD MT      2017  26.3
##  3 RENBAL2  vStorageCap STGELCD ME      2017  18.7
##  4 RENBAL2  vStorageCap STGELCD ND      2017  13.3
##  5 RENBAL2  vStorageCap STGELCD SD      2017  16.7
##  6 RENBAL2  vStorageCap STGELCD WY      2017  43.8
##  7 RENBAL2  vStorageCap STGELCD WI      2017 132.
##  8 RENBAL2  vStorageCap STGELCD ID      2017  13.9
##  9 RENBAL2  vStorageCap STGELCD VT      2017   1.49
## 10 RENBAL2  vStorageCap STGELCD MN      2017 545.
## # ... with 39 more rows
```

```r
(vStorageCap_STGELCH <- getData(scen, name = "vStorageCap", merge = T, stg = "STGELCH"))
```

```
## # A tibble: 42 x 6
##    scenario name        stg     region  year   value
##    <chr>    <chr>       <chr>   <chr>  <int>   <dbl>
##  1 RENBAL2  vStorageCap STGELCH WA      2017 0.366
##  2 RENBAL2  vStorageCap STGELCH MT      2017 0.168
##  3 RENBAL2  vStorageCap STGELCH ME      2017 0.0215
##  4 RENBAL2  vStorageCap STGELCH ND      2017 0.964
##  5 RENBAL2  vStorageCap STGELCH SD      2017 0.119
##  6 RENBAL2  vStorageCap STGELCH WY      2017 0.00257
##  7 RENBAL2  vStorageCap STGELCH WI      2017 0.186
##  8 RENBAL2  vStorageCap STGELCH ID      2017 0.0259
##  9 RENBAL2  vStorageCap STGELCH IA      2017 0.00331
## 10 RENBAL2  vStorageCap STGELCH MA      2017 0.0269
## # ... with 32 more rows
```

```r
sum(vStorageCap_STGELCD$value) # GWh - interdday (~seasonal) storage
```

```
## [1] 11027.53
```

```r
sum(vStorageCap_STGELCH$value) # GWh - intraday (hourly) storage
```

```
## [1] 47.09388
```

```r
# Ir-Trade
```

Shape of supply

```r
(vTechOut_ELC <- getData(scen, name = "vTechOut", comm = "ELC",
                         tech_ = "^E",
                         merge = T, newNames = c("value" = "GWh")))
```

```
## # A tibble: 221,598 x 8
##    scenario name     tech  comm  region  year slice      GWh
##    <chr>    <chr>    <chr> <chr> <chr>  <int> <chr>    <dbl>
##  1 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h10 0.559
##  2 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h11 2.30
```

```
##   3 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h12 4.01
##   4 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h13 5.22
##   5 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h14 5.76
##   6 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h15 5.56
##   7 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h16 4.67
##   8 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h17 3.21
##   9 RENBAL2  vTechOut ESOL  ELC   NV      2017 d001_h18 1.43
## 10 RENBAL2  vTechOut ESOL  ELC   NV      2017 d002_h11 1.51
## # ... with 221,588 more rows
```

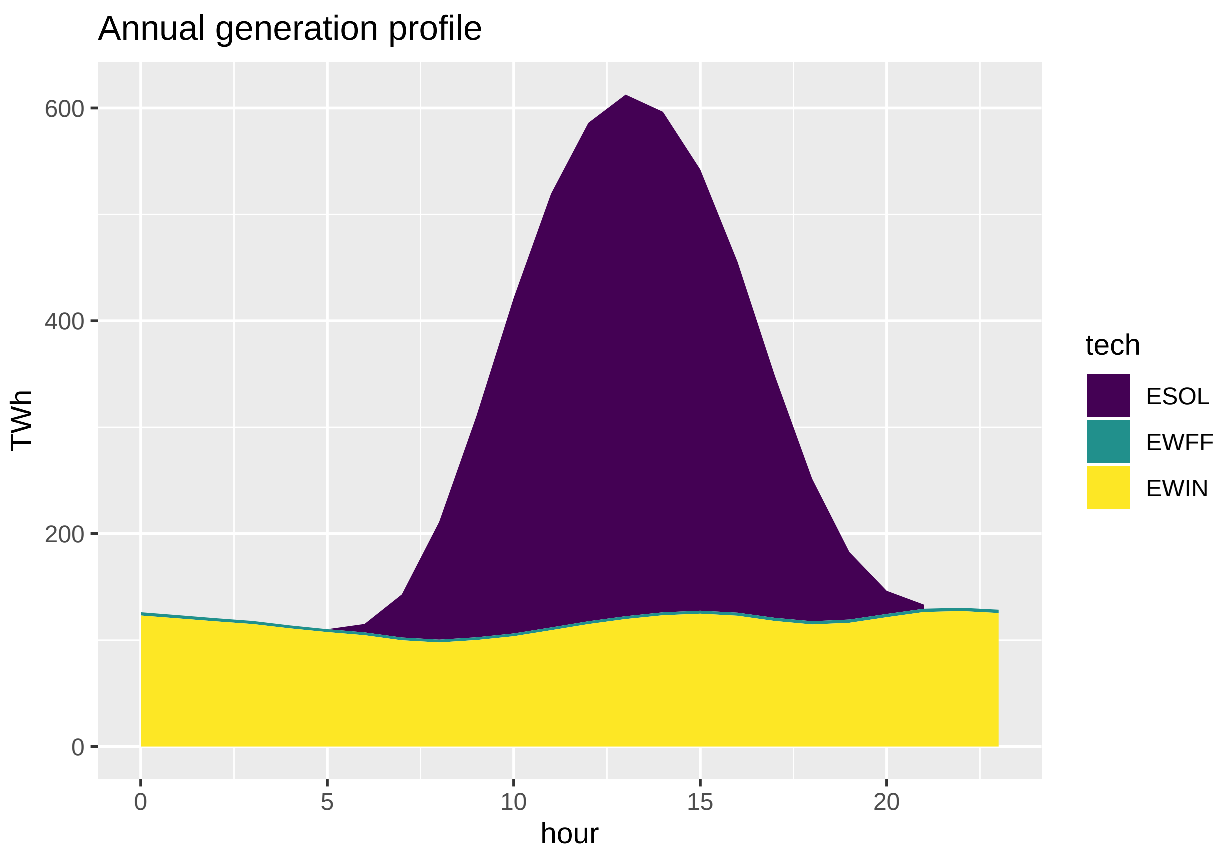
```r
summary(vTechOut_ELC$GWh)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0001   1.0650   5.0000  29.5478  22.1693 725.5340
```

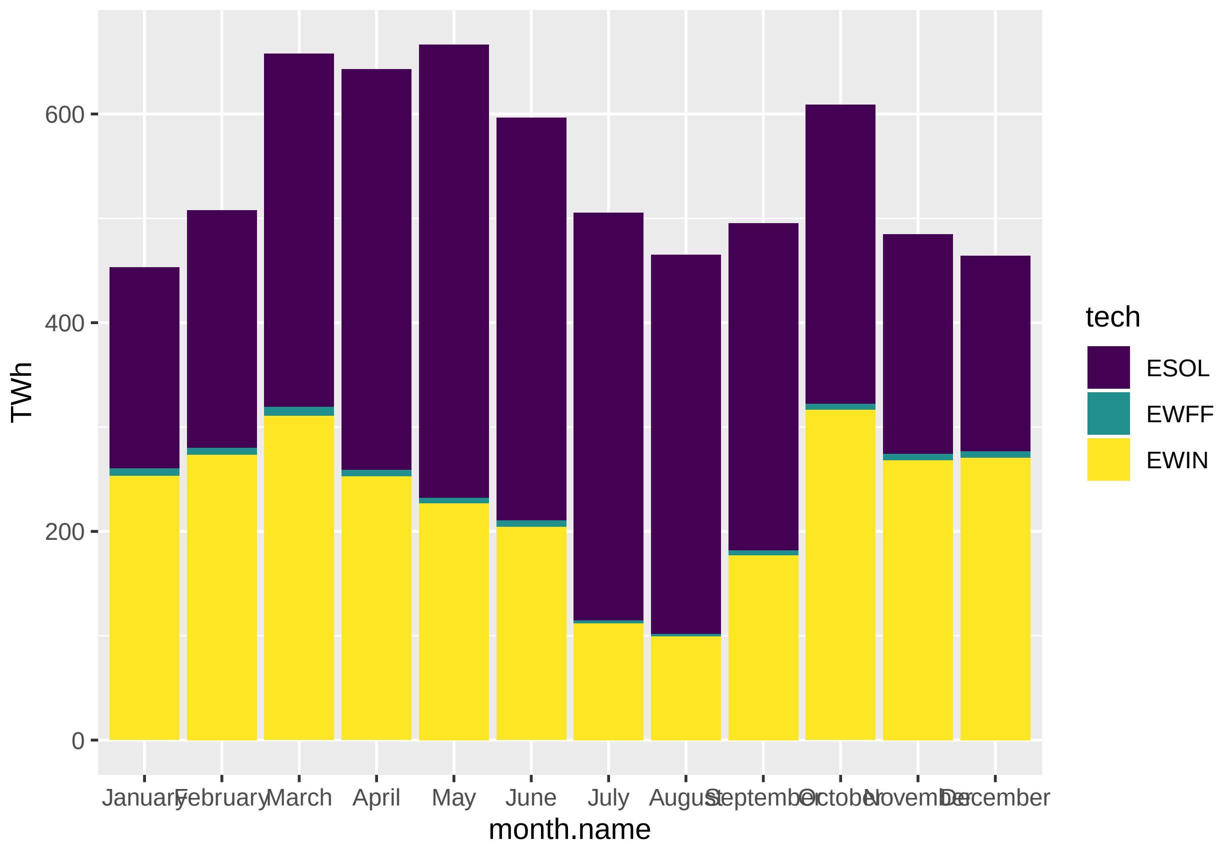
```r
unique(vTechOut_ELC$tech)
```

```
## [1] "ESOL" "EWIN" "EWFF"
```

```r
supy <- vTechOut_ELC %>%
  mutate(hour = as.integer(substr(slice, 7,8))) %>%
  group_by(scenario, year, hour, tech) %>%
  summarize(TWh = sum(GWh)/1e3) %>%
  mutate(tech = factor(tech,
                       levels = c("ESOL", "EWFF", "EWIN"),
                       ordered = T))

ggplot(supy) +
  geom_area(aes(x = hour, y = TWh, fill = tech), stat = "identity") +
  labs(title = "Annual generation profile")
```



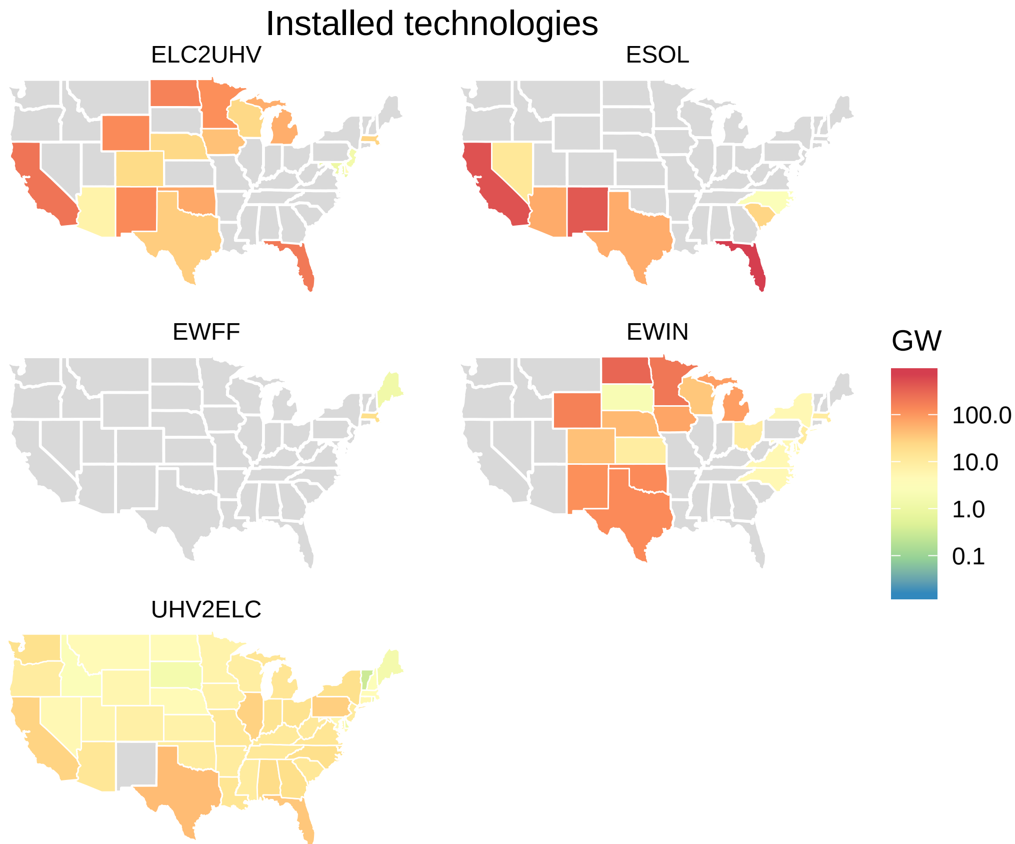

```r
supm <- vTechOut_ELC %>%
  mutate(#hour = as.integer(substr(slice, 7,8)),
         month = month(as.Date ("2017-01-01") + as.integer(substr(slice, 2, 4)) - 1)) %>%
  mutate(month.name = factor(month.name[month], levels = month.name[1:12], ordered = T)) %>%
  group_by(scenario, year, month, month.name, tech) %>%
  summarize(TWh = sum(GWh)/1e3) %>%
  mutate(tech = factor(tech,
                       levels = c("ESOL", "EWFF", "EWIN"),
                       ordered = T))
ggplot(supm) +
  geom_bar(aes(x = month.name, y = TWh, fill = tech), stat = "identity")
```



Capacity maps

```
cap_2017 <- getData(scen, name = "vTechCap", merge = T)

# Technologies capacity
# fig_map_cap_2017_by_tech <-
ggplot(data = usa49r) +
  geom_polygon(aes(x = long, y = lat, group = group), fill = "grey85",
                 colour = "white", alpha = 1, size = .5) +
    geom_polygon(aes(x = long, y = lat, group = group, fill = value),
                   data = right_join(usa49r, cap_2017, by = c("id" = "region")), #fill = "wheat",
               color = "white", alpha = 1, size = 0.25, inherit.aes = F) + # aes fill = id,
  coord_fixed(1.3) +
  theme_void() +
  # scale_fill_distiller(palette = "Spectral", direction = -1, name = "GW") +
  scale_fill_distiller(palette = "Spectral", direction = -1, trans = "log10", name = "GW") +
  labs(title = "Installed technologies") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  facet_wrap(.~tech, ncol = 2)
```

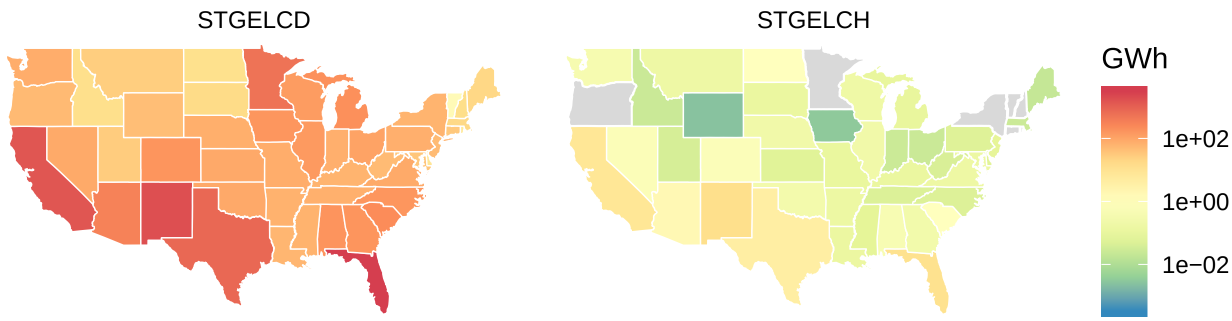
```
## Warning: Column `id`/`region` joining factor and character vector, coercing
## into character vector
```




```r
    # facet_wrap(.~tech, ncol = 2, scales = "free")
# fig_map_cap_2017_by_tech
# ggsave(file.path("fig", scen@name,"fig_map_cap_2017_by_tech.pdf"), fig_map_cap_2017_by_tech, device =

# Storage capacity
stg_2017 <- getData(scen, name = "vStorageCap", merge = T)
# stg_2017$GWh <- convert("PJ", "GWh", stg_2017$value)
stg_2017$GWh <- convert("GWh", "GWh", stg_2017$value)
fig_map_stg_2017 <-
ggplot(data = usa49r) +
  geom_polygon(aes(x = long, y = lat, group = group), fill = "grey85",
                 colour = "white", alpha = 1, size = .5) +
    geom_polygon(aes(x = long, y = lat, group = group, fill = GWh),
                  data = right_join(usa49r, stg_2017, by = c("id" = "region")), #fill = "wheat",
                color = "white", alpha = 1, size = 0.25, inherit.aes = F) + # aes fill = id,
    coord_fixed(1.3) +
    # scale_fill_distiller(palette = "Spectral", direction = -1, name = "GW") +
    scale_fill_distiller(palette = "Spectral", direction = -1, trans = "log10") +
    theme_void() + facet_wrap(.~stg)
```

```
## Warning: Column `id`/`region` joining factor and character vector, coercing
## into character vector
```

```r
fig_map_stg_2017
```




```
# ggsave(file.path("fig", scen@name,"fig_map_stg_2017.pdf"), fig_map_stg_2050, device = "pdf")
```

Trade flows maps

```
trdIr <- getData(scen, name = "vTradeIr", merge = T)
summary(trdIr$value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.017  11.368  20.085  27.305 100.000
```

```
# summary(convert("PJ", "GWh", trdIr$value))
trdIr[which(trdIr$value == max(trdIr$value)), ]
```

```
## # A tibble: 6,602 x 9
##    scenario name     trade      comm  src   dst    year slice    value
##    <chr>    <chr>    <chr>      <chr> <chr> <chr> <int> <chr>    <dbl>
##  1 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h02   100
##  2 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h03   100
##  3 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h04   100
##  4 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h05   100
##  5 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h06   100
##  6 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h07   100
##  7 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d012_h08   100
##  8 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d017_h20   100
##  9 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d017_h21   100
```

```
## 10 RENBAL2  vTradeIr TRD_UHV_ND UHV   ND    SD     2017 d017_h22   100
## # ... with 6,592 more rows
```

```
trdIr_year <- trdIr %>%
  group_by(src, dst, year) %>%
  # summarize(TWh = convert("PJ", "TWh", sum(value)))
  summarize(
    GW = max(value),
    GWh = sum(value))
trdIr_year
```

```
## # A tibble: 173 x 5
## # Groups:   src, dst [173]
##    src   dst    year    GW     GWh
##    <chr> <chr> <int> <dbl>   <dbl>
##  1 AL    FL     2017  35.6  24779.
##  2 AL    GA     2017  33.8  21984.
##  3 AL    MS     2017  77.8  40969.
##  4 AL    TN     2017  78.2  96698.
##  5 AR    LA     2017  14.9  11989.
##  6 AR    MS     2017  91.6 111754.
##  7 AR    OK     2017  21.9    162.
##  8 AR    TN     2017  52.5  32456.
##  9 AZ    CA     2017  26.7  16543.
## 10 AZ    CO     2017  88.5   6567.
## # ... with 163 more rows
```

```
trd_elc_y <- left_join(trdIr_year, trd_map)
```

```
## Joining, by = c("src", "dst")
```

```
summary(trd_elc_y$GWh)
```

```
##     Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
##      0.5     918.0   9870.3  43029.4  46420.3 474792.1
```

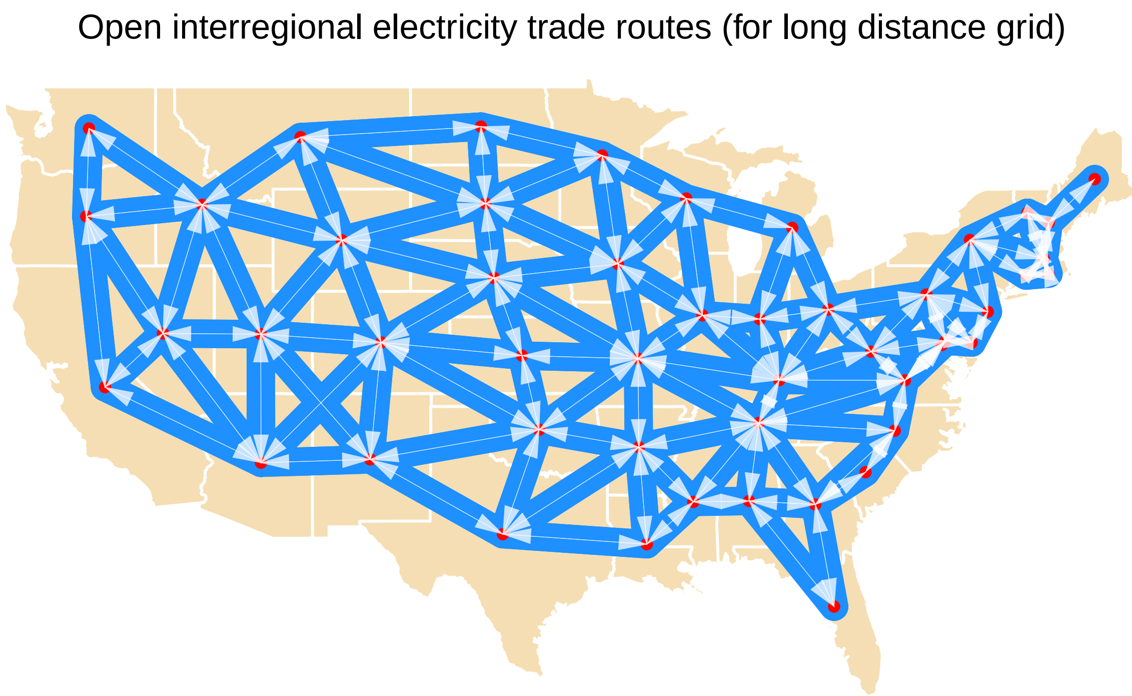
```
summary(trd_elc_y$GW)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.311   8.805  24.882  38.392  65.221 100.000
```

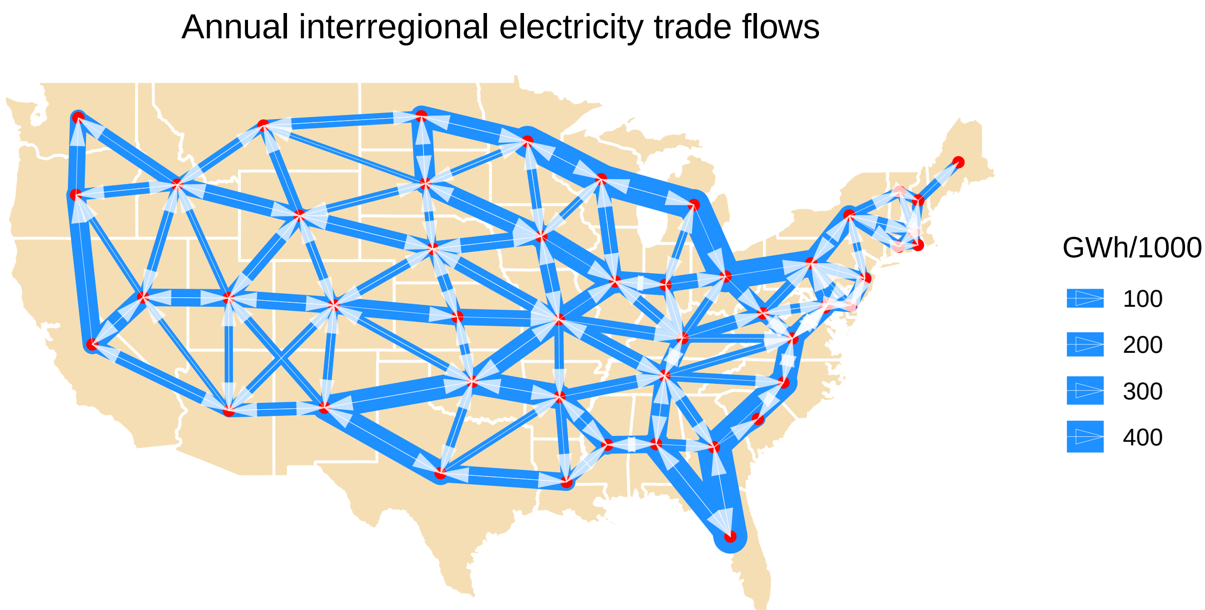
```
trd_elc_y[is.na(trd_elc_y$GWh),]
```

```
## # A tibble: 0 x 10
## # Groups:   src, dst [0]
## # ... with 10 variables: src <chr>, dst <chr>, year <int>, GW <dbl>,
## #   GWh <dbl>, trd <dbl>, xsrc <dbl>, ysrc <dbl>, xdst <dbl>, ydst <dbl>
```

```
trd_flows_map
```




```r
ggplot(data = usa49r) +
    geom_polygon(aes(x = long, y = lat, group = group), fill = "wheat",
                 colour = "white", alpha = 1, size = .5) + # aes fill = id,
    coord_fixed(1.3) +
    guides(fill=FALSE) +  # do this to leave off the color legend
    theme_void() + labs(title = "Annual interregional electricity trade flows")  +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5)) +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst, size = GWh/1e3),
                 data = trd_elc_y, inherit.aes = FALSE, #size = 3,
                 alpha = 1, colour = "dodgerblue", lineend = "round", show.legend = T) +
    geom_point(data = reg_centers, aes(x, y), colour = "red") +
    geom_segment(aes(x=xsrc, y=ysrc, xend=xdst, yend=ydst),
                 data = trd_elc_y, inherit.aes = FALSE, size = .1,
           arrow = arrow(type = "closed", angle = 15,
                         length = unit(0.15, "inches")),
           colour = "white", alpha = 0.75,
           lineend = "butt", linejoin = "mitre", show.legend = T)
```


Annual interregional electricity trade flows
GWh/1000
100
200
300
400