Missing Data

Multiple Imputation using R

This week:

- discuss package `mice`
- more `mi` options
- extracting datasets
- more diagnostics

Installing `mice` package if you use standalone R

Select mirror

Package installer

Check ”Install dependencies”

`mice` package

MICE = Multiple Imputation by Chained Equations

Built-in imputation models are provided for:

- continuous data (predictive mean matching, normal)
- binary data (logistic regression)
- unordered categorical data (polytomous logistic regression)
- and ordered categorical data (proportional odds).

Built-in univariate imputation methods are:

`mice` package methods

pmm	any	Predictive mean matching
midastouch	any	Weighted predictive mean matching
sample	any	Random sample from observed values
cart	any	Classification and regression trees
rf	any	Random forest imputations
mean	numeric	Unconditional mean imputation
norm	numeric	Bayesian linear regression
norm.nob	numeric	Linear regression ignoring model error
norm.boot	numeric	Linear regression using bootstrap
norm.predict	numeric	Linear regression, predicted values
quadratic	numeric	Imputation of quadratic terms
ri	numeric	Random indicator for nonignorable data
logreg	binary	Logistic regression
logreg.boot	binary	Logistic regression with bootstrap
polr	ordered	Proportional odds model
polyreg	unordered	Polytomous logistic regression
lda	unordered	Linear discriminant analysis

`logreg` method (Algorithm 3.5 from van Buuren):

- Estimate regression coefficients $\hat{\boldsymbol{\beta}}$ from $(y_{\text{obs}}, \boldsymbol{X}_{\text{obs}})$ by IRLS
- Obtain $\boldsymbol{V}$, estimated covariance matrix of $\hat{\boldsymbol{\beta}}$
- Draw $\dot{\boldsymbol{z}}_1$, a vector of independent N(0,1) variables
- Calculate the Cholesky decomposition $\boldsymbol{V}^{1/2}$
- Calculate $\dot{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} + \dot{\boldsymbol{z}}_1 \boldsymbol{V}^{\frac{1}{2}}$
- Calculate n_0 predicted probabilities $\dot{\boldsymbol{p}} = \frac{1}{1+e^{-\boldsymbol{X}_{\text{mis}}\dot{\boldsymbol{\beta}}}}$
- Draw $\boldsymbol{u}$, a vector of n_0 random variables from the uniform distribution U(0, 1).
- Calculate imputations $\dot{y}_j = 1$ if $u_j \leq \dot{p}_j$, and $\dot{y}_j = 0$ otherwise, where $j = 1, \ldots, n_0$.

Important functions from the `mice` package

`mice`: does the actual imputation

`pool`: pools together the imputed results

`complete`: returns completed datasets

`md.pattern`: shows patterns of missing data

`pool` function details from the `mice` package

`fmi`: stands for "fraction of missing information"

- Statistical formula separating Fisher information into observed and missing (Rubin, 1987)
- Rule of thumb: set *m* equal to the number of incomplete cases
- Relative efficiency of imputations: FMI/*m* ≈.01
- Annoying in that FMI depends on *m* (Spratt et al., 2010)
- But you could impute a few datasets, check FMI, then impute again...then check FMI again! (White, Royston, Wood, 2011; Graham, Olchowski, & Gilreath, 2007)

Recall: Combining estimates across datasets

Given estimates $Q_1, \ldots, Q_M$ and their corresponding standard errors $s_1, \ldots, s_M$

point estimate: $\theta = 1/m \sum_m Q_m$

variance: $\bar{U} + (1 + m^{-1})B$

where,

$\bar{U} = 1/m \sum_m s_m^2$

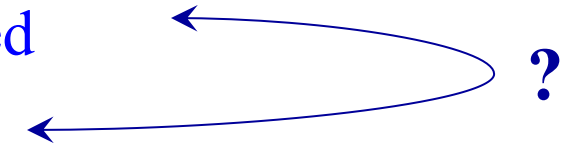
$B = 1/(m-1) \sum_m (Q_m - \theta)^2$

`pool` function details from the `mice` package

- `ubar`: The mean of the variances, formula (3.1.3), Rubin (1987).
- `b`: The within imputation variance, formula (3.1.4), Rubin (1987)
- `t`: Total variance of the pooled estimates, formula (3.1.5), Rubin (1987)
- `dfcom`: Degrees of freedom for estimates in the complete data analysis.
- `df`: Degrees of freedom for *t* reference distribution, calculated according to the article of Barnard and Rubin (1999).

Basic steps when imputing using `mi/mice`

1. Load the data
2. Create a missing_data object, look at the data and the missing data patterns
3. Examine the default choices for imputation models
4. Make changes to imputation models if necessary
5. Impute until converged
6. Plot diagnostics
7. Iterate between 4-6 if necessary
8. Run final pooled analysis



Fit MI to airquality dataset using both `mice and mi`

First create more missing data

```
# Generate MAR data from airquality
library(mice)

# First get just the fully observed data
air.miss = airquality[, 3:4]
a = ampute(air.miss)

# Put it back together with the other two variables
air.miss = cbind(airquality[, 1:2], a$amp)
```

Load data into mi

```
library(mi)
mdf <- missing_data.frame(air.miss)
summary(mdf)
image(mdf)
show(mdf)
```

summary(mdf)

```
     Ozone           Solar.R           Wind             Temp           Month            Day       
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.000   Min.   : 1.0  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:73.00   1st Qu.:6.000   1st Qu.: 8.0  
 Median : 31.50   Median :205.0   Median : 9.700   Median :80.00   Median :7.000   Median :16.0  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.654   Mean   :78.52   Mean   :6.993   Mean   :15.8  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0  
 NA's   :37       NA's   :7       NA's   :13       NA's   :10                                    
```

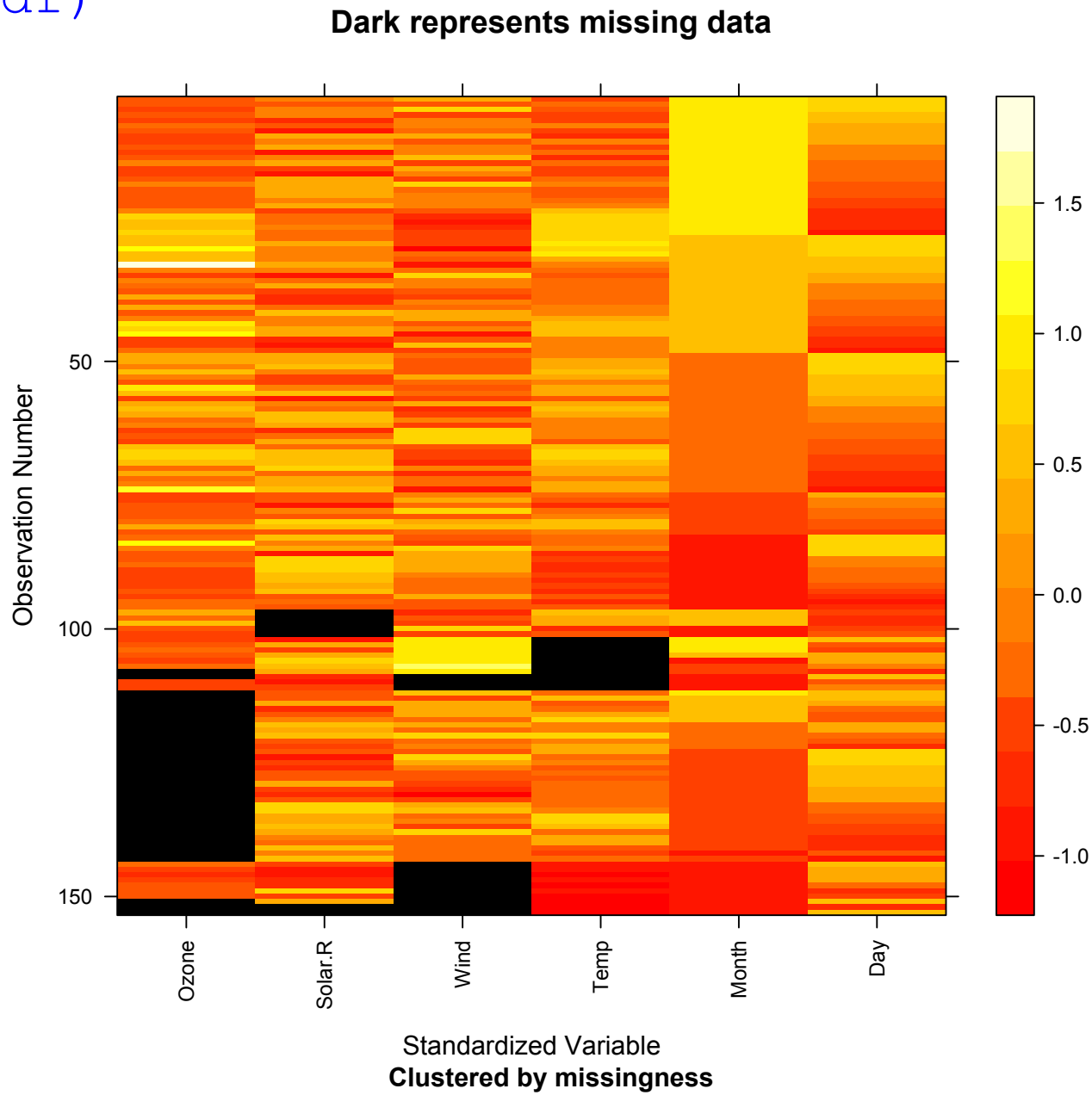
```
> tabulate(mdf@patterns)
[1] 96  7  5 32  6  2  1  1  2  1

> levels(mdf@patterns)
[1] "nothing"              "Wind"                 "Solar.R"
"Ozone"                "Temp"                 "Wind, Temp"
[7] "Ozone, Wind"          "Ozone, Temp"          "Ozone, Solar.R, Wind"
"Ozone, Wind, Temp"
```

```
show(mdf)
```

	type	missing	method	model
Ozone	continuous	37	ppd	linear
Solar.R	continuous	7	ppd	linear
Wind	continuous	13	ppd	linear
Temp	continuous	10	ppd	linear
Month	ordered-categorical	0	<NA>	<NA>
Day	continuous	0	<NA>	<NA>

	family	link	transformation
Ozone	gaussian	identity	standardize
Solar.R	gaussian	identity	standardize
Wind	gaussian	identity	standardize
Temp	gaussian	identity	standardize
Month	<NA>	<NA>	<NA>
Day	<NA>	<NA>	standardize

```
image(mdf)
```


Dark represents missing data
Observation Number
50
100
150
Ozone
Solar.R
Wind
Temp
Month
Day
1.5
1.0
0.5
0.0
-0.5
-1.0
Standardized Variable
Clustered by missingness

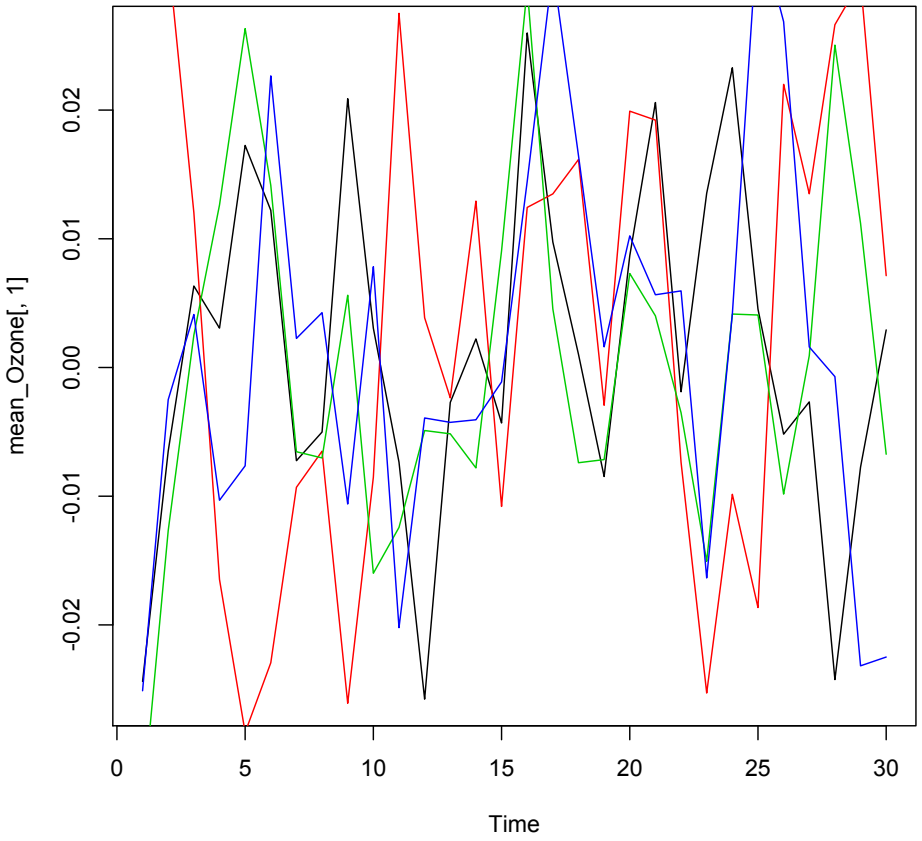
Suppose we want to make changes

```
> mdf <- change(mdf, y = c("Month"), what = "type", to = "con")
```

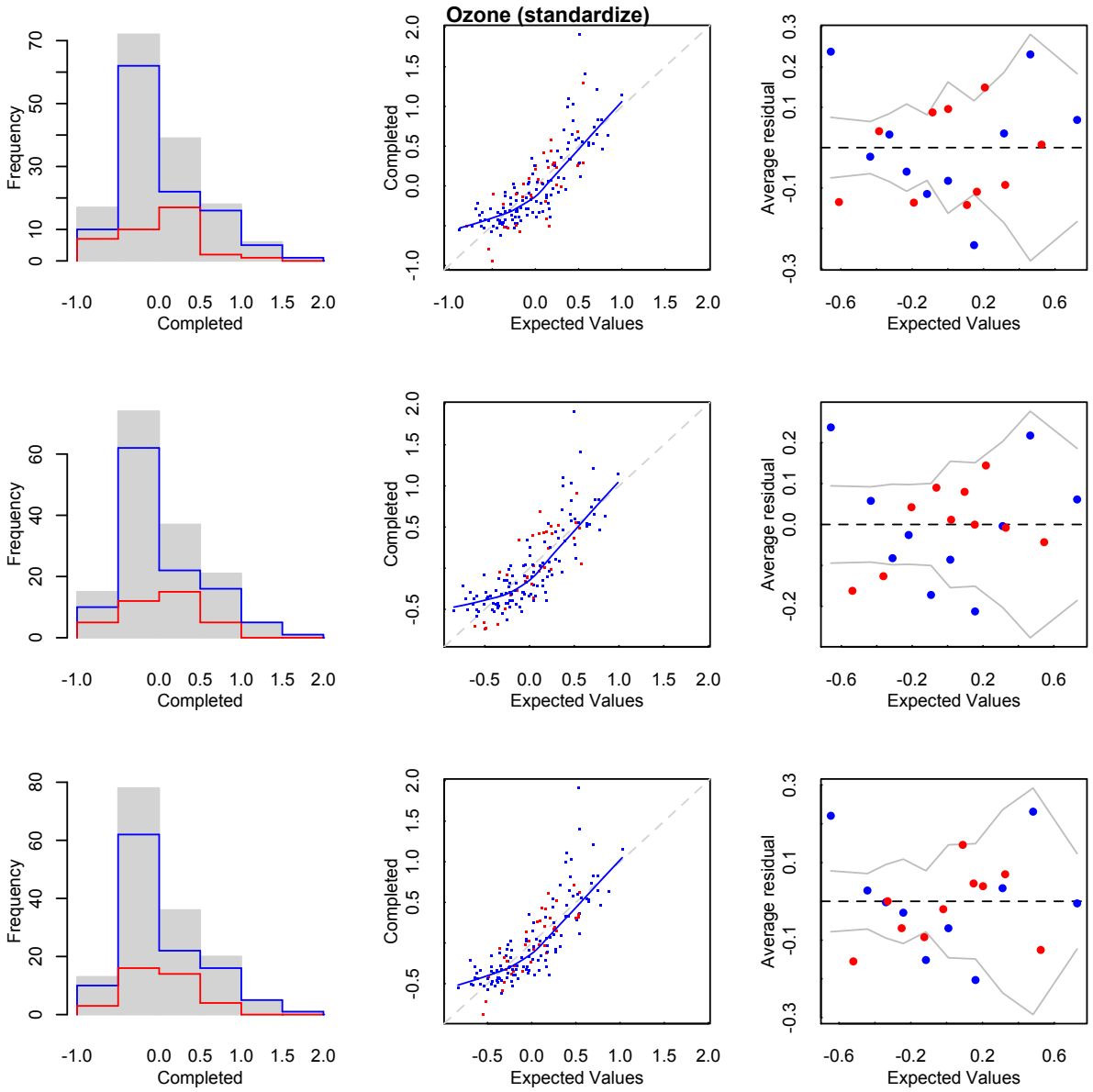
	type	missing	method	model
Ozone	continuous	37	ppd	linear
Solar.R	continuous	7	ppd	linear
Wind	continuous	13	ppd	linear
Temp	continuous	10	ppd	linear
Month	continuous	0	<NA>	<NA>
Day	continuous	0	<NA>	<NA>

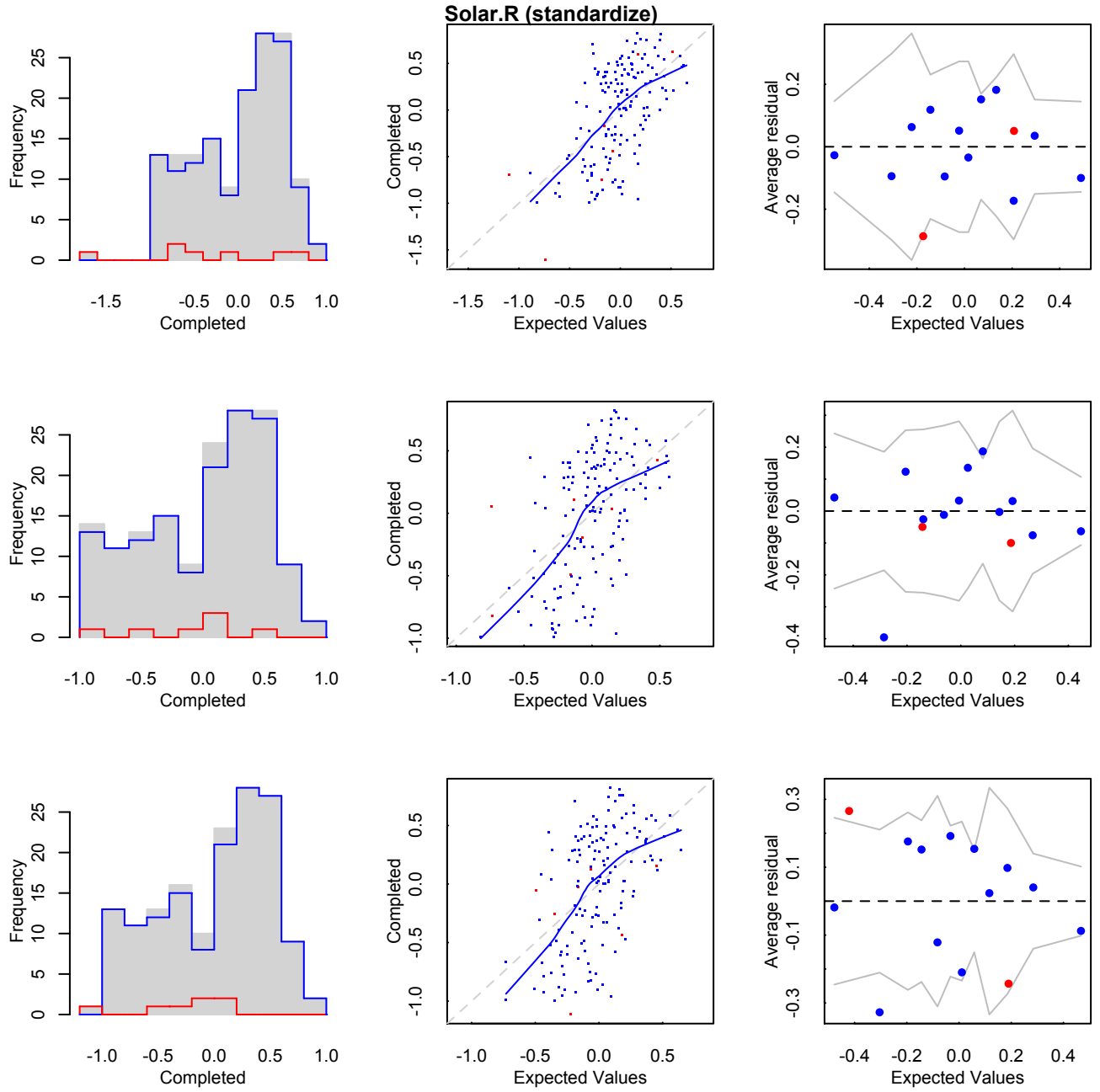
	family	link	transformation
Ozone	gaussian	identity	standardize
Solar.R	gaussian	identity	standardize
Wind	gaussian	identity	standardize
Temp	gaussian	identity	standardize
Month	<NA>	<NA>	standardize
Day	<NA>	<NA>	standardize

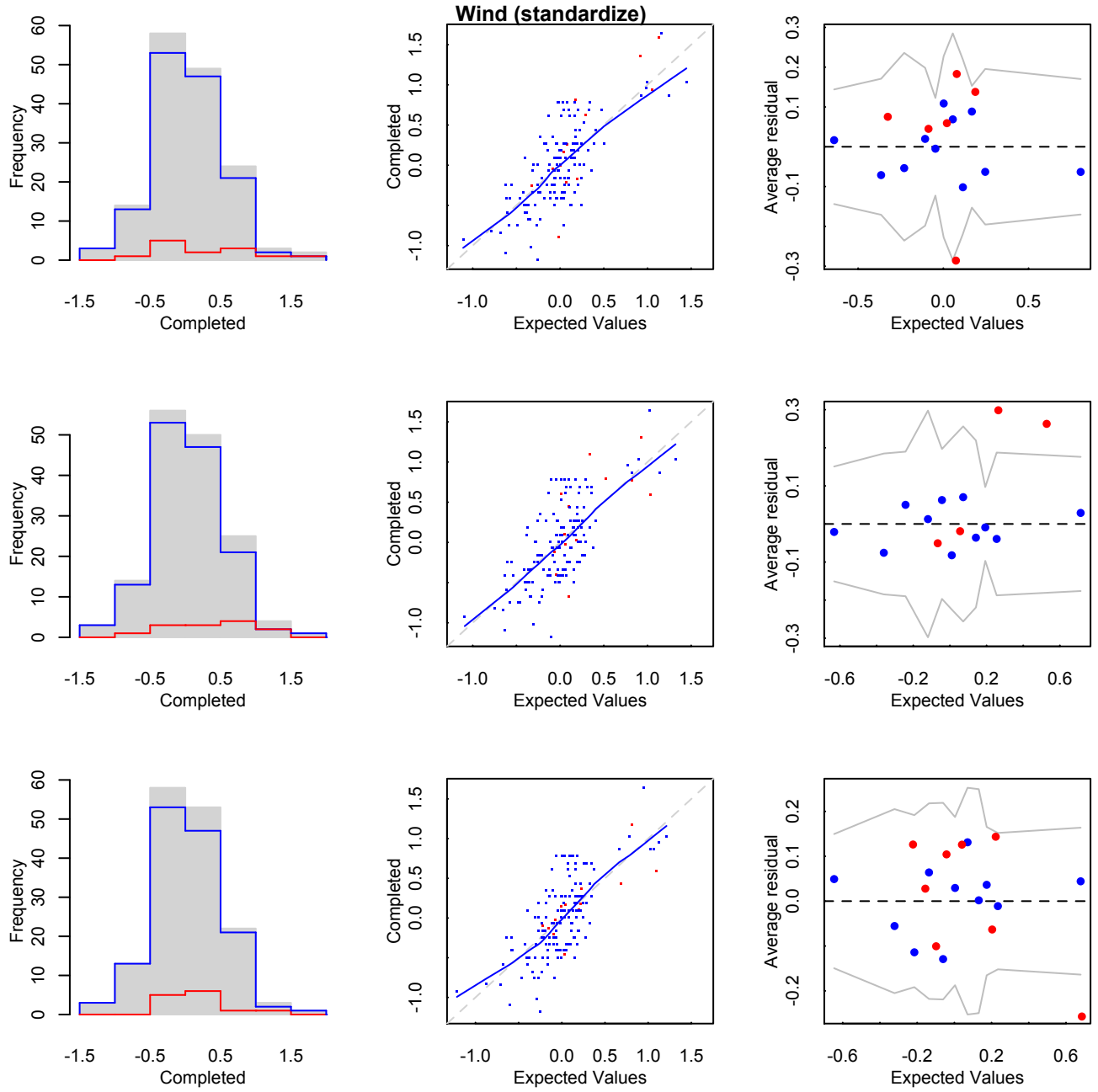
```
imp.air <- mi(mdf, seed=124)
converged <- mi2BUGS(imp.mcar)
mean_Ozone = converged[, , 1]
```

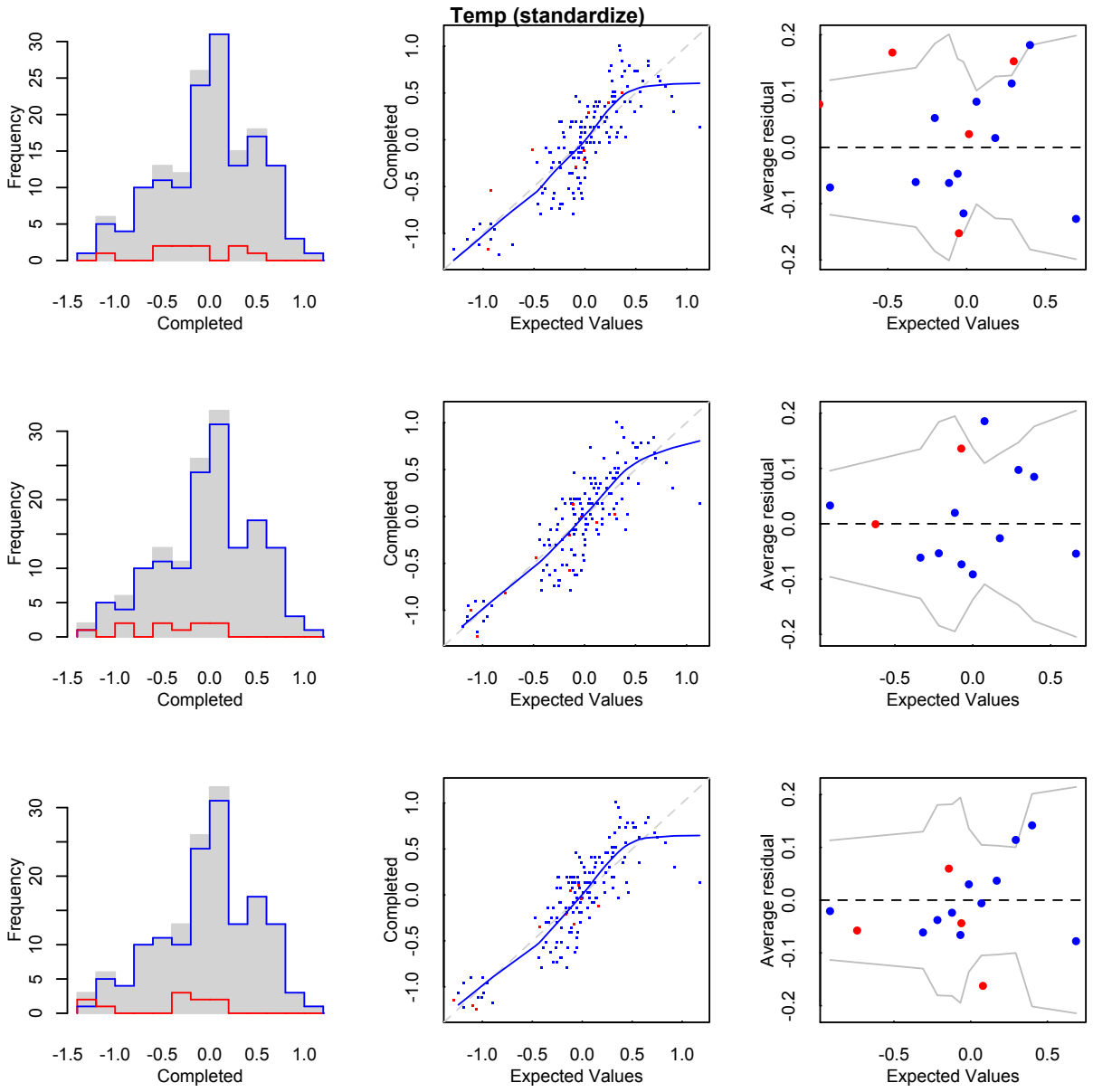


```
plot(imp.air)
```


Ozone (standardize)
Frequency
Completed
Expected Values
Average residual


Solar.R (standardize)
Frequency
Completed
Completed
Expected Values
Average residual
Expected Values


Wind (standardize)
Frequency
Completed
Expected Values
Average residual


Temp (standardize)
Frequency
Completed
Expected Values
Average residual

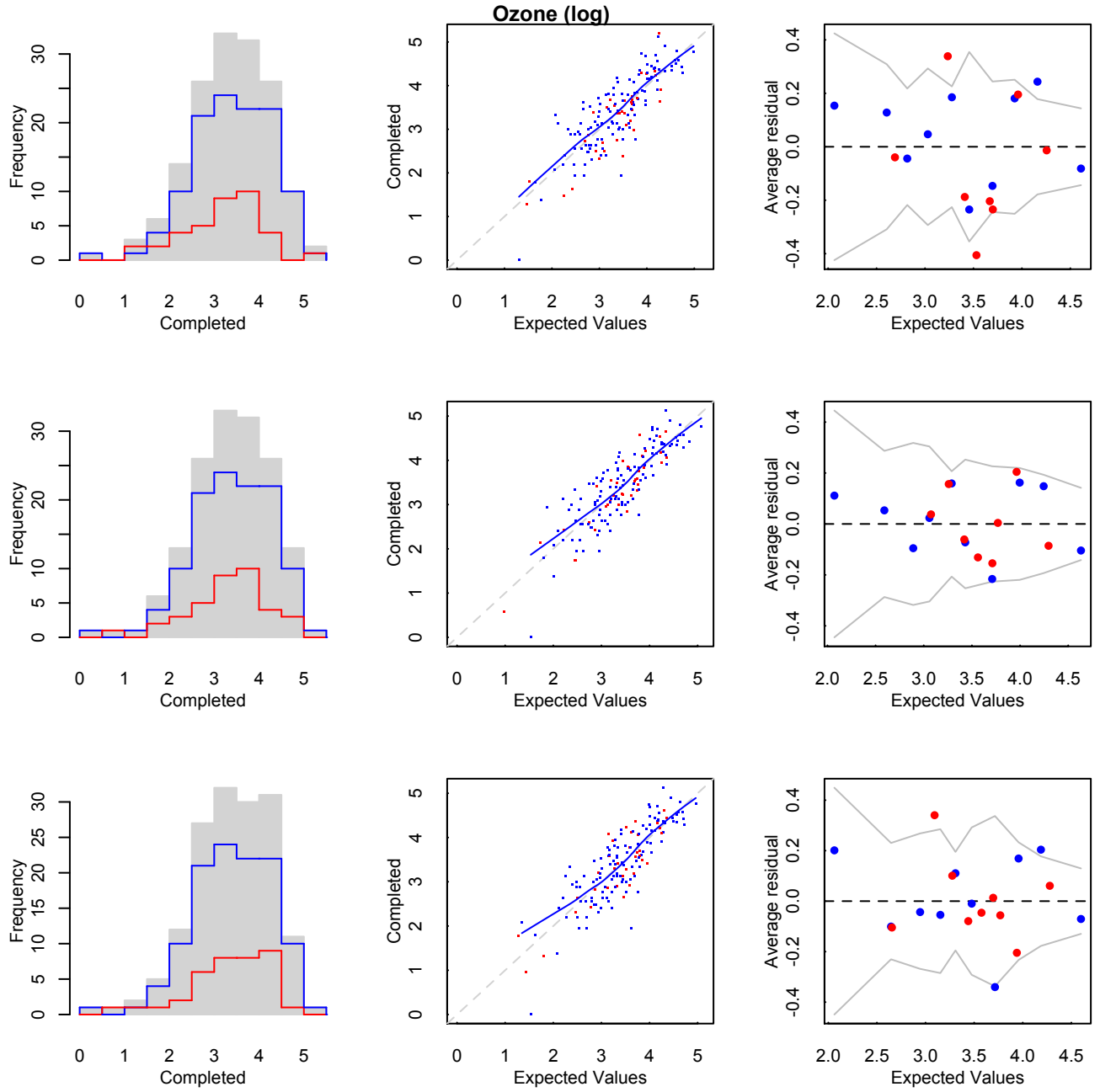
make some changes…

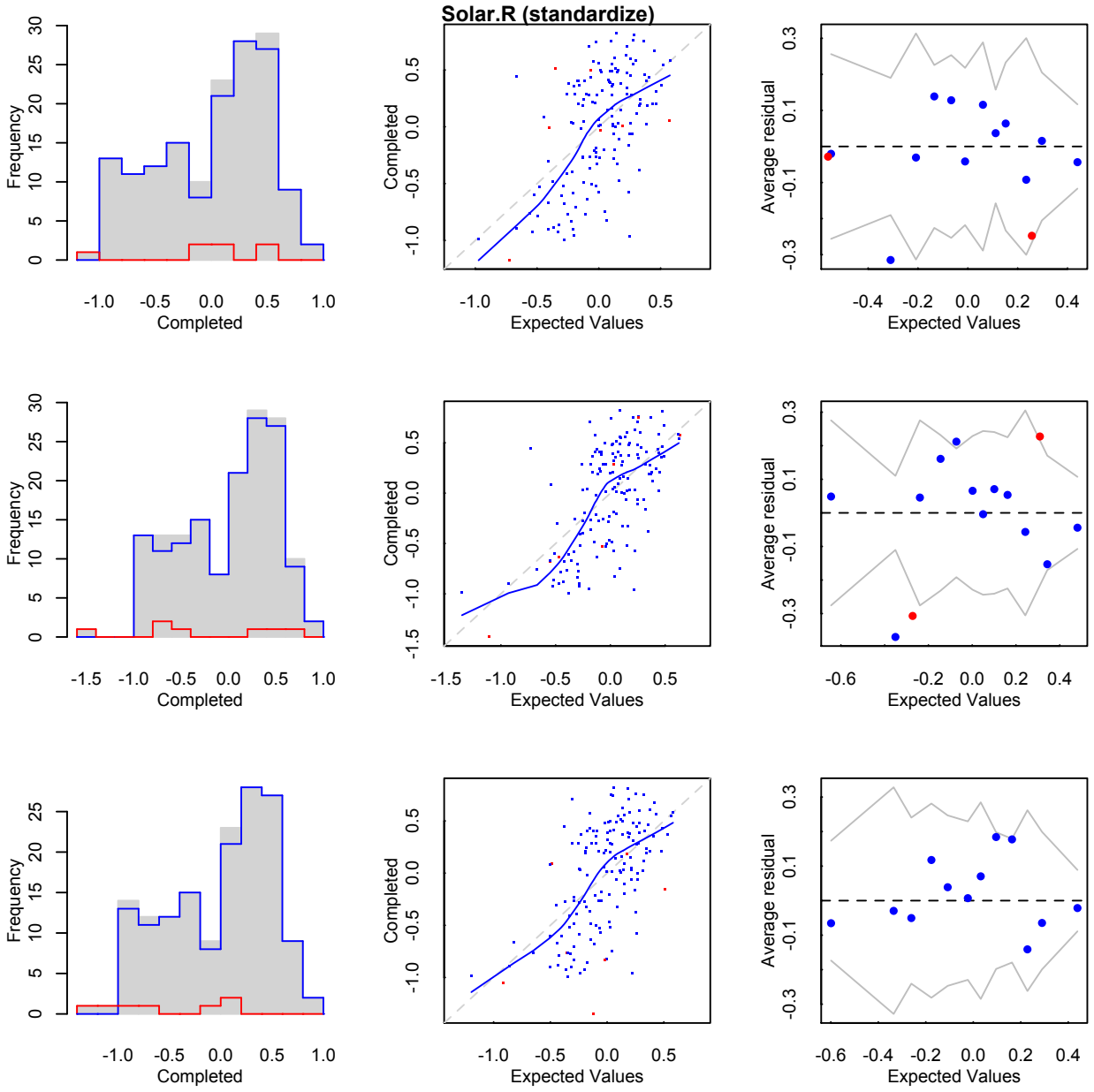
Let's say we want Ozone to be log-transformed, which happens automatically if we change the imputation method to positive-continuous .

```
mdf <- change(mdf, y = "Ozone", what = "type", to
= "pos")
```

After the changes…

```
> imp.air <- mi(mdf, seed = 124)
> converged <- mi2BUGS(imputations)
> plot(imp.air)
```


Ozone (log)
Frequency
Completed
Completed
Expected Values
Average residual
Expected Values
Frequency
Completed
Completed
Expected Values
Average residual
Expected Values
Frequency
Completed
Completed
Expected Values
Average residual
Expected Values


Solar.R (standardize)
Frequency
Completed
Expected Values
Average residual

to deal with Solar.R try using pmm:

```
mdf = change(mdf, y = c(”Solar.R"), what
  = "imputation_method", to = "pmm")

imp.air3 <- mi(mdf, seed = 124)
plot(imp.air3)
```

Check graphs: looks like we solved the problem!

Other things you might need/want to do…

- Extract one or several "completed" datasets (i.e. observed plus imputed values)
- For one dataset

```
> air.comp1 = complete(imp.air3, m=1)
```

- For five datasets (saved as a "list")

```
> air.comp.1to5 = complete(imp.air3, m=5)
```

- To extract just the 2nd of the 5 datasets from this last command…

```
> air.comp2 = air.comp.1to5[[2]]
```

What can we do with these completed datasets?

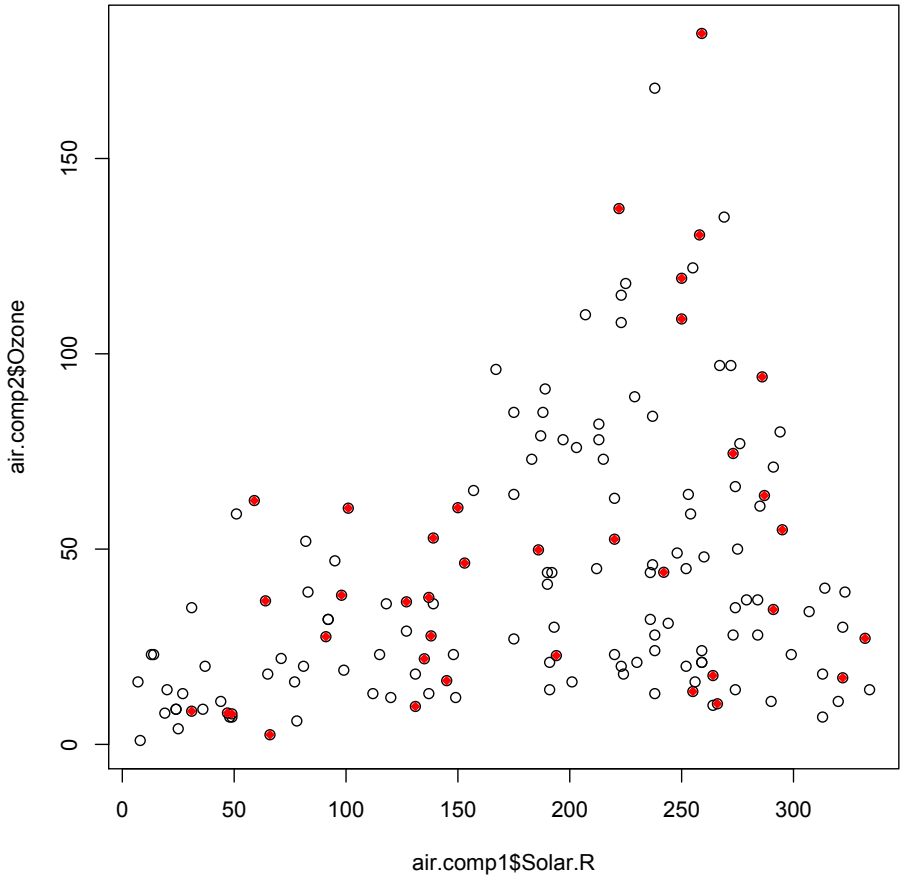
- Look at summary statistics across imputed datasets

```
> lapply(air.comp.1to5, summary)
```

- Perform an analysis in each dataset
- Perform additional diagnostics

Additional diagnostics

```
plot(air.comp1$Solar.R, air.comp1$Ozone)
points(air.comp1$Solar.R[air.comp1$missing_Ozone==1],
   air.comp1$Ozone[air.comp1$missing_Ozone==1],col="red",
   pch=18)
```



red dots are imputed values

Pooled analyses

```
> res.air = pool(Ozone ~ Wind + Temp + Solar.R,
data=imp.air3)
> display(res.air)
bayesglm(formula = Ozone ~ Wind + Temp + Solar.R, data
= imp.air)
            coef.est coef.se
(Intercept) -68.80    21.08
Wind         -3.46     0.69
Temp          1.67     0.23
Solar.R       0.08     0.02
n = 149, k = 4
residual deviance = 59810.2, null deviance = 154506.6
(difference = 94696.4)
overdispersion parameter = 401.4
residual sd is sqrt(overdispersion) = 20.04>
```

Missing data types currently supported

- for more info see

  > help(mi)

- then click on missing_variable anywhere in the text where it's highlighted

Missing Variable Types currently accommodated by `mi`

Class name [transformation]	Default family and link	Default fit_model
unordered-categorical	binomial(link = 'logit')	mlogit
ordered-categorical	binomial(link = 'logit')	bayespolr
interval	gaussian(link = 'identity‘)	survreg
binary	binomial(link = 'logit')	bayesglm
continuous[standardize]	gaussian(link = 'identity')	bayesglm
positive-continuous[log]	gaussian(link = 'identity')	bayesglm
proportion[identity]	binomial(link = 'logit')	betareg
bounded-continuous [identity]	gaussian(link = 'identity')	betareg
count	quasipoisson(link = 'log')	bayesglm
semi-continuous* (see next slide)		

Semi-continuous variables

Semi-continuous variables have support on one or more points and an interval. They are two types:

1) nonnegative-continuous

These are variables which have some values at 0 and the rest positive (e.g. income often looks this way)

Semi-continuous variables

2) SC_proportion

These are proportions where some of the observed values are exactly zero and/or exactly one

In each case we first fit a model (models) for the point mass (at 0 or 1) and then for those with values not at the point mass we fit an appropriate model:

linear regression with log transform for semi-continuous

Beta regression for proportion with “squeeze transformation”