# Package 'connected'

July 22, 2025

**Title** Visualize and Improve Connectedness of Factors in Tables

**Version** 1.1

**Description** Visualize the connectedness of factors in two-way tables.
Perform two-way filtering to improve the degree of connectedness.
See Weeks & Williams (1964) <doi:10.1080/00401706.1964.10490188>.

**License** GPL (>= 2)

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** grDevices, lattice, lfe, reshape2, stats

**Suggests** agridat, dplyr, janitor, knitr, lme4, lucid, rmarkdown,
testthat (>= 3.0.0)

**VignetteBuilder** knitr

**LazyData** true

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**URL** https://kwstat.github.io/connected/

**BugReports** https://github.com/kwstat/connected/issues

**NeedsCompilation** no

**Author** Kevin Wright [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0617-8673>)

**Maintainer** Kevin Wright <kw.stat@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-03-05 13:30:02 UTC

## Contents

con_check . . . . . 2
con_concur . . . . . 3
con_filter . . . . . 4

con_view . . . . . 6
data_eccleston . . . . . 8
data_fernando . . . . . 9
data_searle . . . . . 10
data_student . . . . . 11
data_tosh . . . . . 12
data_weeks1 . . . . . 12
data_weeks2 . . . . . 13

**Index** **15**

---

`con_check` *Check connectedness of multiple factors in a dataframe*

---

## Description

Multiple factors in a dataframe are said to be connected if a model matrix based on those factors is full rank.

This function provides a formula interface to the lfe::compfactor() function to check for connectedness of the factors.

## Usage

```
con_check(data = NULL, formula = NULL, WW = TRUE, dropNA = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | A dataframe |
| `formula` | A formula with multiple factor names in the dataframe, like `y ~ f1 + f2 + f3` |
| `WW` | Pass-through argument to `compfactor` |
| `dropNA` | If TRUE, observed data that are `NA` will be dropped. |

## Value

A vector with integers representing the group membership of each observation.

## Author(s)

Kevin Wright

## References

None

## Examples

```
# In the data_eccleston dataframe, each pair of factors is connected.
con_check(data_eccleston, ~ row + trt)
con_check(data_eccleston, ~ col + trt)
con_check(data_eccleston, ~ row + col)
# But all three factors are COMPLETELY disconnected into 16 groups.
con_check(data_eccleston, ~ row + col + trt)
```

---

con_concur *View concurrence of two factors in a dataframe using a matrix plot.*

---

## Description

Draws a concurrence plot of 2 factors in a dataframe. For example, in a multi-environment yield trial (testing multiple crop varieties in multple environments) it is interesting to examine the balance of the testing pattern. For each pair of environments, how many genotypes are tested in both environments? The concurrence plot shows the amount of connectedness (number of varieties) of the environments with each other.

By default, missing values in the response are deleted.

Replicated combinations of the two factors are ignored. (This could be changed if someone has a need.)

## Usage

```
con_concur(
  data,
  formula,
  dropNA = TRUE,
  xlab = "",
  ylab = "",
  cex.x = 0.7,
  cex.y = 0.7,
  ...
)
```

## Arguments

| | |
|---|---|
| data | A dataframe |
| formula | A formula with multiple factor names in the dataframe, like y ~ f1 / f2. |
| dropNA | If TRUE, observed data that are NA will be dropped. |
| xlab | Label for x axis |
| ylab | Label for y axis |
| cex.x | Scale factor for x axis tick labels. Default 0.7. |
| cex.y | Scale factor for y axis tick labels Default 0.7. |
| ... | Other parameters passed to the levelplot() function. |

## Value

A lattice graphics object

## Author(s)

Kevin Wright

## References

None

## Examples

```
require(lattice)
bar = transform(lattice::barley, env=factor(paste(site,year)))
set.seed(123)
bar <- bar[sample(1:nrow(bar), 70, replace=TRUE),]
con_concur(bar, yield ~ variety / env, cex.x=0.75, cex.y=.3)
```

---

# con_filter *Filter a dataframe using two-way criteria to increase connectedness*

---

## Description

Traditional filtering (subsetting) of data is typically performed via some criteria based on the *columns* of the data.

In contrast, this function performs filtering of data based on the *joint* rows and columns of a matrix-view of two factors.

Conceptually, the idea is to re-shape two or three columns of a dataframe into a matrix, and then delete entire rows (or columns) of the matrix if there are too many missing cells in a row (or column).

The two most useful applications of two-way filtering are to:

1. Remove a factor level that has few interactions with another factor. This is especially useful in linear models to remove rare factor combinations.
2. Remove a factor level that has any missing interactions with another factor. This is especially useful with biplots of a matrix to remove rows or columns that have missing values.

A formula syntax is used to specify the two-way filtering criteria.

Some examples may provide the easiest understanding.

dat <- data.frame(state=c("NE","NE", "IA", "NE", "IA"), year=c(1,2,2,3,3), value=11:15)

When the 'value' column is re-shaped into a matrix it looks like:

state/year | 1 | 2 | 3 | NE | 11 | 12 | 14 | IA | | 13 | 15 |

Drop states with too much missing combinations. Keep only states with "at least 3 years per state" con_filter(dat, ~ 3 * year / state) NE 1 11 NE 2 12 NE 3 14

Keep only years with "at least 2 states per year" con_filter(dat, ~ 2 * state / year) NE 2 12 IA 2 13 NE 3 14 IA 3 15

If the constant number in the formula is less than 1.0, this is interpreted as a *fraction*. Keep only states with "at least 75% of years per state" con_filter(dat, ~ .75 * year / state)

It is possible to include another factor on either side of the slash "/". Suppose the data had another factor for political party called "party". Keep only states with "at least 2 combinations of party:year per state" con_filter(dat, ~ 2 * party:year / state)

If the formula contains a response variable, missing values are dropped first, then the two-way filtering is based on the factor combinations. con_filter(dat, value ~ 2 * state / year)

## Usage

```
con_filter(data, formula, verbose = TRUE, returndropped = FALSE)
```

## Arguments

| | |
|---|---|
| `data` | A dataframe |
| `formula` | A formula with two factor names in the dataframe that specifies the criteria for filtering, like `y ~ 2 * f1 / f2` |
| `verbose` | If TRUE, print some diagnostic information about what data is being deleted. (Similar to the 'tidylog' package). |
| `returndropped` | If TRUE, return the dropped rows instead of the kept rows. Default is FALSE. |

## Value

The original dataframe is returned, minus rows that are filtered out.

## Author(s)

Kevin Wright

## References

None.

## Examples

```
dat <- data.frame(
  gen = c("G3", "G4", "G1", "G2", "G3", "G4", "G5",
          "G1", "G2", "G3", "G4", "G5",
          "G1", "G2", "G3", "G4", "G5",
          "G1", "G2", "G3", "G4", "G5"),
  env = c("E1", "E1", "E1", "E1", "E1", "E1", "E1",
          "E2", "E2", "E2", "E2", "E2",
          "E3", "E3", "E3", "E3", "E3",
          "E4", "E4", "E4", "E4", "E4"),
  yield = c(65, 50, NA, NA, 65, 50, 60,
            NA, 71, 76, 80, 82,
            90, 93, 95, 102, 97,
```

```
            98, 102, 105, 130, 135))

# How many observations are there for each combination of gen*env?
with( subset(dat, !is.na(yield)) , table(gen,env) )

# Note, if there is no response variable, the two-way filtering is based
# only on the presence of the factor combinations.
dat1 <- con_filter(dat, ~ 4*env / gen)

# If there is a response variable, missing values are dropped first,
# then the two-way filtering is based on the factor combinations.

dat1 <- con_filter(dat, yield ~ 4*env/gen)
dat1 <- con_filter(dat, yield ~ 5*env/ gen)
dat1 <- con_filter(dat, yield ~ 6*gen/ env)
dat1 <- con_filter(dat, yield ~ .8 *env / gen)
dat1 <- con_filter(dat, yield ~ .8* gen / env)
dat1 <- con_filter(dat, yield ~ 7 * env / gen)
```

---

`con_view` *View connectedness of two factors in a dataframe using a levelplot*

---

## Description

If there is replication for the treatment combination cells in a two-way table, the replications are averaged together (or counted) before constructing the heatmap.

By default, rows and columns are clustered using the 'incidence' matrix of 0s and 1s.

The function checks to see if the cells in the heatmap form a connected set. If not, the data is divided into connected subsets and the subset group number is shown within each cell.

By default, missing values in the response are deleted.

Factor levels are shown along the left and bottom sides.

The number of cells in each column/row is shown along the top/right sides.

If the 2 factors are disconnected, the group membership ID is shown in each cell.

## Usage

```
con_view(
  data,
  formula,
  fun.aggregate = mean,
  xlab = "",
  ylab = "",
  cex.num = 0.75,
  cex.x = 0.7,
  cex.y = 0.7,
```

```
  col.regions = RedGrayBlue,
  cluster = "incidence",
  dropNA = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| data | A dataframe |
| formula | A formula with two (or more) factor names in the dataframe like yield ~ f1 *f2 |
| fun.aggregate | The function to use for aggregating data in cells. Default is mean. |
| xlab | Label for x axis |
| ylab | Label for y axis |
| cex.num | Disjoint group number. |
| cex.x | Scale factor for x axis tick labels. Default 0.7. |
| cex.y | Scale factor for y axis tick labels Default 0.7. |
| col.regions | Function for color regions. Default RedGrayBlue. |
| cluster | If "incidence", cluster rows and columns by the incidence matrix. If FALSE, no clustering is performed. |
| dropNA | If TRUE, observed data that are NA will be dropped. |
| ... | Other parameters passed to the levelplot() function. |

## Value

A lattice graphics object

## Author(s)

Kevin Wright

## Examples

```
require(lattice)
bar = transform(lattice::barley, env=factor(paste(site,year)))
set.seed(123)
bar <- bar[sample(1:nrow(bar), 70, replace=TRUE),]
con_view(bar, yield ~ variety * env, cex.x=1, cex.y=.3, cluster=FALSE)

# Create a heatmap of cell counts
w2b = colorRampPalette(c('wheat','black'))
con_view(bar, yield ~ variety * env, fun.aggregate=length,
  cex.x=1, cex.y=.3, col.regions=w2b, cluster=FALSE)

# Example from paper by Fernando et al. (1983).
set.seed(42)
data_fernando = transform(data_fernando,
```

```
  y=stats::rnorm(9, mean=100))
con_view(data_fernando, y ~ gen*herd, cluster=FALSE,
     main = "Fernando unsorted")
con_view(data_fernando, y ~ gen*herd, cluster=TRUE,
     main = "Fernando unsorted")

# Example from Searle (1971), Linear Models, p. 325
dat2 = transform(data_searle,
  y=stats::rnorm(nrow(data_searle)) + 100)

con_view(dat2, y ~ f1*f2, cluster=FALSE, main="data_searle unsorted")
con_view(dat2, y ~ f1*f2, main="data_searle clustered")
```

---

`data_eccleston` *Data from Eccleston & Russell*

---

## Description

Data from Eccleston & Russell

## Usage

```
data_eccleston
```

## Format

An object of class `data.frame` with 16 rows and 3 columns.

## Details

A dataframe with 3 treatment factors. Each pair of factors is connected, but the 3 factors are disconnected. The 'trt' column uses numbers to match Eccleston (Table 1, Design 1) and letters to match Foulley (Table 13.3).

## Source

Eccleston, J. and K. Russell (1975). Connectedness and orthogonality in multi-factor designs. Biometrika, 62, 341-345. https://doi.org/10.1093/biomet/62.2.341

## References

Foulley, J. L., Bouix, J., Goffinet, B., & Elsen, J. M. (1990). Connectedness in Genetic Evaluation. Advanced Series in Agricultural Sciences, 277–308. https://doi.org/10.1007/978-3-642-74487-7_13

## Examples

```
# Each pair of factors is connected
con_check(data_eccleston, ~ row + trt)
con_check(data_eccleston, ~ col + trt)
con_check(data_eccleston, ~ row + col)
# But all three factors are COMPLETELY disconnected
con_check(data_eccleston, ~ row + col + trt)

set.seed(42)
data_eccleston <- transform(data_eccleston,
  y = rnorm(nrow(data_eccleston), mean=100))
con_view(data_eccleston, y ~ row*col, xlab="row", ylab="col")
con_view(data_eccleston, y ~ row*trt, xlab="row", ylab="trt")
con_view(data_eccleston, y ~ col*trt, xlab="col", ylab="trt")
```

---

# data_fernando *Data from Fernando et al.*

---

## Description

Data from Fernando et al.

## Usage

```
data_fernando
```

## Format

An object of class `data.frame` with 9 rows and 2 columns.

## Details

A dataframe with 2 treatment factors. The treatment combinations form 2 disconnected groups.

## Source

Fernando et al. (1983). Identifying All Connected Subsets In A Two-Way Classification Without Interaction. J. of Dairy Science, 66, 1399-1402. Table 1. https://doi.org/10.3168/jds.S0022-0302(83)81951-1

## Examples

```
library(lfe)
cbind(data_fernando,
      .group=con_check(data_fernando, ~ gen + herd))
library(connected)
set.seed(42)
data_fernando = transform(data_fernando,
```

```
  y=stats::rnorm(9, mean=100))
con_view(data_fernando, y ~ gen*herd, cluster=FALSE,
  main = "Fernando unsorted")
con_view(data_fernando, y ~ gen*herd, main="Fernando clustered")
```

---

`data_searle` *Data from Searle*

---

## Description

Data from Searle

## Usage

```
data_searle
```

## Format

An object of class `data.frame` with 14 rows and 2 columns.

## Details

A dataframe with 2 treatment factors. The treatment combinations form 3 disconnected groups.

## Source

Searle (1971). Linear Models. Page 324.

## Examples

```
cbind(data_searle,
      .group=con_check(data_searle, ~ f1 + f2))
data_searle = transform(data_searle,
  y = rnorm(nrow(data_searle), mean=100))
con_view(data_searle, y ~ f1*f2, cluster=FALSE, main="Searle unsorted")
con_view(data_searle, y ~ f1*f2, main="Searle clustered")
```

`data_student` *Simulated Data of Student Test Scores*

## Description

Simulated Data of Student Test Scores

## Usage

```
data("data_student")
```

## Format

A data frame with 41 observations on the following 4 variables.

`student` student ID

`class` class/subject

`test1` score on test 1

`test2` score on test 2

## Details

This simulated data imagines the following scenario:

In a school, 12 students (A-M) were tested in 6 classes (math, chemistry, physics, art, horticulture, welding).

Not all students were enrolled in all classes.

In each class, students were given test 1, then later test 2. Test scores could be 0-100.

(Using 2 tests is typical for learning assessment, measuring intervention effectiveness, comparison of test forms, etc.)

Some students missed class on the day of a test and had no score for that test.

Student B felt ill during art test 1 and was allowed to re-take test 1 (both test 1 scores are included and the same test 2 score was used).

## Source

None

## References

None

## Examples

```
data(data_student)
```

## data_tosh *Data from Tosh*

### Description

Data from Tosh

### Usage

```
data_tosh
```

### Format

An object of class `data.frame` with 15 rows and 3 columns.

### Details

A dataframe with 3 treatment factors. The treatment combinations form 2 disconnected groups.

### Source

Tosh, J. J., and J. W. Wilton. (1990). Degree of connectedness in mixed models. Proceedings of the 4th World Congress on Genetics applied to Livestock Production, 480-483. Page 481.

### Examples

```
cbind(data_tosh,
      .group=con_check(data_tosh, ~ a + b + c))
data_tosh = transform(data_tosh,
  y = rnorm(nrow(data_tosh), mean=100))
library(connected)
con_view(data_tosh, y ~ b * c)
```

## data_weeks1 *Data from Weeks & Williams example 1*

### Description

Data from Weeks & Williams example 1

### Usage

```
data_weeks1
```

### Format

An object of class `data.frame` with 16 rows and 3 columns.

### Details

A dataframe with 3 treatment factors. The treatment combinations are connected.

Note: This data is based on Table 1 of Weeks & Williams. Table 2 is missing treatment combination (1,2,4).

### Source

Weeks, David L. & Donald R. Williams (1964). A Note on the Determination of Connectedness in an N-Way Cross Classification. Technometrics, 6:3, 319-324. Table 1. http://dx.doi.org/10.1080/00401706.1964.10490188

### Examples

```
library(lfe)
cbind(data_weeks1,
      .group=con_check(data_weeks1, ~ f1+f2+f3))
```

---

## `data_weeks2` *Data from Weeks & Williams example 2*

---

### Description

Data from Weeks & Williams example 2

### Usage

```
data_weeks2
```

### Format

An object of class `data.frame` with 62 rows and 3 columns.

### Details

A dataframe with 3 treatment factors. The treatment combinations form 4 disconnected groups.

Note: This data is based on Table 3 of Weeks & Williams. The groups defined in the text are missing some combinations.

### Source

Weeks, David L. & Donald R. Williams (1964). A Note on the Determination of Connectedness in an N-Way Cross Classification. Technometrics, 6:3, 319-324. Table 3. http://dx.doi.org/10.1080/00401706.1964.10490188

## Examples

```
library(lfe)
cbind(data_weeks2,
      .group=con_check(data_weeks2, ~f1 + f2 + f3))
```

# Index

∗ **datasets**
data_eccleston, 8
data_fernando, 9
data_searle, 10
data_student, 11
data_tosh, 12
data_weeks1, 12
data_weeks2, 13

con_check, 2
con_concur, 3
con_filter, 4
con_view, 6

data_eccleston, 8
data_fernando, 9
data_searle, 10
data_student, 11
data_tosh, 12
data_weeks1, 12
data_weeks2, 13