CMI_ABCD_JIVE

Lucinda Sisk

1/12/2020

Drop empty columns, combine data to ensure same IDs

```
# Drop ID columns; drop all columns that sum to 0
brain_datano0 <- brain_data %>% select(-c("subid")) %>% select_if(colSums(.) >
    0) %>% mutate(subid = brain_data$subid)

# Standardize (scale and center) brain data
brain_data_scaled <- brain_datano0 %>% standardize()

# Standardize (scale and center) phenotypic data
pheno_data_scaled <- pheno_data %>% standardize() %>% rename(subid = "subjectkey")

# Merge data frames to ensure they are in same order
combined_df <- right_join(pheno_data_scaled, brain_data_scaled, by = "subid")

#############################################################################################

new_pheno <- combined_df %>% select("interview_age", "cbcl_scr_syn_anxdep_t":"trauma_num") %>%
    mutate_all(funs(replace_na(., 0)))  #Replace NAs with 0's --> check if OK
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
new_brain <- combined_df %>% select("area_L_1":"thickness_R_20442") %>% mutate_all(funs(replace_na(.,
    0)))  #Replace NAs with 0's --> check if OK



# Convert non-continuous data to factors new_pheno$sex <-
# as.factor(new_pheno$sex) new_pheno$site_id_l <-
# as.factor(new_pheno$site_id_l)

# Transpose so IDs are columns
new_pheno_mat <- t(new_pheno)
new_brain_mat <- t(new_brain)
```

Prepare matrices, run jive

- Data requirements: A list of two or more linked data matrices on which to perform the JIVE decomposition. These matrices must have the same column dimension, which is assumed to be common.

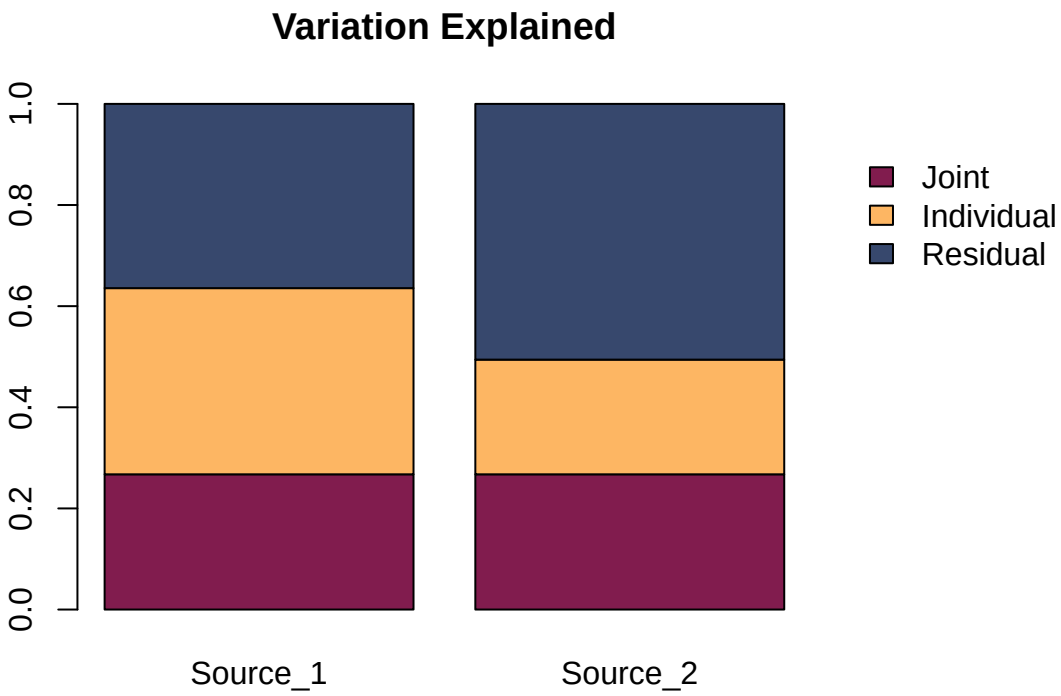
```
# data1 <- list(area_data, myelin_data, stress_data, thick_data, cbcl_data)
data <- list(new_pheno_mat, new_pheno_mat)
```

```
# Run JIVE analysis

# Estimate JIVE ranks based on permutation testing (best validated)
# Row-orthogonality enforced between the joint and individual estimates and
# also between each individual estimate. Compute ranks
(try(cmi_jive_result <- jive(data, scale = TRUE)))
```

```
## Estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 8 8
## JIVE algorithm converged after  93  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 10 10
## JIVE algorithm converged after  42  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 9
## JIVE algorithm converged after  47  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 10
## JIVE algorithm converged after  189  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 10 9
## JIVE algorithm converged after  174  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 9
## JIVE algorithm converged after  47  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 10
## JIVE algorithm converged after  189  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 9
## JIVE algorithm converged after  47  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 10 10
## JIVE algorithm converged after  42  iterations.
## Re-estimating  joint and individual ranks via permutation...
## Running JIVE algorithm for ranks:
## joint rank: 1 , individual ranks: 9 10
## JIVE algorithm converged after  189  iterations.
```

```
## Final joint rank: 1 , final individual ranks: 9 10
```

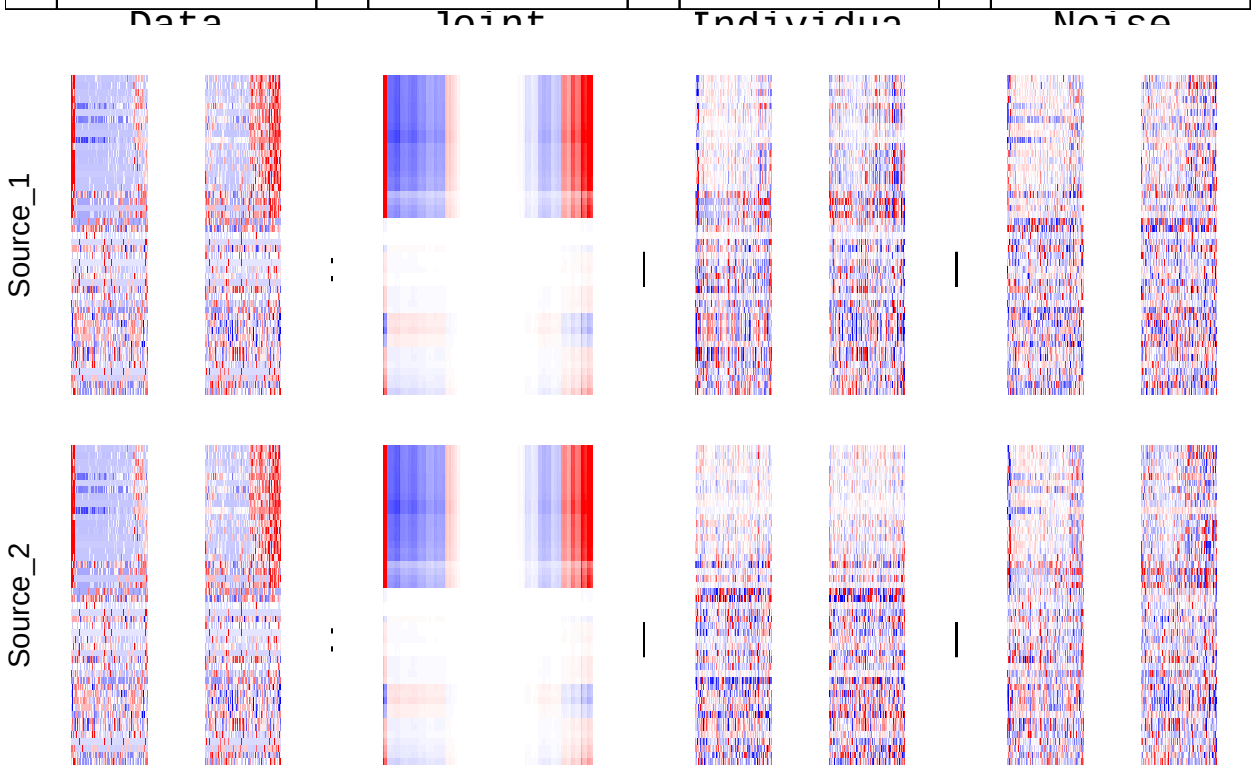
```
# Get Results
result_joint_rank <- cmi_jive_result$rankJ
result_individ_rank <- cmi_jive_result$rankA

# Plot variance explained by individual and joint ranks, and noise
(try(cmi_jive_var <- showVarExplained(cmi_jive_result, col = c("#811c4e", "#fdb663",
    "#37486d"))))
```




```
## $rect
## $rect$w
## [1] 0.6972646
##
## $rect$h
## [1] 0.192
##
## $rect$left
## [1] 0.05
##
## $rect$top
## [1] 0.8
##
##
## $text
## $text$x
## [1] 0.2942462 0.2942462 0.2942462
##
## $text$y
## [1] 0.752 0.704 0.656
```

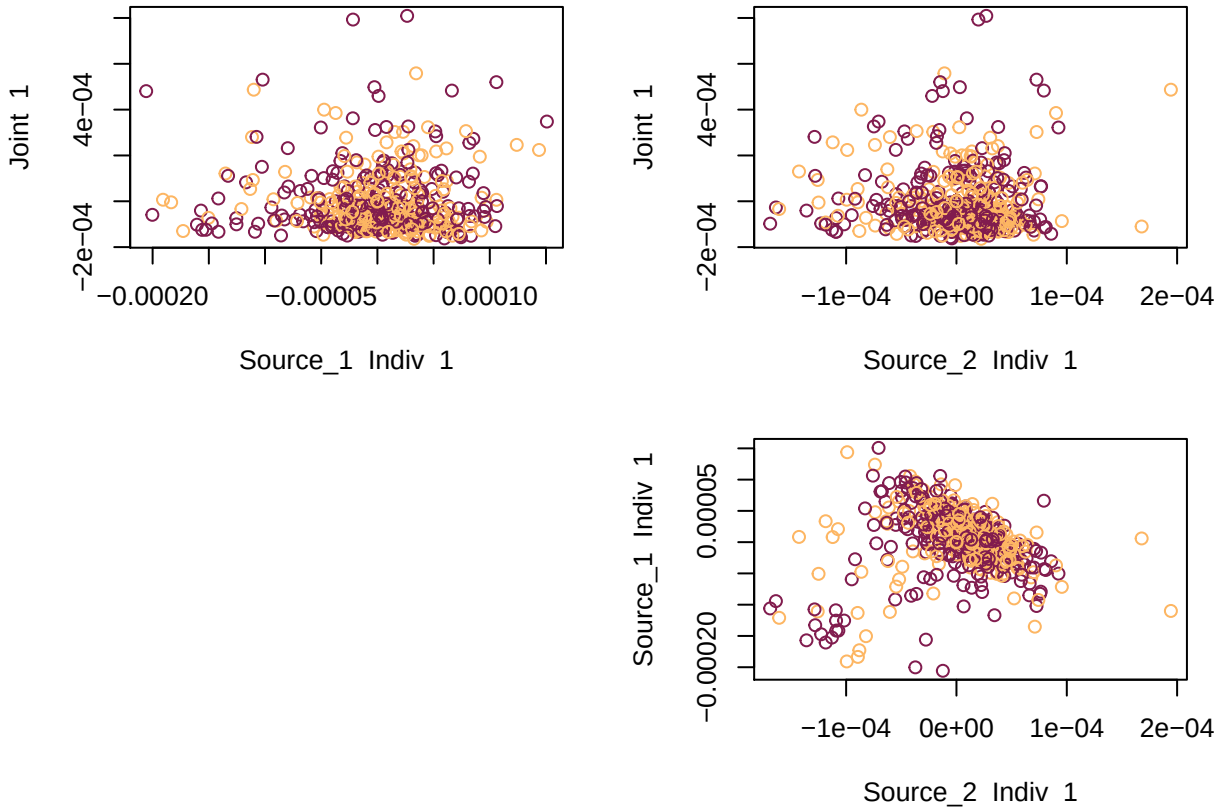
```
# Plot heatmaps of results
(try(cmi_jive_heatmaps <- showHeatmaps(cmi_jive_result, order_by = 0, show_all = TRUE)))
```

22	2	13	5	16	1	19	10
23	3	14	6	17	8	20	11
24	4	15	7	18	9	21	12




```
## NULL
```

```r
# Plot PCA
try(cmi_jive_pca <- showPCA(cmi_jive_result, n_joint = 1, n_indiv = c(1, 1),
    Colors = c("#811c4e", "#fdb663")))
```




```r
# No clustering efects apparent
```

Results:

- Joint Rank: 1
- Individual Ranks: 9, 10

```r
# ##Save images #Save variance images png(paste(here,
# '/CMI_JIVE_VarExplained.png', sep=''),height=300,width=450) cmi_jive_var
# dev.off() #Save heatmaps png(paste(here, '/CMI_JIVE_Heatmaps.png',
# sep=''),height=465,width=705) cmi_jive_heatmaps dev.off()
```