Lecture 6
Split-Apply-Combine

BIOF 339

October 17, 2017

Goals for today

- Learn how to merge data sets
- Learn how to reshape data sets from wide to long
- Split-apply-combine
    - Split a dataset into a list of several datasets
    - Do something to each dataset
    - Put the results back together
- Use it for
    - Running tests for many variables
    - Visualizing data with p-value annotation

The data for today

This data set is taken from a breast cancer proteome database available here and modified for this exercise.

- Clinical data: CSV|XLSX
- Proteome data: CSV|XLSX

The data for today

```
# Excel
library(readxl)
clinical_data <- read_excel('lecture6_data/BreastCancer_Clinical.xlsx')

# CSV
clinical_data <- read.csv('lecture6_data/BreastCancer_Clinical.csv',
                          stringsAsFactors=F)
```

```
#    Complete.TCGA.ID Gender Age.at.Initial.Pathologic.Diagnosis ER.Status
#  1     TCGA-A2-A0CM FEMALE                                  40  Negative
#  2     TCGA-BH-A18Q FEMALE                                  56  Negative
#    PR.Status HER2.Final.Status Tumor Tumor..T1.Coded Node Node.Coded
#  1  Negative          Negative    T2         T_Other   N0   Negative
#  2  Negative          Negative    T2         T_Other   N1   Positive
#    Metastasis Metastasis.Coded AJCC.Stage Converted.Stage
#  1         M0         Negative  Stage IIA       Stage IIA
#  2         M0         Negative  Stage IIB   No_Conversion
#    Survival.Data.Form Vital.Status Days.to.Date.of.Last.Contact
#  1           followup     DECEASED                          754
#  2         enrollment     DECEASED                         1692
```

The data for today

```
expression_data <- read.csv('lecture6_data/BreastCancer_Expression.csv',
                            stringsAsFactors=F)
head(expression_data[,1:5],2)
```

```
#          TCGA_ID NP_958782 NP_958785 NP_958786 NP_000436
#  1 TCGA-AO-A12D  1.096131  1.111370  1.111370  1.107561
#  2 TCGA-C8-A131  2.609943  2.650422  2.650422  2.646374
```

Merging data

Note that

1. The names of the ID columns are different
2. They are in different orders

The `merge` function takes care of both of these issues

Merging data

```
final_data = merge(clinical_data,
                   expression_data[,1:5],
                   by.x = 'Complete.TCGA.ID',
                   by.y = 'TCGA_ID')
head(final_data,2)
```

```
#    Complete.TCGA.ID Gender Age.at.Initial.Pathologic.Diagnosis ER.Status
#  1     TCGA-A2-A0CM FEMALE                                  40  Negative
#  2     TCGA-A2-A0D2 FEMALE                                  45  Negative
#    PR.Status HER2.Final.Status Tumor Tumor..T1.Coded Node Node.Coded
#  1  Negative          Negative    T2         T_Other   N0   Negative
#  2  Negative          Negative    T2         T_Other   N0   Negative
#    Metastasis Metastasis.Coded AJCC.Stage Converted.Stage
#  1         M0         Negative  Stage IIA       Stage IIA
#  2         M0         Negative  Stage IIB       Stage IIA
#    Survival.Data.Form Vital.Status Days.to.Date.of.Last.Contact
#  1           followup     DECEASED                          754
#  2           followup       LIVING                         1027
#    Days.to.date.of.Death OS.event OS.Time PAM50.mRNA
#  1                   754        1     754 Basal-like
```

Merging data

```
dim(clinical_data)
```

```
#  [1] 77 30
```

```
dim(expression_data)
```

```
#  [1] 83 11
```

```
dim(final_data)
```

```
#  [1] 80 34
```

Note that we have extra rows, which usually means **duplication of rows/ids**. Something to check on.

Reshaping data

R usually can split data on rows.

If we want to split on variables (columns), we have to transform the data from *wide* to *long* so that each row is the data for one individual *for one variable*.

First, of course, we have to select the variables we want to include.

Selecting variables

```
head(final_data,2)
```

```
#    Complete.TCGA.ID Gender Age.at.Initial.Pathologic.Diagnosis ER.Status
#  1     TCGA-A2-A0CM FEMALE                                  40  Negative
#  2     TCGA-A2-A0D2 FEMALE                                  45  Negative
#    PR.Status HER2.Final.Status Tumor Tumor..T1.Coded Node Node.Coded
#  1  Negative          Negative    T2         T_Other   N0   Negative
#  2  Negative          Negative    T2         T_Other   N0   Negative
#    Metastasis Metastasis.Coded AJCC.Stage Converted.Stage
#  1         M0         Negative  Stage IIA       Stage IIA
#  2         M0         Negative  Stage IIB       Stage IIA
#    Survival.Data.Form Vital.Status Days.to.Date.of.Last.Contact
#  1           followup     DECEASED                          754
#  2           followup       LIVING                         1027
#    Days.to.date.of.Death OS.event OS.Time PAM50.mRNA
#  1                   754        1     754 Basal-like
#  2                    NA        0    1027 Basal-like
#    SigClust.Unsupervised.mRNA SigClust.Intrinsic.mRNA miRNA.Clusters
#  1                        -12                     -13              4
#  2                        -12                     -13              4
```

Selecting variables

```
library(dplyr)
selected_data <- select(final_data, Complete.TCGA.ID, ER.Status,
                        starts_with('NP'))
head(selected_data,2)
```

```
#    Complete.TCGA.ID ER.Status NP_958782 NP_958785 NP_958786  NP_000436
#  1     TCGA-A2-A0CM  Negative 0.6834035 0.6944241 0.6980976 0.68707705
#  2     TCGA-A2-A0D2  Negative 0.1074909 0.1041645 0.1074909 0.09751166
```

Reshaping data

We have two goals here:

1. Keep the ER status aligned with each expression level, so we would need to repeat the column
2. Make each row have the ER status for an individual and the corresponding expression for one protein

Reshaping data

```
library(reshape2)

reshaped_data <- melt(selected_data,
                      id.vars=c('Complete.TCGA.ID','ER.Status'))
head(reshaped_data, 5)
```

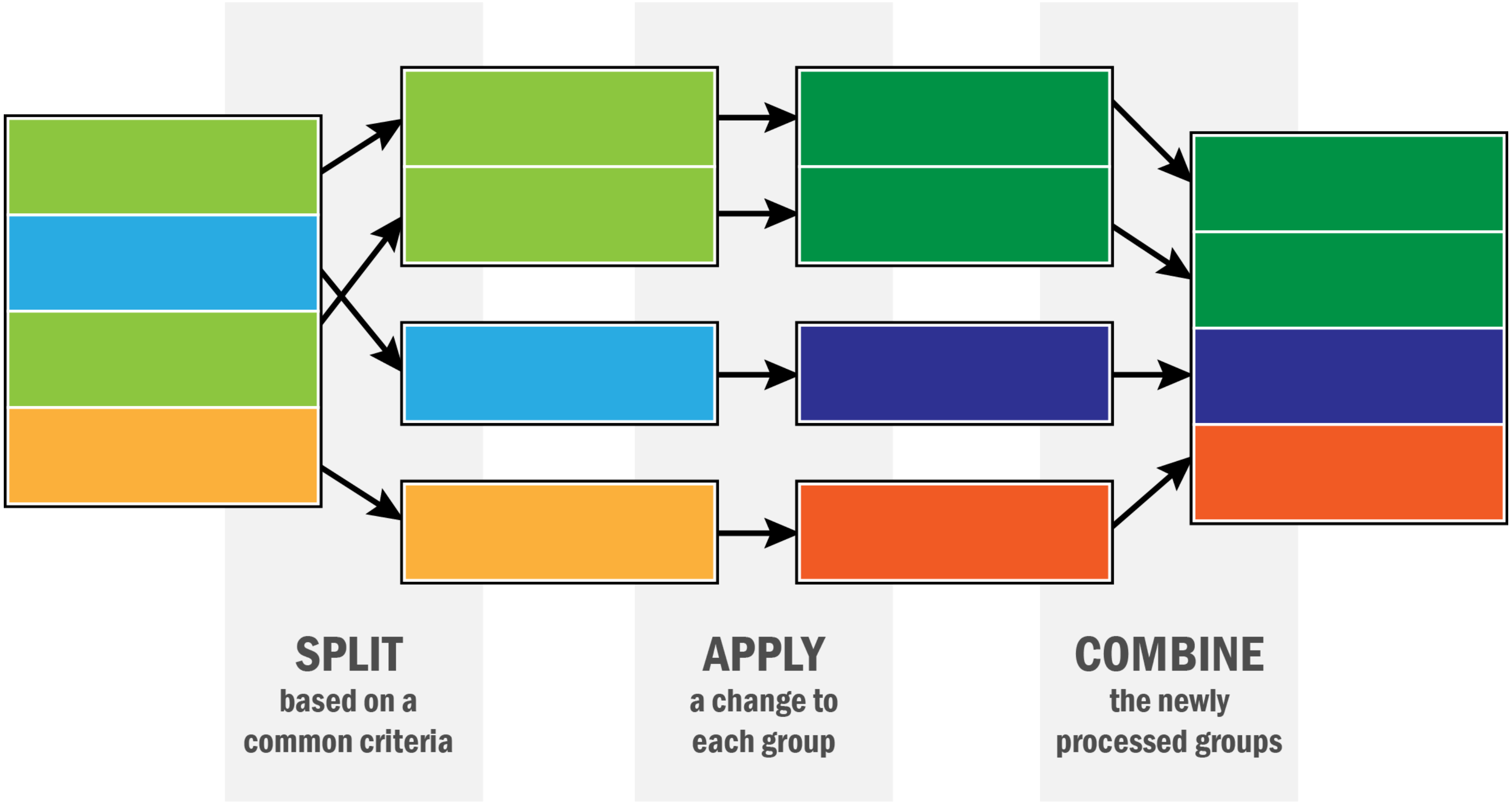
```
#    Complete.TCGA.ID ER.Status  variable      value
#   1     TCGA-A2-A0CM  Negative NP_958782  0.6834035
#   2     TCGA-A2-A0D2  Negative NP_958782  0.1074909
#   3     TCGA-A2-A0EQ  Negative NP_958782 -0.9126703
#   4     TCGA-A2-A0EV  Positive NP_958782  0.4529859
#   5     TCGA-A2-A0EX  Positive NP_958782  1.1851082
```

Reshaping data

```
reshaped_data <- arrange(reshaped_data, Complete.TCGA.ID) # from dplyr
head(reshaped_data,5)
```

```
#   Complete.TCGA.ID ER.Status  variable     value
#  1     TCGA-A2-A0CM  Negative NP_958782 0.6834035
#  2     TCGA-A2-A0CM  Negative NP_958785 0.6944241
#  3     TCGA-A2-A0CM  Negative NP_958786 0.6980976
#  4     TCGA-A2-A0CM  Negative NP_000436 0.6870771
#  5     TCGA-A2-A0D2  Negative NP_958782 0.1074909
```

Split-apply-combine



Splitting data by protein

```
split_data <- split(reshaped_data, reshaped_data$variable)
class(split_data)
```

```
#  [1] "list"
```

```
length(split_data)
```

```
#  [1] 4
```

```
names(split_data)
```

```
#  [1] "NP_958782" "NP_958785" "NP_958786" "NP_000436"
```

Splitting data

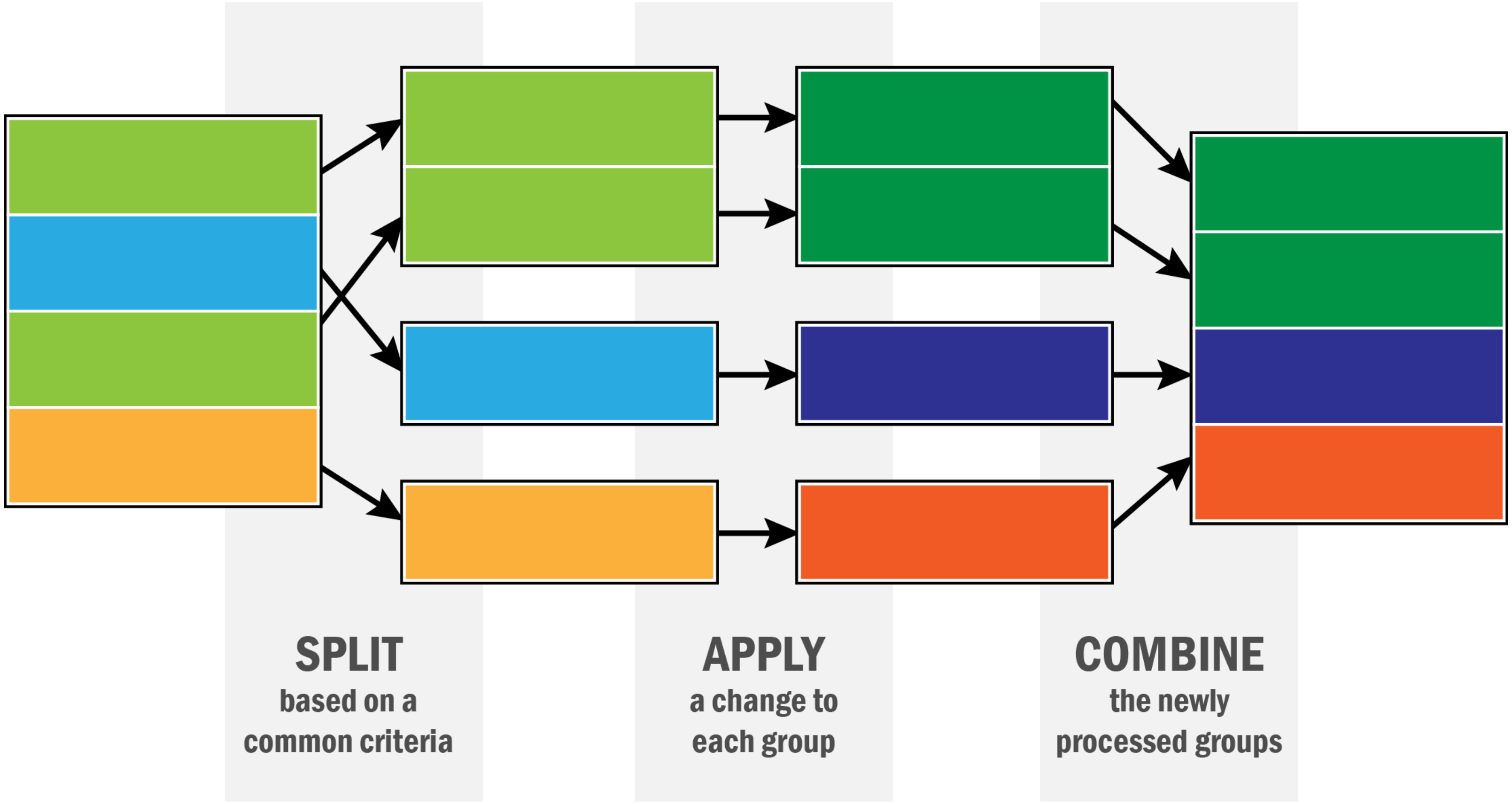
```
head(split_data[['NP_958782']],8)
```

```
#     Complete.TCGA.ID ER.Status  variable      value
#  1      TCGA-A2-A0CM  Negative NP_958782  0.6834035
#  5      TCGA-A2-A0D2  Negative NP_958782  0.1074909
#  9      TCGA-A2-A0EQ  Negative NP_958782 -0.9126703
#  13     TCGA-A2-A0EV  Positive NP_958782  0.4529859
#  17     TCGA-A2-A0EX  Positive NP_958782  1.1851082
#  21     TCGA-A2-A0EY  Positive NP_958782  1.1748810
#  25     TCGA-A2-A0SW  Positive NP_958782 -0.4877725
#  29     TCGA-A2-A0SX  Negative NP_958782 -0.3985598
```

```
class(split_data[['NP_958782']])
```

```
#  [1] "data.frame"
```

Applying a function to the split data



Applying a function to the split data

We're going to run t-tests to see if the expression for each protein differs by ER status

First we create a function that we'll run on every protein-specific data set

```
run_t_test <- function(d){
  test <- t.test(value ~ ER.Status, data=d)
  pvalue <- test$p.value
  return(pvalue)
}
```

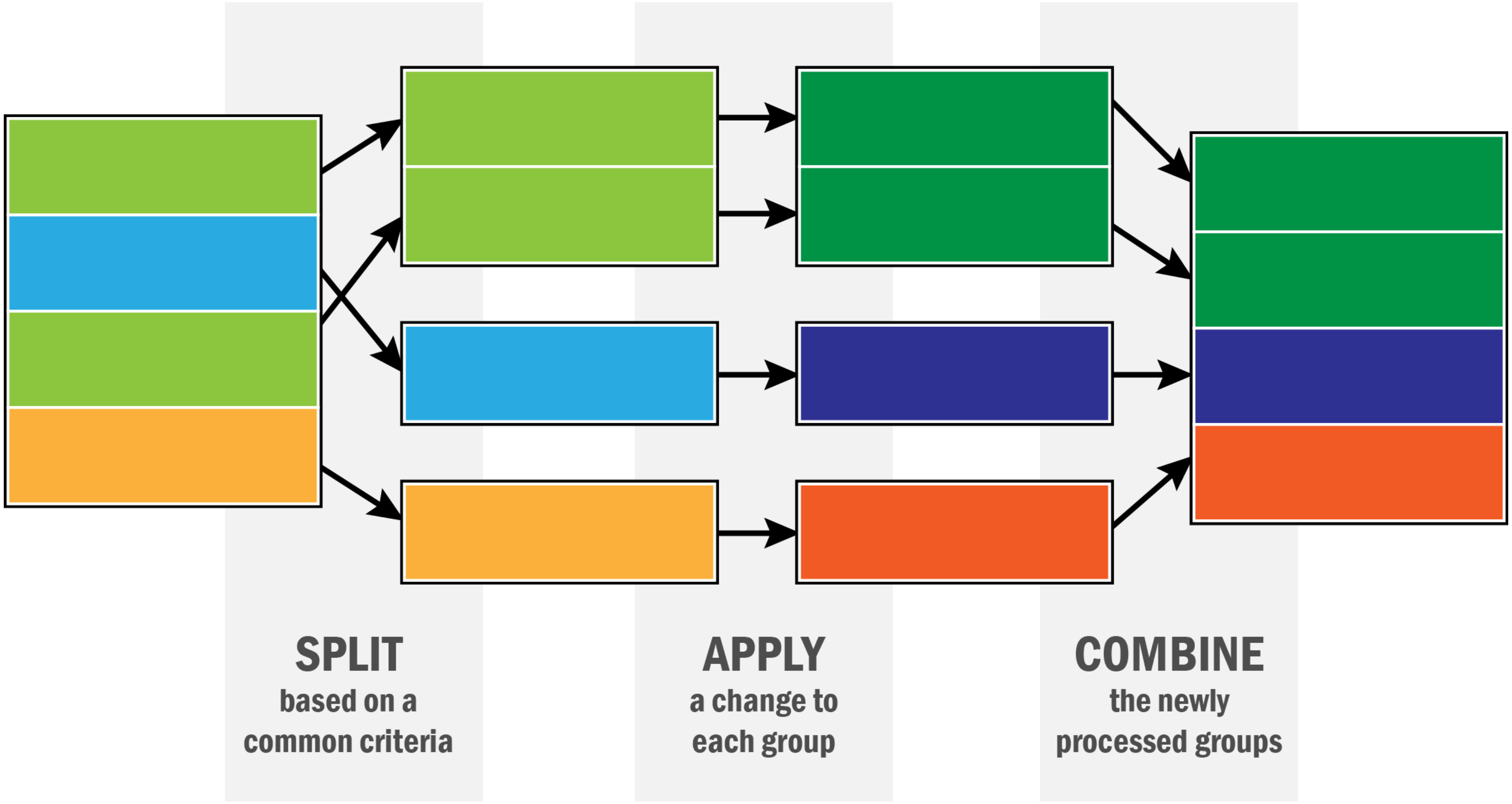
Applying a function to the split data

Now we can apply the same function to each element of the list of data frames using the `lapply` command

```
pvalues <- lapply(split_data, run_t_test)
pvalues
```

```
#  $NP_958782
#  [1] 0.5287476
#
#  $NP_958785
#  [1] 0.5243093
#
#  $NP_958786
#  [1] 0.519653
#
#  $NP_000436
#  [1] 0.5337173
```

Combining the data



Combining the data

```
pvalues_final <- unlist(pvalues)
pvalues_final
```

```
#  NP_958782 NP_958785 NP_958786 NP_000436
#  0.5287476 0.5243093 0.5196530 0.5337173
```

Combining the data

```
pvalues_final <- plyr::ldply(pvalues) # Use ldply from the plyr package
pvalues_final
```

```
#          .id        V1
#  1 NP_958782 0.5287476
#  2 NP_958785 0.5243093
#  3 NP_958786 0.5196530
#  4 NP_000436 0.5337173
```

The `ldply` function inputs a list and exports a data.frame with the elements concatenated if possible.

Plotting

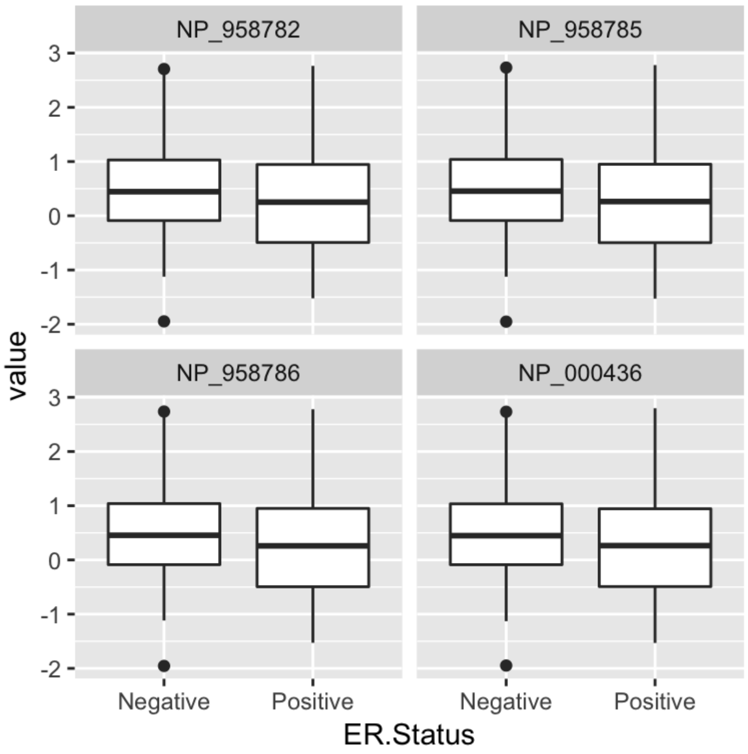
The data

```
head(reshaped_data, 5)
```

```
#    Complete.TCGA.ID ER.Status  variable     value
#  1     TCGA-A2-A0CM  Negative NP_958782 0.6834035
#  2     TCGA-A2-A0CM  Negative NP_958785 0.6944241
#  3     TCGA-A2-A0CM  Negative NP_958786 0.6980976
#  4     TCGA-A2-A0CM  Negative NP_000436 0.6870771
#  5     TCGA-A2-A0D2  Negative NP_958782 0.1074909
```

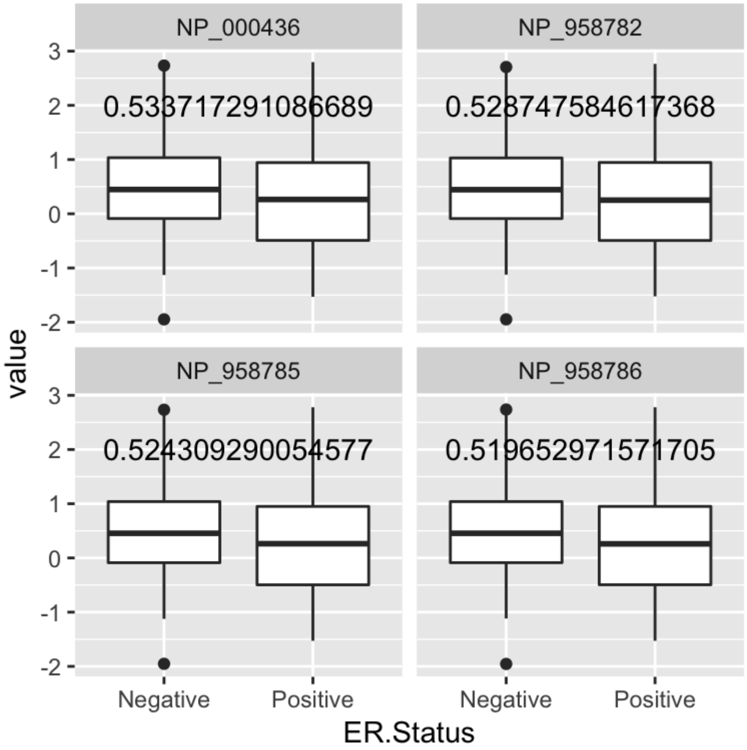
A panel of plots

```
library(ggplot2)
ggplot(reshaped_data, aes(x = ER.Status, y = value))+
  geom_boxplot()+
  facet_wrap(~variable, ncol=2)
```



Adding p-values to the plot

```
names(pvalues_final) <- c('variable','pvalue')
ggplot(reshaped_data, aes(x = ER.Status, y = value))+
  geom_boxplot()+
  facet_wrap(~variable, ncol=2)+
  geom_text(data=pvalues_final, aes(x=1.5, y=2, label=pvalue))
```



Formatting p-values

```
pvalues_final[,'pvalue'] <- format.pval(pvalues_final[,'pvalue'],
                                        digits=3)
ggplot(reshaped_data, aes(x = ER.Status, y = value))+
  geom_boxplot()+
  facet_wrap(~variable, ncol=2)+
  geom_text(data=pvalues_final, aes(x=1.5, y=2, label=pvalue))
```

Formatting p-values

```
pvalues_final <- mutate(pvalues_final,
                        pvalue = format.pval(pvalue, digits=3))
ggplot(reshaped_data, aes(x = ER.Status, y = value))+
  geom_boxplot()+
  facet_wrap(~variable, ncol=2)+
  geom_text(data=pvalues_final, aes(x=1.5, y=2, label=pvalue))
```

The dplyr package

Action words

`dplyr` basically gives you 6 actions to do on data.frame objects:

1. mutate: Change particular variables
2. select: Select (or deselect) variables
3. arrange: Order by some variables
4. filter: Select rows by some criteria
5. group_by: Group by some variable (so the split part of our exercise)
6. summarise: Summarise a variable using some function