Adjust data: assesing linear models

Carla Casanova

2022-06-21

```
library(datawizard)
library(RadAR)
library(dplyr)
library(ggplot2)
```

Load data

Load `RadAR` object with radiomic features:

```
load("/Users/carlacasanovasuarez/Documents/Master Bioinformatics UAB/Prácticas Radiomics/Radiomic featu

rdr_L1
```

```
## class: SummarizedExperiment
## dim: 1232 1778
## metadata(1): extractor
## assays(1): values
## rownames(1232): Elongation.original Flatness.original ...
##   Complexity.wavelet.LLL Strength.wavelet.LLL
## rowData names(4): feature_name image_type feature_description
##   feature_type
## colnames(1778): 23140000005 23140000006 ... 26658003992 26658003997
## colData names(200): filename sample_id ... RACE1.t3 change.fv1
```

Filter by image type, only radiomic features from original images will be used:

```
## filter by image types
rdr_filt_original <- filter_by_image_type(rdr = rdr_L1, image_type = c("original"))

rdr_filt_original
```

```
## class: SummarizedExperiment
## dim: 101 1778
## metadata(1): extractor
## assays(1): values
## rownames(101): Elongation.original Flatness.original ...
##   Complexity.original Strength.original
## rowData names(4): feature_name image_type feature_description
##   feature_type
## colnames(1778): 23140000005 23140000006 ... 26658003992 26658003997
## colData names(200): filename sample_id ... RACE1.t3 change.fv1
```

Store radiomic features and standardize data to allow comparisons between different features:

```
# Store radiomic features for original image type (101 features)
table_original <- assay(rdr_filt_original)

# Transpose before scaling in order to scale by features (otherwise scale by
# patients)
table_scaled <- t(scale(t(table_original)))
```

Prepare variables and radiomic table

Store original colnames and rownames:

```
# Store col and row names from the original table
features_original <- rownames(table_scaled)
patients_original <- colnames(table_scaled)
```

Prepare radiomic tables to use `adjust()` function. Variables to be adjusted must be placed as columns and col names starting with integers must be avoided:

```
table_scaled.T <- as.data.frame(t(table_scaled))

# Change invalid column names (white spaces, integers, etc)
names(table_scaled.T)[names(table_scaled.T) == "10Percentile.original"] <- "tenPercentile.original"
names(table_scaled.T)[names(table_scaled.T) == "90Percentile.original"] <- "ninetyPercentile.original"
```

Prepare variables of interest in a data frame:

```
dd.variables <- table_scaled.T %>%
    mutate(Center = as.factor(colData(rdr_filt_original)$CENTREID), Age = colData(rdr_filt_original)$AGE
        Smoker = as.factor(colData(rdr_filt_original)$SMOKER), Cough = as.factor(colData(rdr_filt_origi
        Country = as.factor(colData(rdr_filt_original)$COUNTRY), Sex = as.factor(colData(rdr_filt_origi
        GOLDCD = as.factor(colData(rdr_filt_original)$GOLDCD))
```

Now, remove variables with `NaN` values since prediction won't be comparable with original data (complete patients):

```
dd.variables = dd.variables[complete.cases(dd.variables), ]

dim(dd.variables)
```

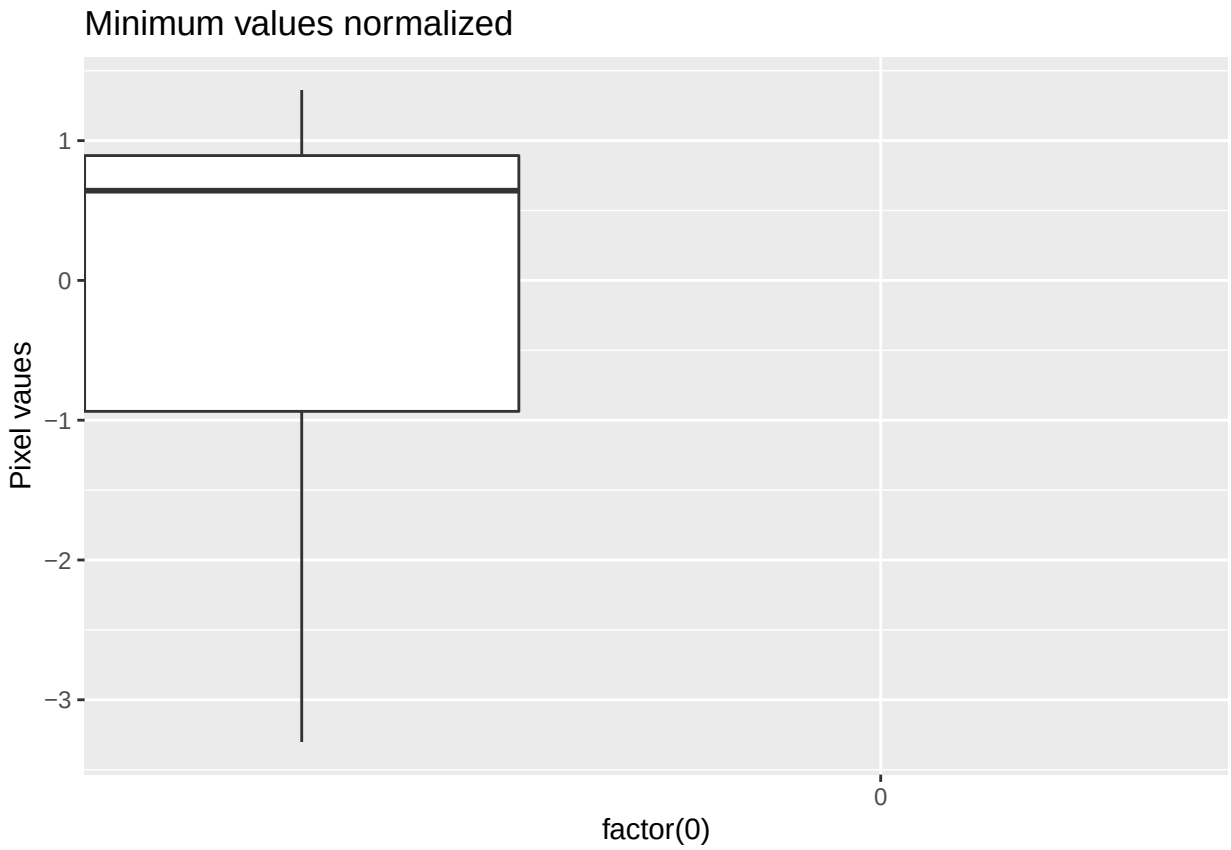
```
## [1] 1764  108
```

Check for outliers in variables of interest

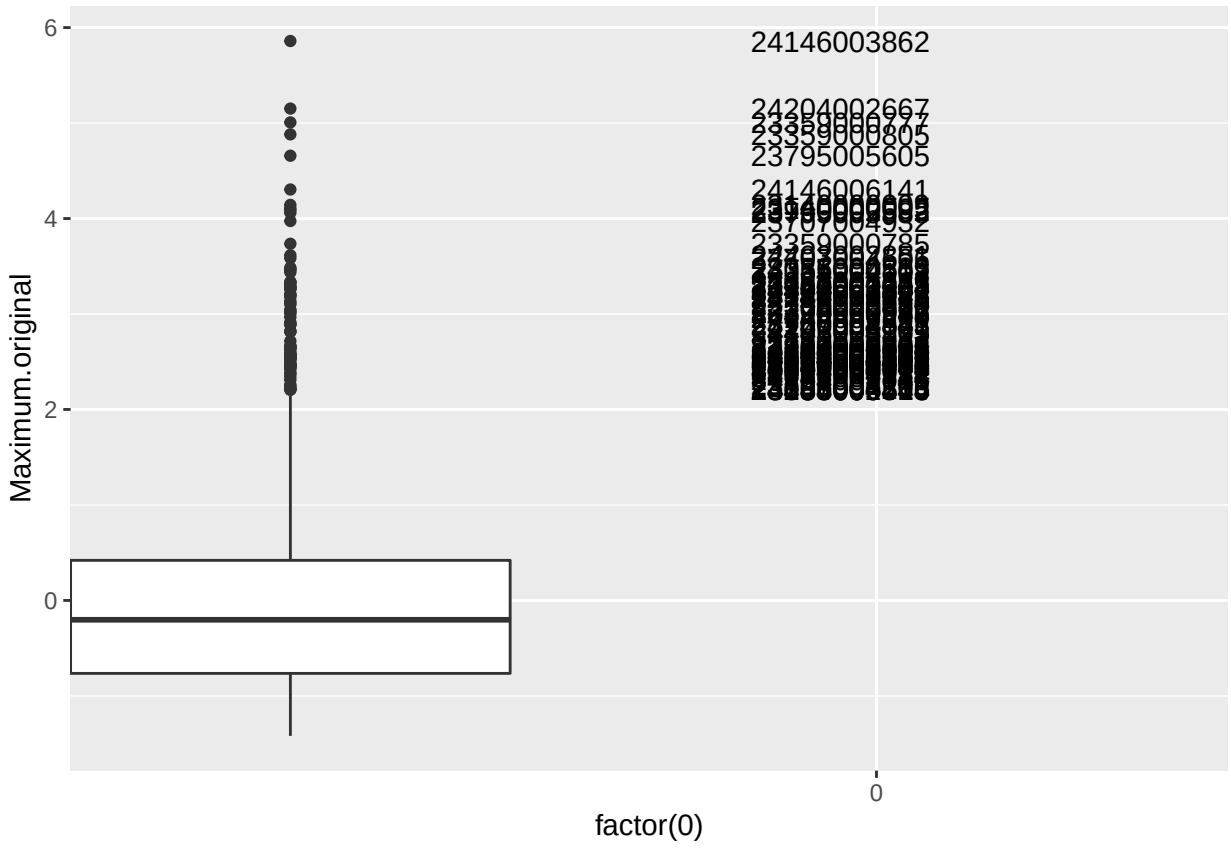
```
# Plotting boxplot
is_outlier <- function(x) {
    return(x < quantile(x, 0.25) - 1.5 * IQR(x) | x > quantile(x, 0.75) + 1.5 * IQR(x))
}
```

Check if there are outliers and their patient ID for `minimum`:

```
table_scaled.T %>%
    mutate(outlier = ifelse(is_outlier(Minimum.original), rownames(table_scaled.T),
        as.numeric(NA))) %>%
    ggplot(., aes(y = Minimum.original)) + ylab("Pixel vaues") + ggtitle("Minimum values normalized") +
    geom_boxplot() + geom_text(aes(x = factor(0), label = outlier), na.rm = TRUE,
    hjust = +0.7)
```



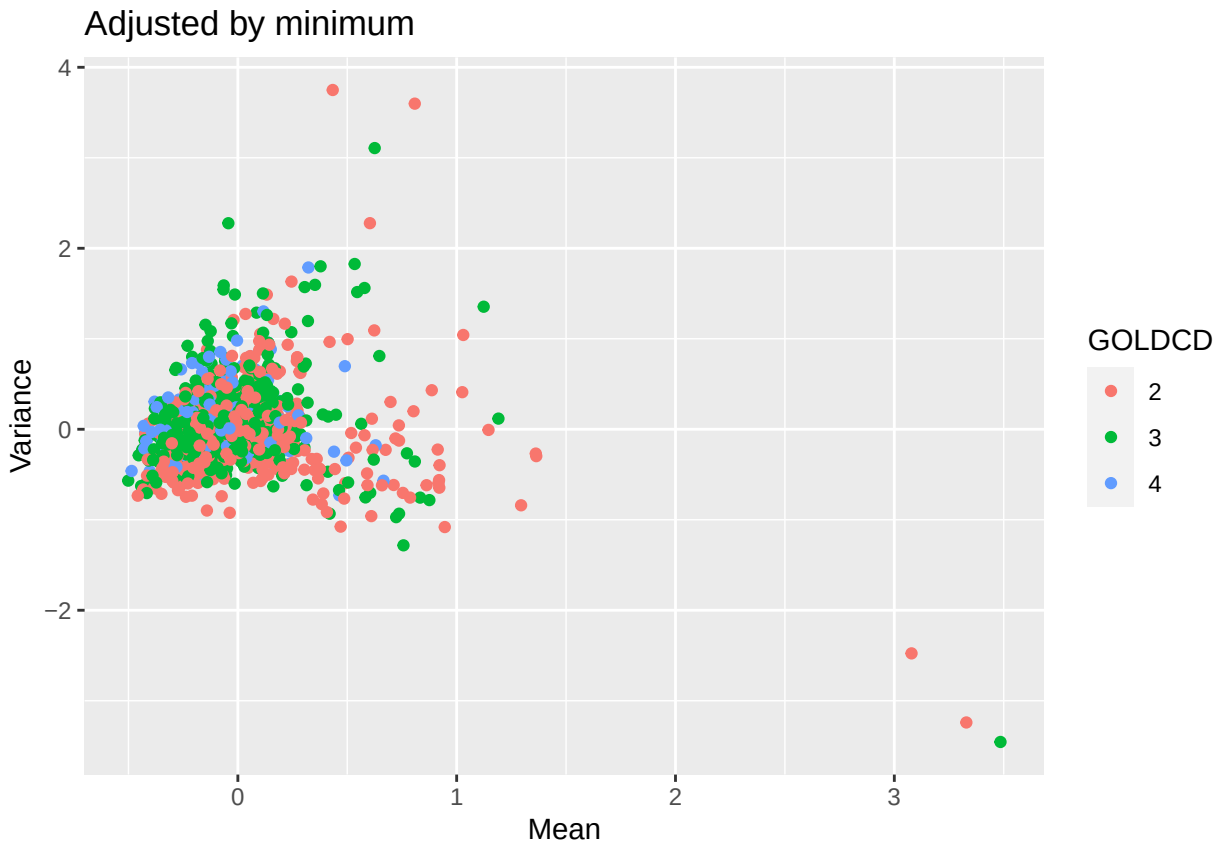

```
table_scaled.T %>%
    mutate(outlier = ifelse(is_outlier(Maximum.original), rownames(table_scaled.T),
        as.numeric(NA))) %>%
    ggplot(., aes(y = Maximum.original)) + geom_boxplot() + geom_text(aes(x = factor(0),
    label = outlier), na.rm = TRUE, hjust = +0.7)
```



Adjust

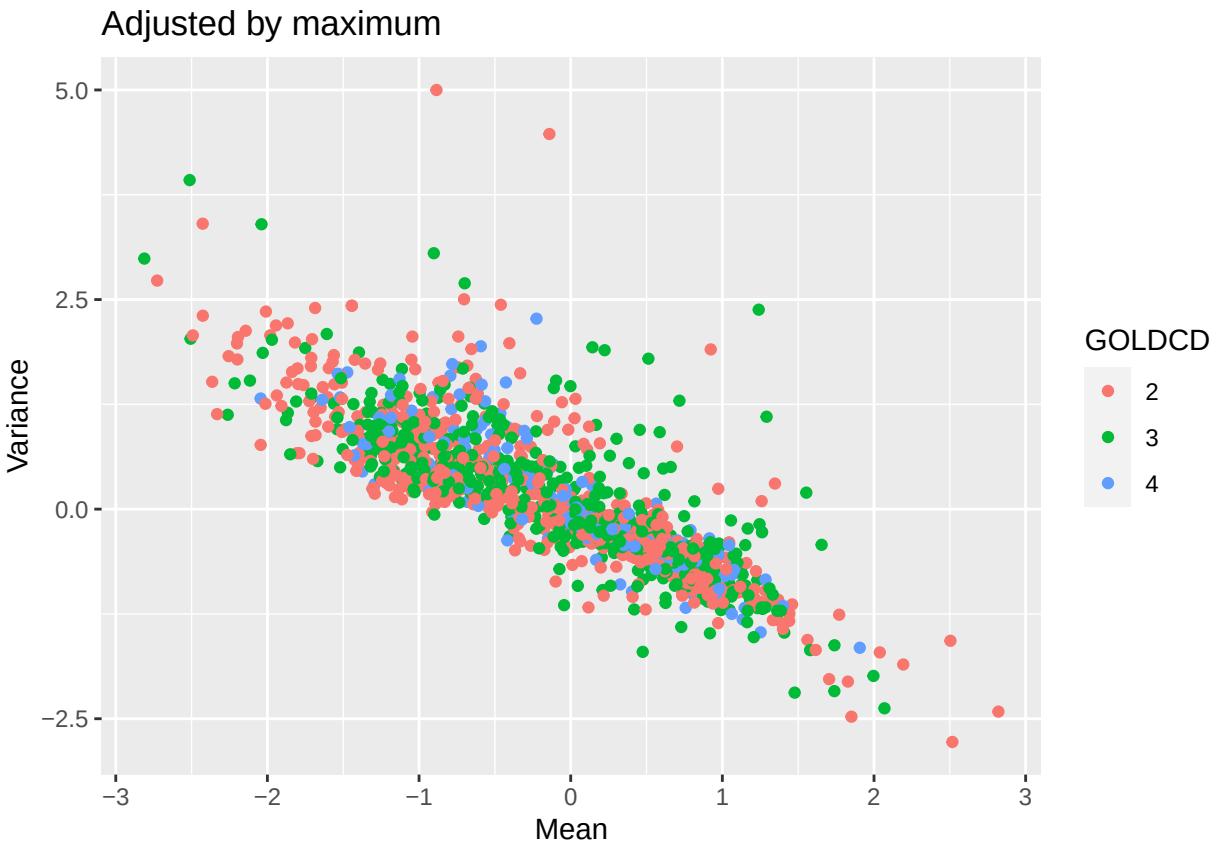
Adjust by `minimum pixel value`:

```
# Adjust all data for the original image type You can adjust just some columns
# with: select = c('Mean.original', 'Variance.original')
clinical_features_min_vox_adj <- adjust(dd.variables, effect = "Minimum.original",
    keep_intercept = TRUE, exclude = c("Age", "Sex", "Smoker", "Cough", "Country",
        "GOLDCD"))

qplot(clinical_features_min_vox_adj[, "Mean.original"], clinical_features_min_vox_adj[,
    "Variance.original"], colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance",
    main = "Adjusted by minimum") + labs(colour = "GOLDCD")
```



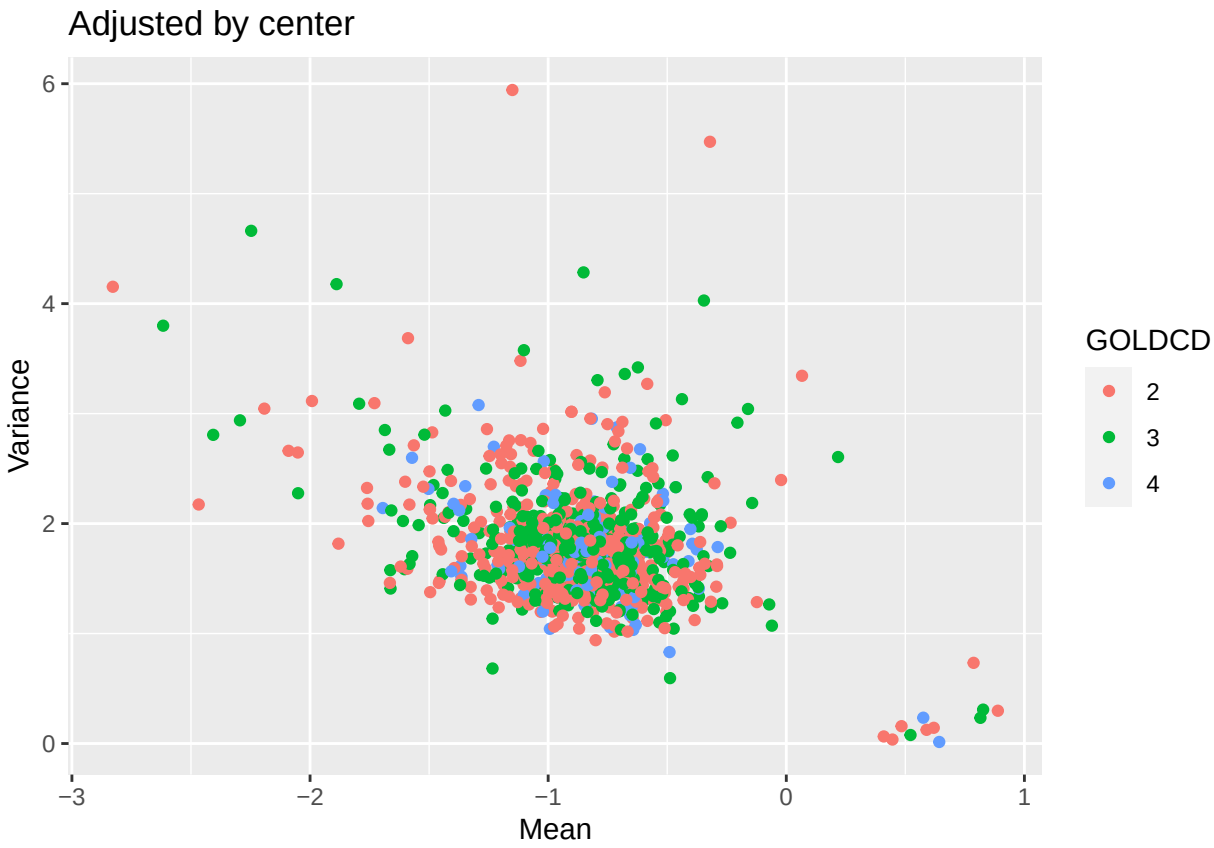
Adjust by `maximum pixel value`:

```
# Adjust all data for the original image type You can adjust just some columns
# with: select = c('Mean.original', 'Variance.original')
clinical_features_max_vox_adj <- adjust(dd.variables, effect = "Maximum.original",
    keep_intercept = TRUE, exclude = c("Age", "Sex", "Smoker", "Cough", "Country",
        "GOLDCD"))

qplot(clinical_features_max_vox_adj[, "Mean.original"], clinical_features_max_vox_adj[,
    "Variance.original"], colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance",
    main = "Adjusted by maximum") + labs(colour = "GOLDCD")
```



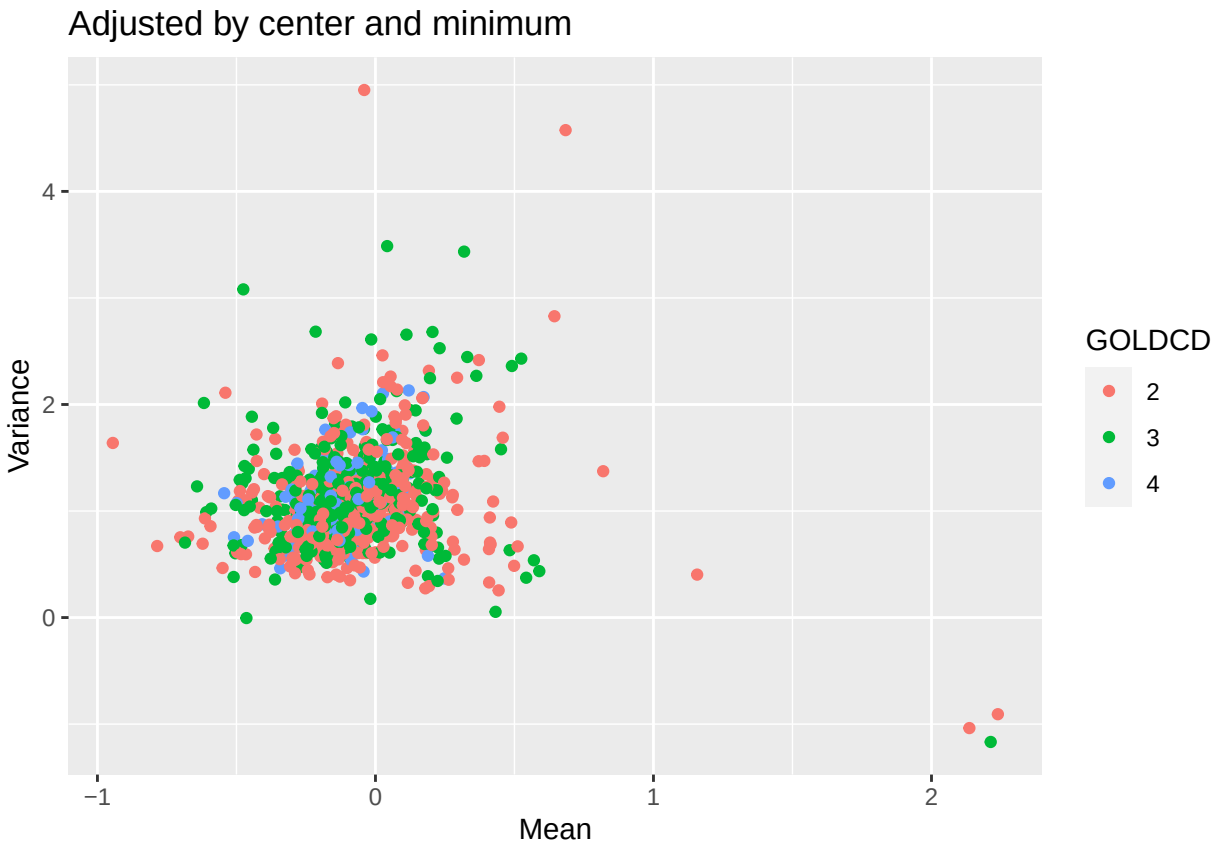
Adjust by `Center`:

```
# Adjust all data for the variable image type You can adjust just some columns
# with: select = c('Mean.original', 'Variance.original')
clinical_features_center_adj <- adjust(dd.variables, effect = "Center", keep_intercept = TRUE,
    exclude = c("Age", "Sex", "Smoker", "Cough", "Country", "GOLDCD"))

qplot(clinical_features_center_adj[, "Mean.original"], clinical_features_center_adj[,
    "Variance.original"], colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance",
    main = "Adjusted by center") + labs(colour = "GOLDCD")
```



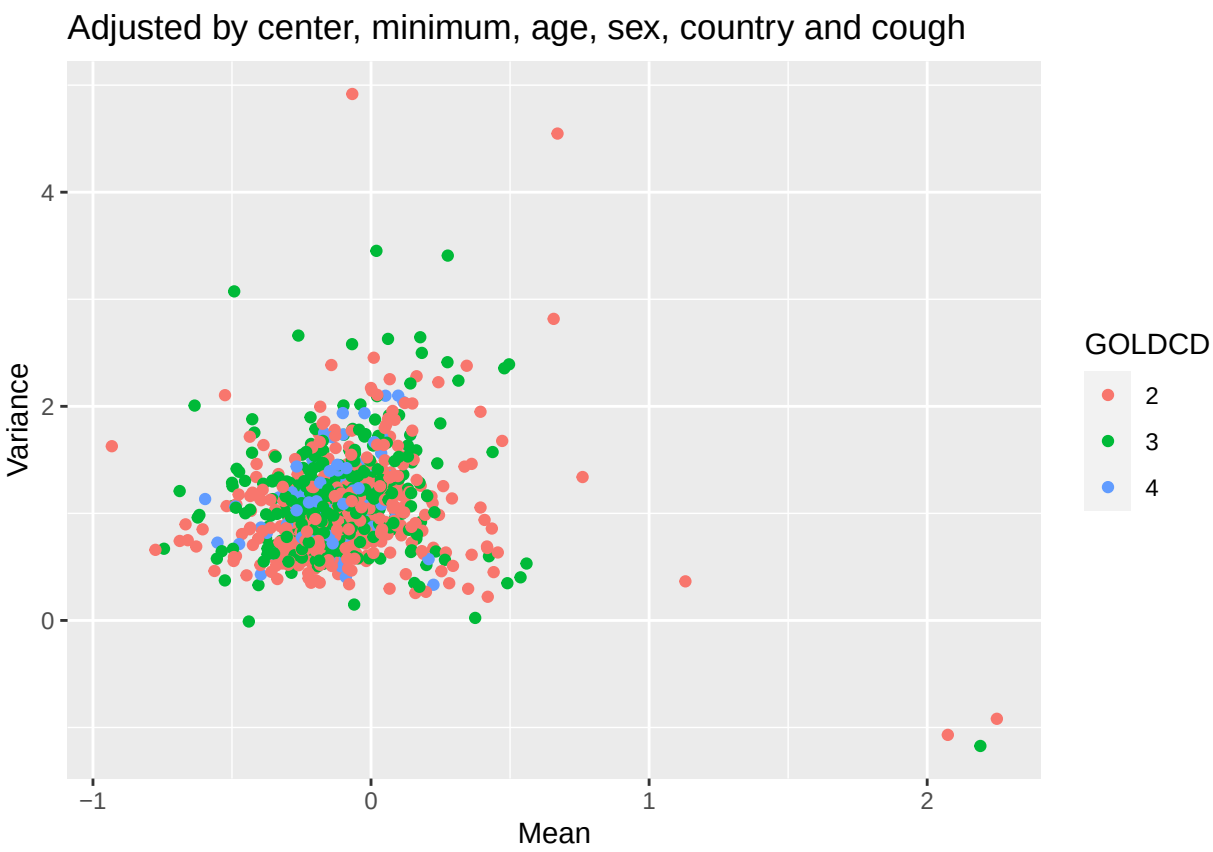
Adjust by `Center` and `minimum`:

```
# Adjust all data for the variable image type You can adjust just some columns
# with: select = c('Mean.original', 'Variance.original')
clinical_features_center_min_adj <- adjust(dd.variables, effect = c("Center", "Minimum.original"),
    keep_intercept = TRUE, exclude = c("Age", "Sex", "Smoker", "Cough", "Country"))

qplot(clinical_features_center_min_adj[, "Mean.original"], clinical_features_center_min_adj[,
    "Variance.original"], colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance",
    main = "Adjusted by center and minimum") + labs(colour = "GOLDCD")
```



Adjust by `Center` and `minimum` plus some other covariates (age, sex, etc):

```
# Adjust all data for the variable image type You can adjust just some columns
# with: select = c('Mean.original', 'Variance.original')
clinical_features_multi_adj <- adjust(dd.variables, effect = c("Center", "Minimum.original",
    "Sex", "Country", "Cough"), keep_intercept = TRUE, exclude = c("Age", "Sex",
    "Smoker", "Cough", "Country"))

qplot(clinical_features_multi_adj[, "Mean.original"], clinical_features_multi_adj[,
    "Variance.original"], colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance",
    main = "Adjusted by center, minimum, age, sex, country and cough") + labs(colour = "GOLDCD")
```



Asses best adjustment

Prepare functions to asses linear models: `RSS` and `RMSE`.

```
# RSS se ajusta mejor si es más bajo
rss <- function(fitted, actual) {
    sum((fitted - actual)^2)
}

# RMSE se ajusta mejor si es más bajo
rmse <- function(fitted, actual) {
    sqrt(mean((fitted - actual)^2))
}
```

Create the linear regression models:

```
# By center
simple_model_mean_center <- lm(Mean.original ~ Center, data = dd.variables)

# By minimum
simple_model_mean_min <- lm(Mean.original ~ Minimum.original, data = dd.variables)

# By minimum
simple_model_mean_max <- lm(Mean.original ~ Maximum.original, data = dd.variables)

# By center and minimum
model_mean_center_min <- lm(Mean.original ~ Minimum.original + Center, data = dd.variables)
```

```
# Multi
model_mean_multi <- lm(Mean.original ~ Minimum.original + Center + Sex + Age + Cough,
    data = dd.variables)
```

Asses all three models:

```
# RSS
rss(fitted(model_mean_multi), dd.variables$Mean.original)
```

```
## [1] 57.40919
```

```
rss(fitted(model_mean_center_min), dd.variables$Mean.original)
```

```
## [1] 58.41367
```

```
rss(fitted(simple_model_mean_center), dd.variables$Mean.original)
```

```
## [1] 141.3399
```

```
rss(fitted(simple_model_mean_min), dd.variables$Mean.original)
```

```
## [1] 110.5897
```

```
rss(fitted(simple_model_mean_max), dd.variables$Mean.original)
```

```
## [1] 1130.321
```

```
# RMSE
rmse(fitted(model_mean_multi), dd.variables$Mean.original)
```

```
## [1] 0.180402
```

```
rmse(fitted(model_mean_center_min), dd.variables$Mean.original)
```

```
## [1] 0.1819734
```

```
rmse(fitted(simple_model_mean_center), dd.variables$Mean.original)
```

```
## [1] 0.283063
```

```
rmse(fitted(simple_model_mean_min), dd.variables$Mean.original)
```

```
## [1] 0.2503849
```

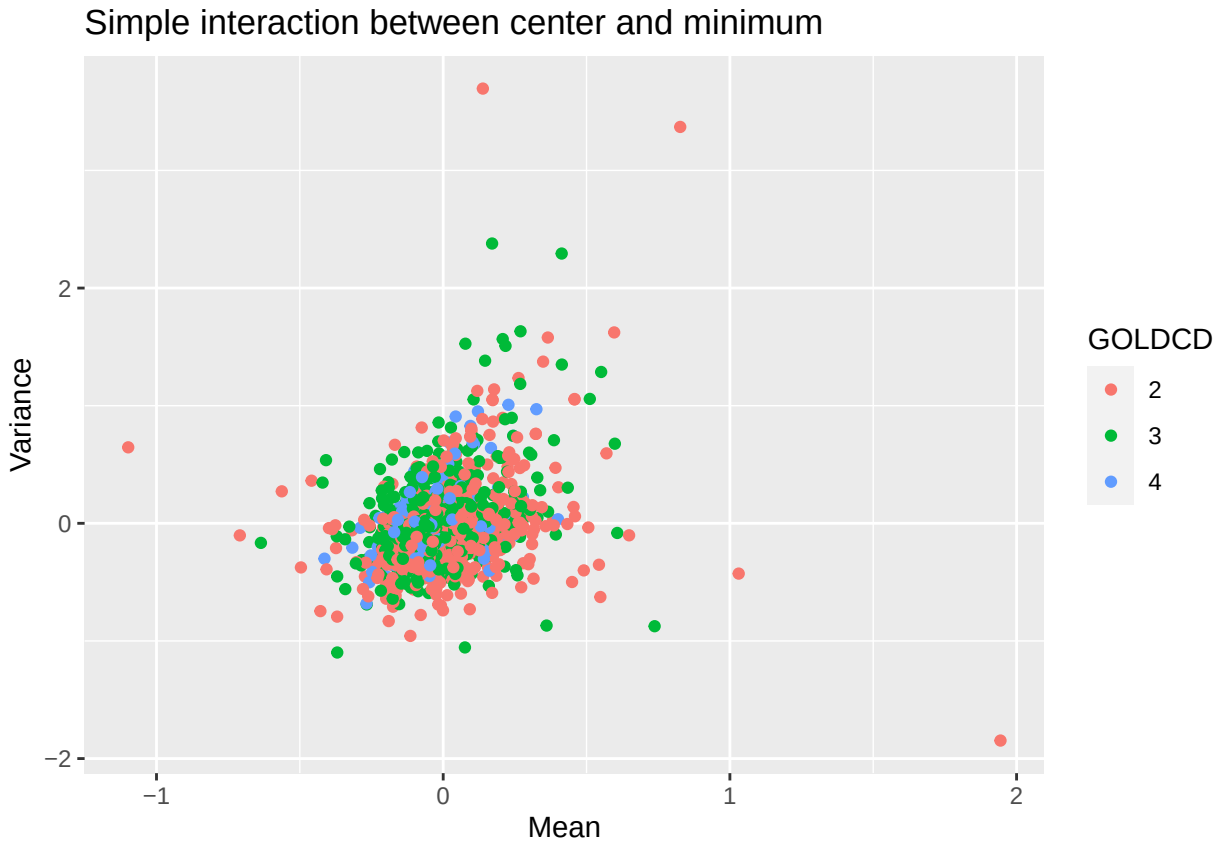
```
rmse(fitted(simple_model_mean_max), dd.variables$Mean.original)
```

```
## [1] 0.800482
```

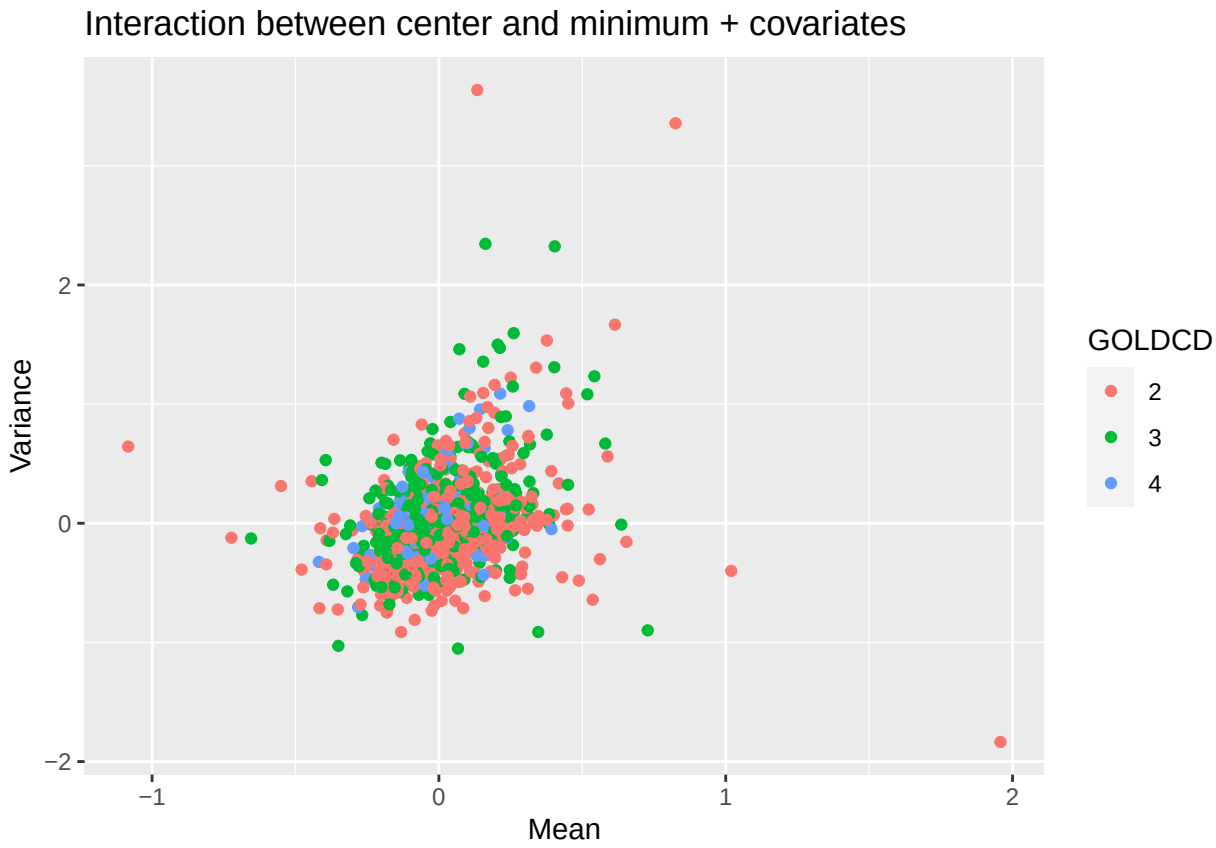
See the difference between addition and interaction of covariates:

```
# By center and minimum when interacting
model_mean_center_min_int <- lm(Mean.original ~ Minimum.original * Center, data = dd.variables)
model_variance_center_min_int <- lm(Variance.original ~ Minimum.original * Center,
    data = dd.variables)

# By center and minimum when interacting + multi
multi_model_mean_center_min_int <- lm(Mean.original ~ Age + Sex + Country + Minimum.original *
    Center, data = dd.variables)
multi_model_variance_center_min_int <- lm(Variance.original ~ Age + Sex + Country +
    Minimum.original * Center, data = dd.variables)

# Plot results
```

```
qplot(residuals(model_mean_center_min_int), residuals(model_variance_center_min_int),
    colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance", main = "Simple interaction between c
    labs(colour = "GOLDCD")
```




```
qplot(residuals(multi_model_mean_center_min_int), residuals(multi_model_variance_center_min_int),
    colour = dd.variables$GOLDCD, xlab = "Mean", ylab = "Variance", main = "Interaction between center a
    labs(colour = "GOLDCD")
```




```
# RSS
rss(fitted(model_mean_center_min_int), dd.variables$Mean.original)
```

```
## [1] 40.12806
```

```
rss(fitted(multi_model_mean_center_min_int), dd.variables$Mean.original)
```

```
## [1] 39.84061
```

```
# RMSE
rmse(fitted(model_mean_center_min_int), dd.variables$Mean.original)
```

```
## [1] 0.1508255
```

```
rmse(fitted(multi_model_mean_center_min_int), dd.variables$Mean.original)
```

```
## [1] 0.1502843
```

Conclusion

`Minimum pixel value` adjust data better than `Center` (RSS equal to 110.5897 and 141.3399 respectively), but when both are used as covariates the model fits better (RSS equal to 58.41367). When using multivariate model (adding `Age`, `Sex`, `Country`, `Cough`), the model is slightly improved (RSS equal to 57.40919). In addition, `maximum pixel value` does not fit the model well and it might be possible due to the presence of several outliers (RSS equal to 1130.321).

Nevertheless, the best improvement is when `Center` and `minimum` interact with each other (RSS equal to 40.12806). Adding more covariates to the previous interaction also improves slightly this model (RSS equal to 39.84061).

Export adjusted data using the best model

```
# Data frame with variables of interest
variables <- cbind(Center = colData(rdr_filt_original)$CENTREID, Minimum = table_scaled.T$Minimum.origi
# Empty data frame to store residuals once the model is adjusted
mat_res <- DataFrame(row.names = patients_original)

# Store residuals iterating by columns (by features)
mat_res <- apply(table_scaled.T, 2, function(x) {
    ex <- data.frame(feature = x)
    ex <- cbind(ex, variables)
    residuals(lm(feature ~ Minimum * Center, data = ex, na.action = na.exclude))
})
```

Before storing the adjusted data back to the `rdr` object, prepare the table for having the same format than the original data:

```
# Restore original col names (remember that some changes were previously made)
colnames(mat_res) <- features_original

# Transpose the dataframe to place features as rows and individuals as columns
assay_adjusted_original <- as.data.frame(t(mat_res))
```

Finally, add a new `assay` to `rdr` object and save it:

```
# This code allows adding assays to rdr object
assays(rdr_filt_original)$adjusted_min_int_center <- assay_adjusted_original
```

Save the object:

```
save(rdr_filt_original, file = "/Users/carlacasanovasuarez/Documents/Master Bioinformatics UAB/Práctica
```