

SCZ vs CTL

Alex Joyce

6/29/2021

QC Initial Steps and Groups Assignments

We will perform a couple of quality control steps to deal with negative values in the data and adjust based on signal saturation (optional). Next, we will define a new column to represent the grouping. And then, we will extract end point signal values

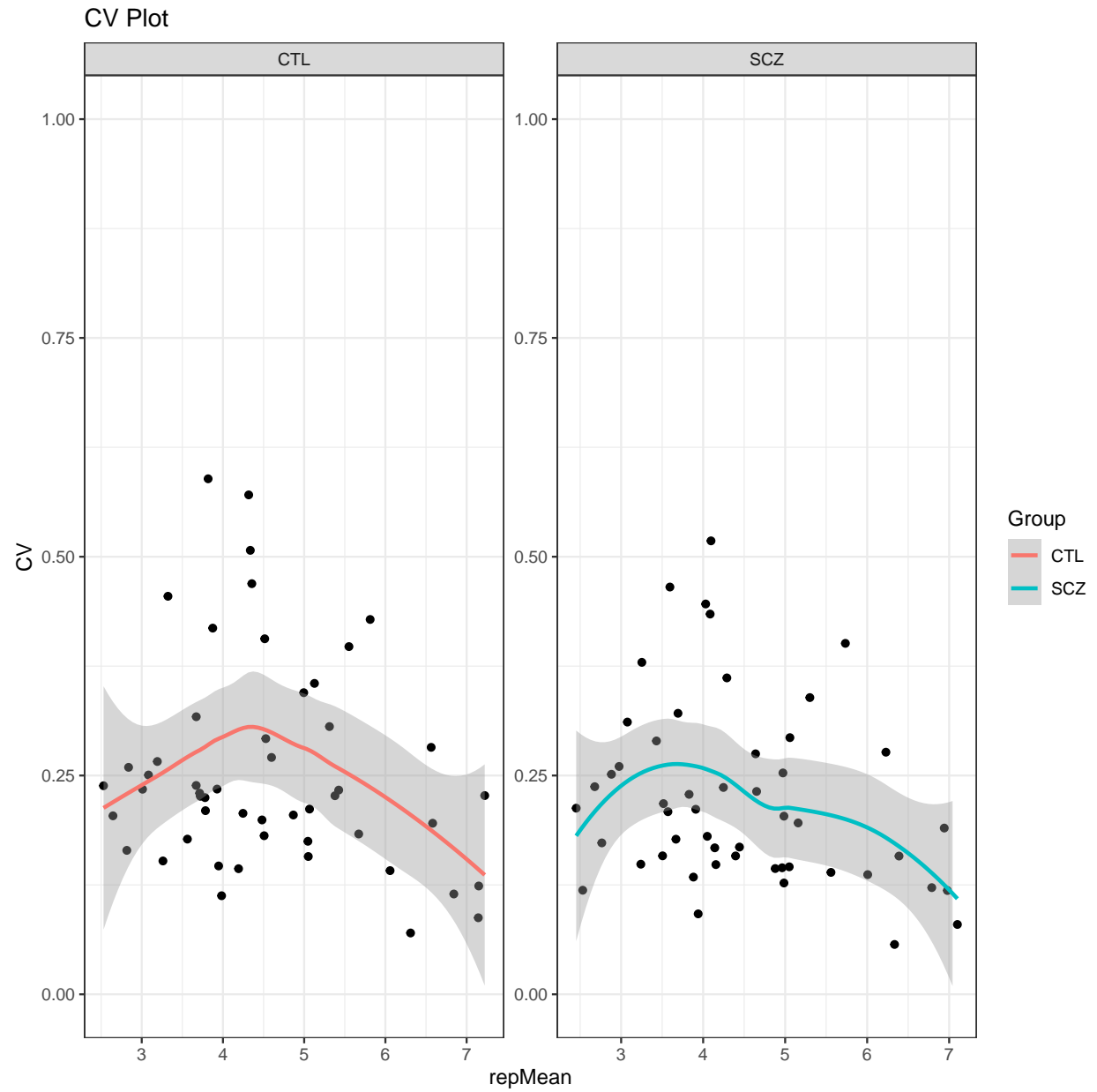
QC Steps and Model Fitting

We will filter out peptides with low signals. In order to combine the values from different exposure times into a single value, a simple linear regression model of the *Median_SigmBg* as a function of exposure time is fitted. The slope of the model fit and R^2 are then used for quality control and samples comparison. The slope is also scaled by multiplying by 100 and log2 transformed (*Slope_Transformed*). We then filter out peptides with poor linear fit and references peptides.

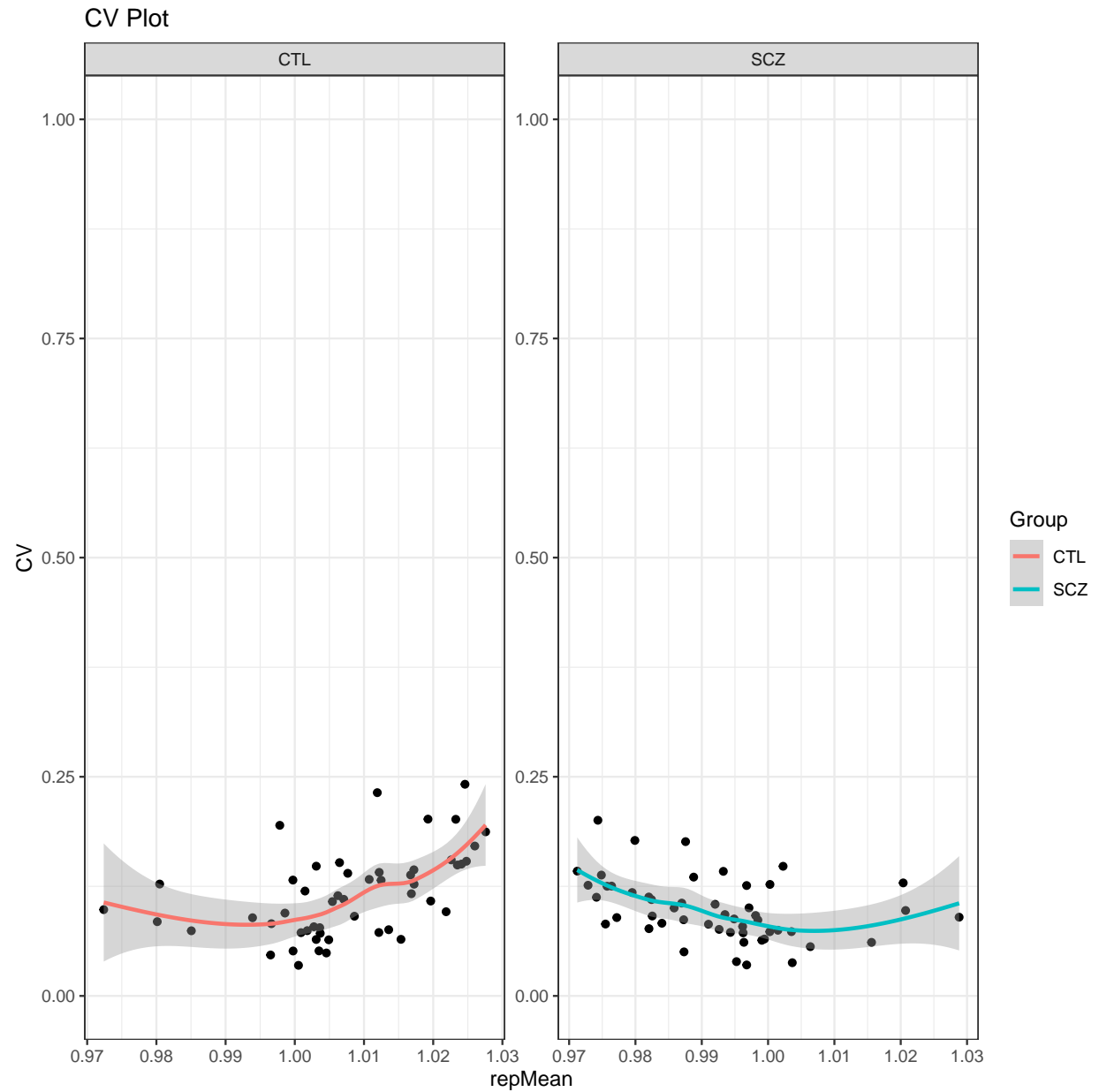
Global CV Plots

We will plot the coefficient of variation on both the normal and normalized fits. This will help us to identify groups with high variation that could be explained by sample outliers.

```
# Plot a CV figure using the modeled scaled data
krsa_cv_plot(data_modeled$scaled, new_pep)
#> `geom_smooth()` using formula 'y ~ x'
```



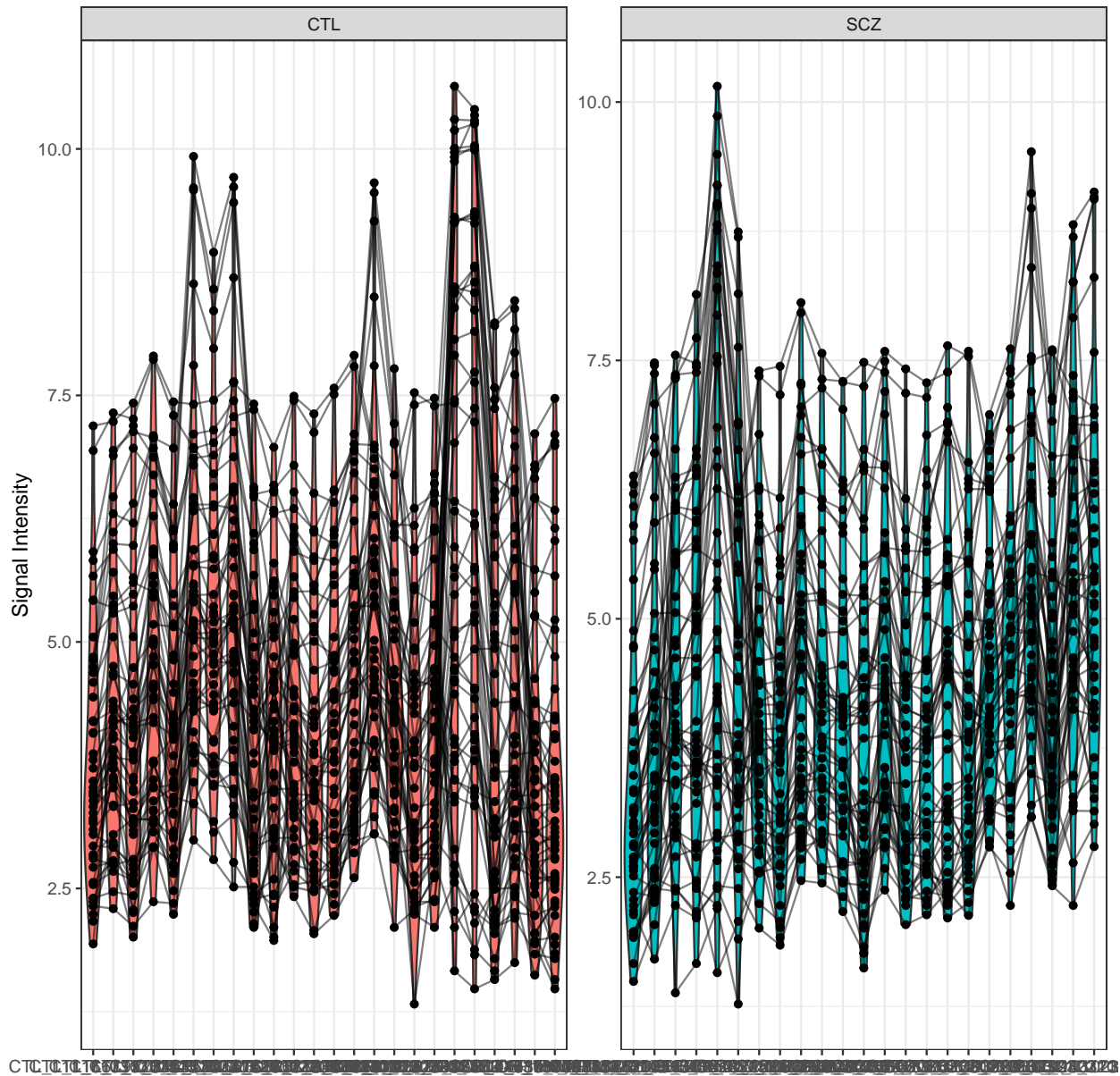
```
# Plot a CV figure using the modeled normalized data
krsa_cv_plot(data_modeled$normalized, new_pep)
#> `geom_smooth()` using formula 'y ~ x'
```



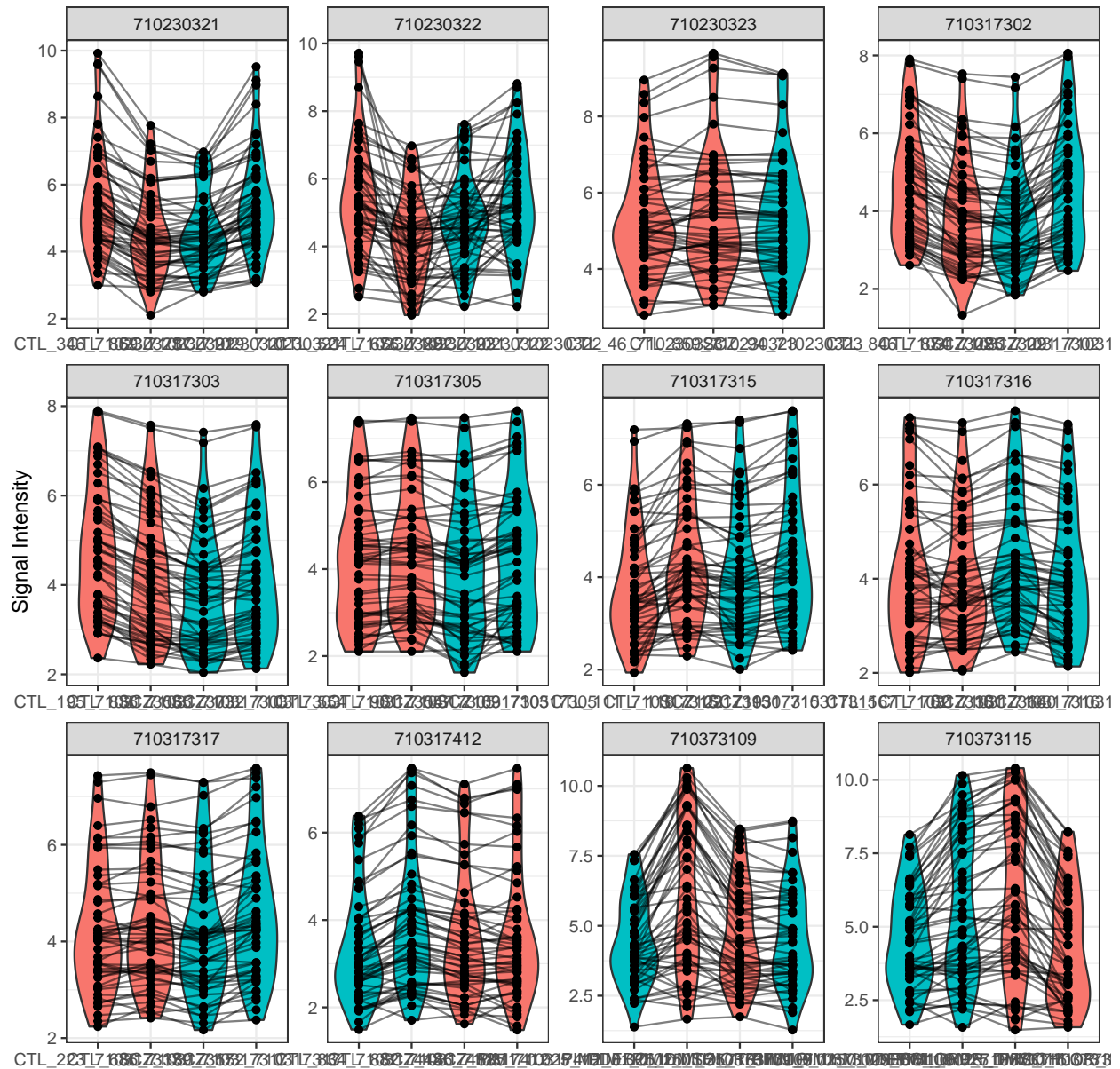
Global Violin Plots

We will plot violin figures to examine global signal differences between groups/samples.

```
# Plot a violin figure and facet by the (Group) variable
krsa_violin_plot(data_modeled$scaled, new_pep, "Group")
```



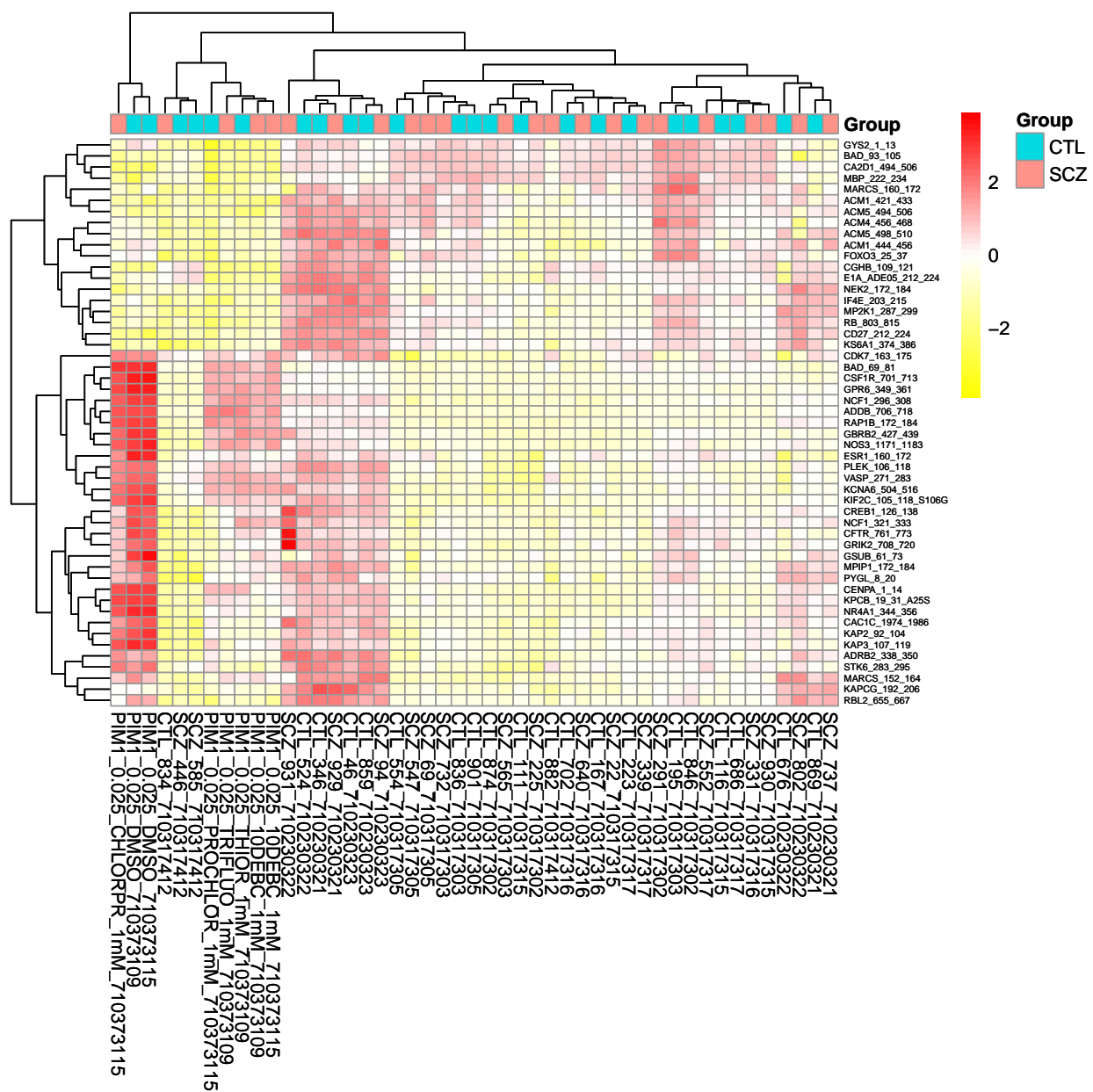
```
# Plot a violin figure and facet by the (Barcode) variable
krsa_violin_plot(data_modeled$scaled, new_pep, "Barcode")
```

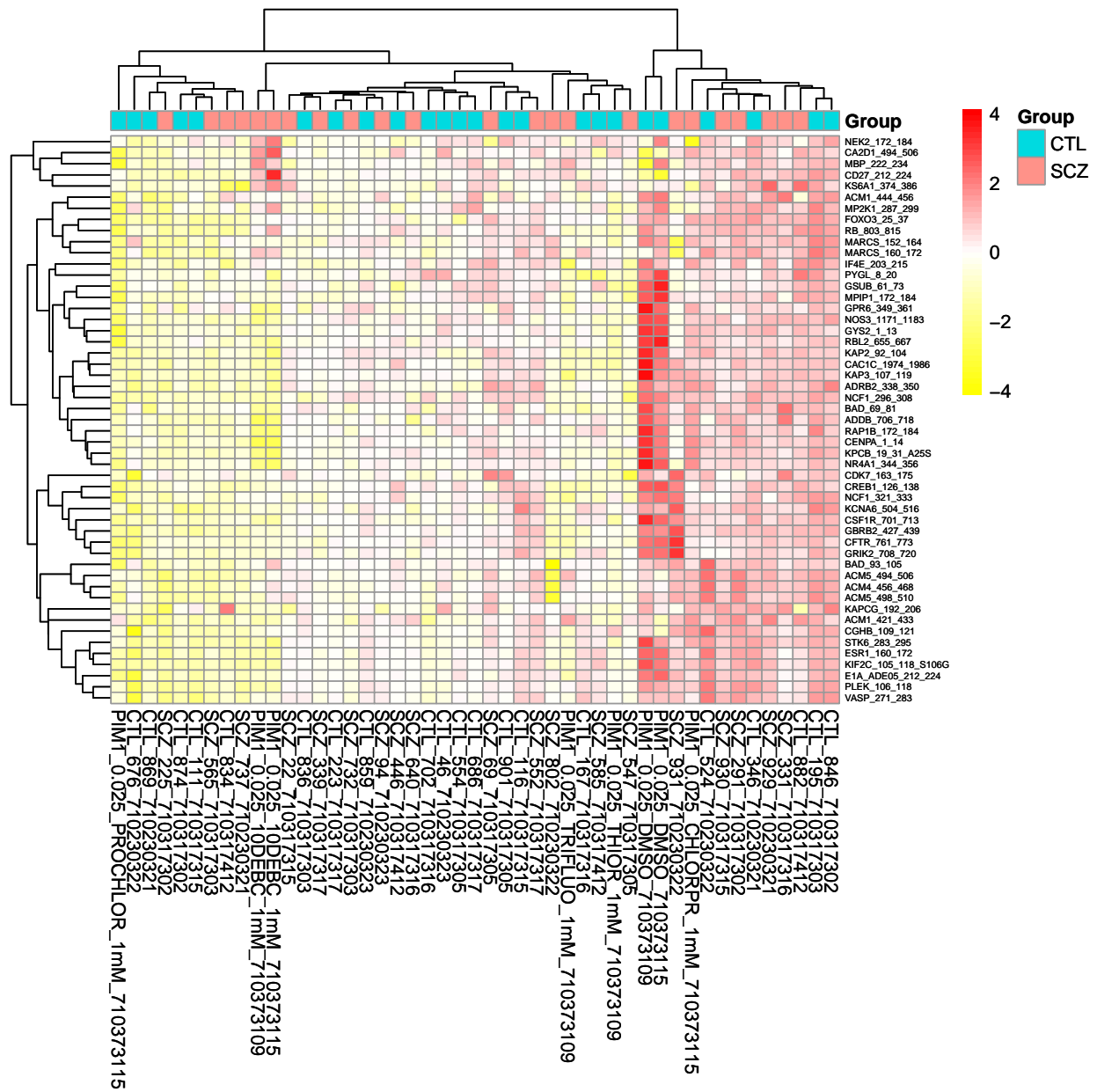


Global Heatmaps

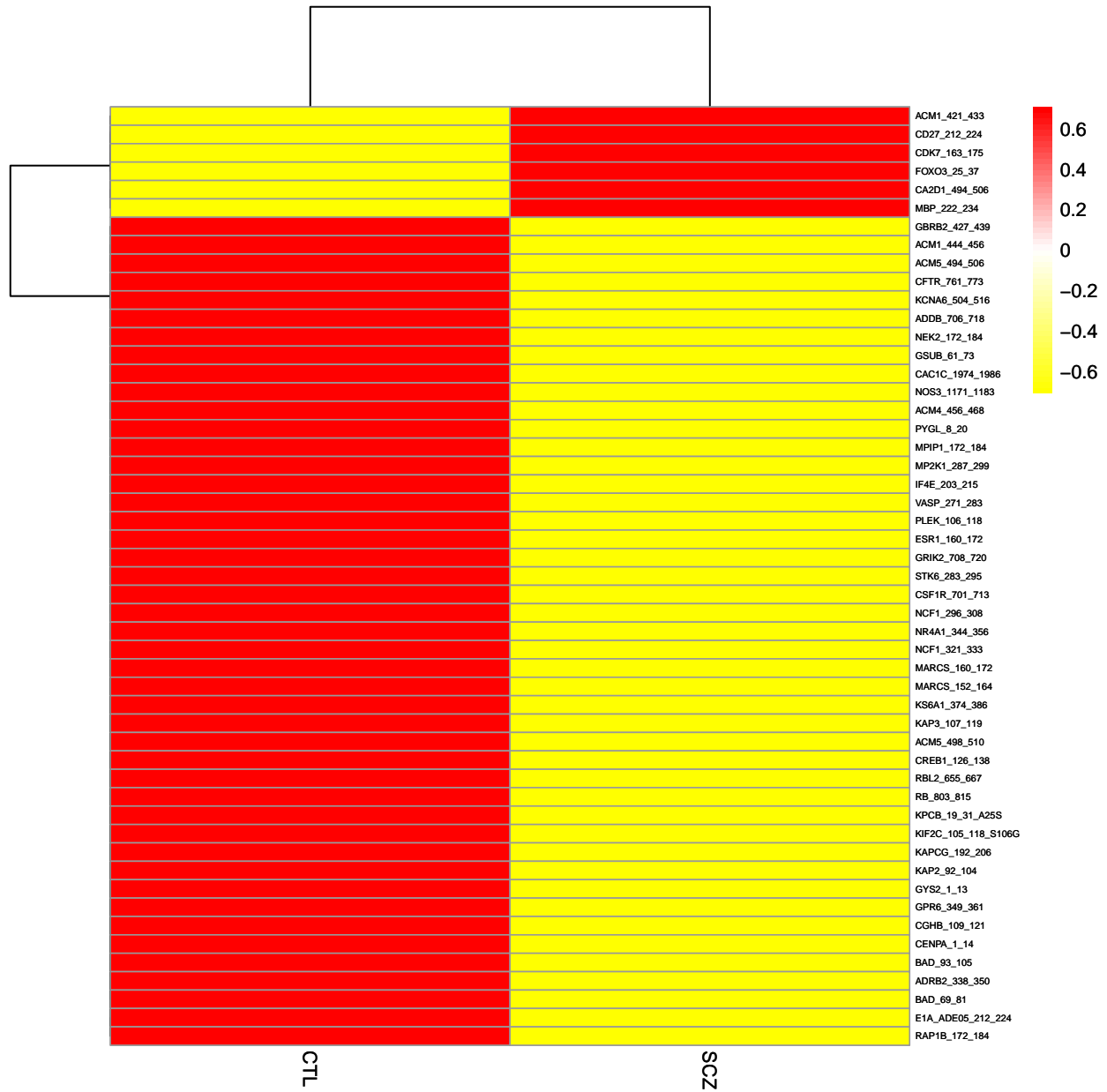
The heatmap represent all the peptides present on the chip except the positive/internal controls and peptides that failed to pass QC. The heatmaps are scaled by row to highlight the peptide signal differences across the samples. A hierarchical unsupervised clustering is applied both on the peptides and the samples to potentially group similar signatures.

```
# Generates a heatmap using the modeled scaled data
krsa_heatmap(data_modeled$scaled, new_pep, scale = "row")
```





```
# Generates a heatmap using the modeled grouped data
krsa_heatmap_grouped(data_modeled$grouped, new_pep, scale = "row")
```



Group Comparison

To compare between samples, a two-group comparison is performed. In this case, the two group comparisons are:

- **SCZ vs CTL**

The *Slope_Transformed* ratio between each group, paired by chip, is calculated to the fold change. Based on the fold change, peptides that pass a certain fold change threshold are considered significant hits. Also, quality control steps applied in each comparison to filter out peptides that do not reach specific criteria:

- The *Medain_SigmBg* at max exposure *200ms* must be above a certain value

- R^2 of the linear model fit must be above a threshold value

These *Filtering Parameters* (fold change threshold, QC criteria) can be modified to adjust the stringency of the analysis. The *Filtering Parameters* that are used for this analysis:

- The *Medain_SigmBg* at max exposure *200ms* must be equal or above 5
- R^2 of the linear model fit must be above or equal 0.8
- Log fold change (LFC) cutoffs at (0.2,0.3)

```
# Define Groups to be compared
comparisons <- list(Comp1 = c("SCZ", "CTL"))

# This function will run both QC steps (krsa_filter_lowPeps, krsa_filter_nonLinear) and krsa_filter_ref
krsa_quick_filter(data = data_pw_200, data2 = data_modeled$scaled,
  signal_threshold = 5, r2_threshold = 0.9,
  groups = comparisons$Comp1) -> pep_passed_qc
#> Filtered out 92 Peptides
#> Filtered out 6 Peptides
#> Filtered out 9 Peptides

# This function calculates log2 fold change values between the defined groups
# The byChip argument lets you calculates the log2 fold change the results within each chip
krsa_group_diff(data_modeled$scaled, comparisons$Comp1, pep_passed_qc, byChip = T) -> diff_df

# save LFC table
#write_delim(diff_df, "comp1_LFC_df.txt", delim = "\t")

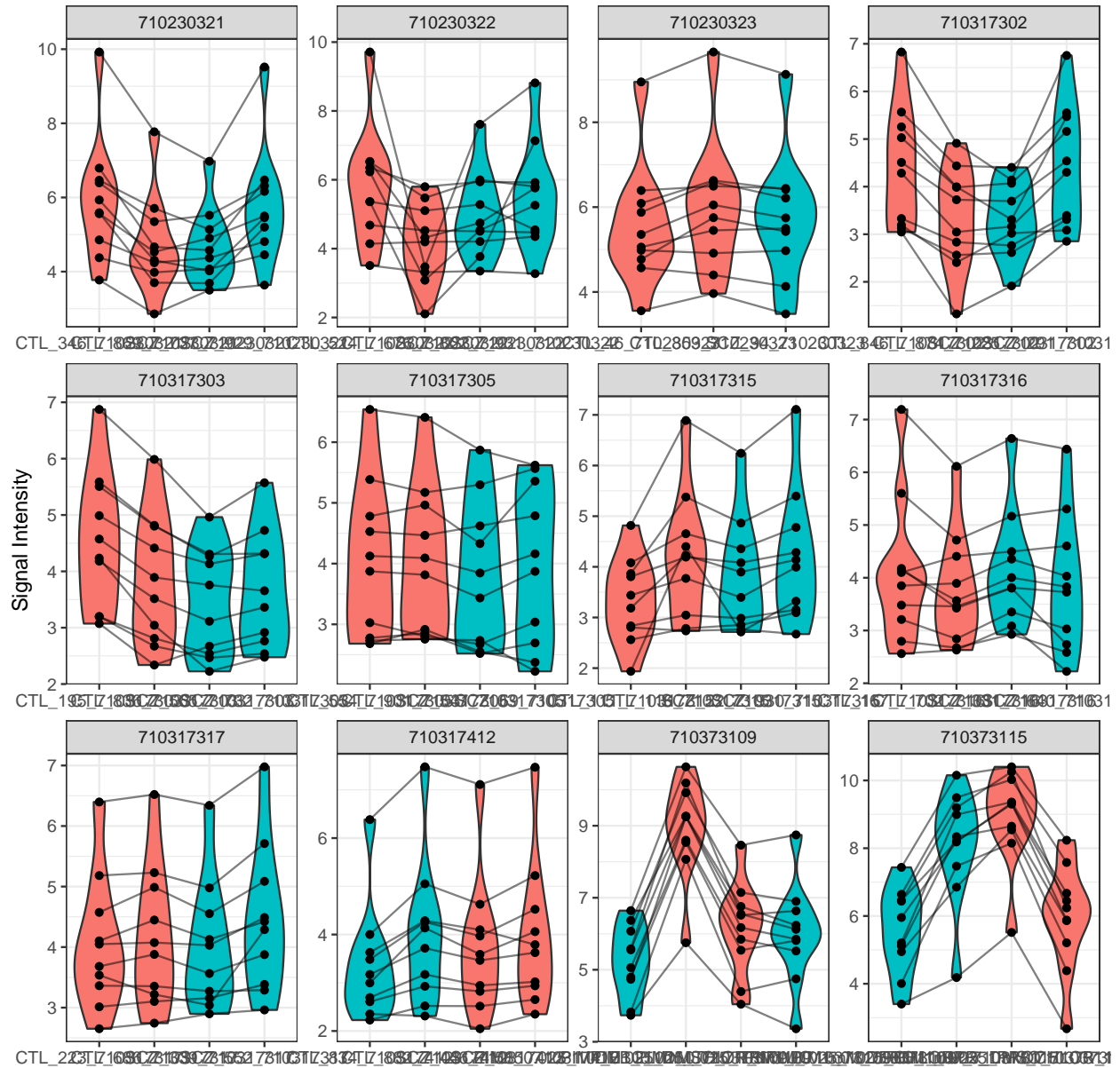
# Extract top peptides based on the LFC cutoff
krsa_get_diff(diff_df, totalMeanLFC, c(0.2, 0.3)) %>% list("meanLFC" = .) -> sigPeps
krsa_get_diff_byChip(diff_df, LFC, c(0.2, 0.3)) -> sigPepsPerChip

# Combine the peptides hits in one list
sigPeps_total <- list(sigPeps, sigPepsPerChip) %>% unlist(recursive = F) %>% unlist(recursive = F)
```


B vs A (Violin Plot)

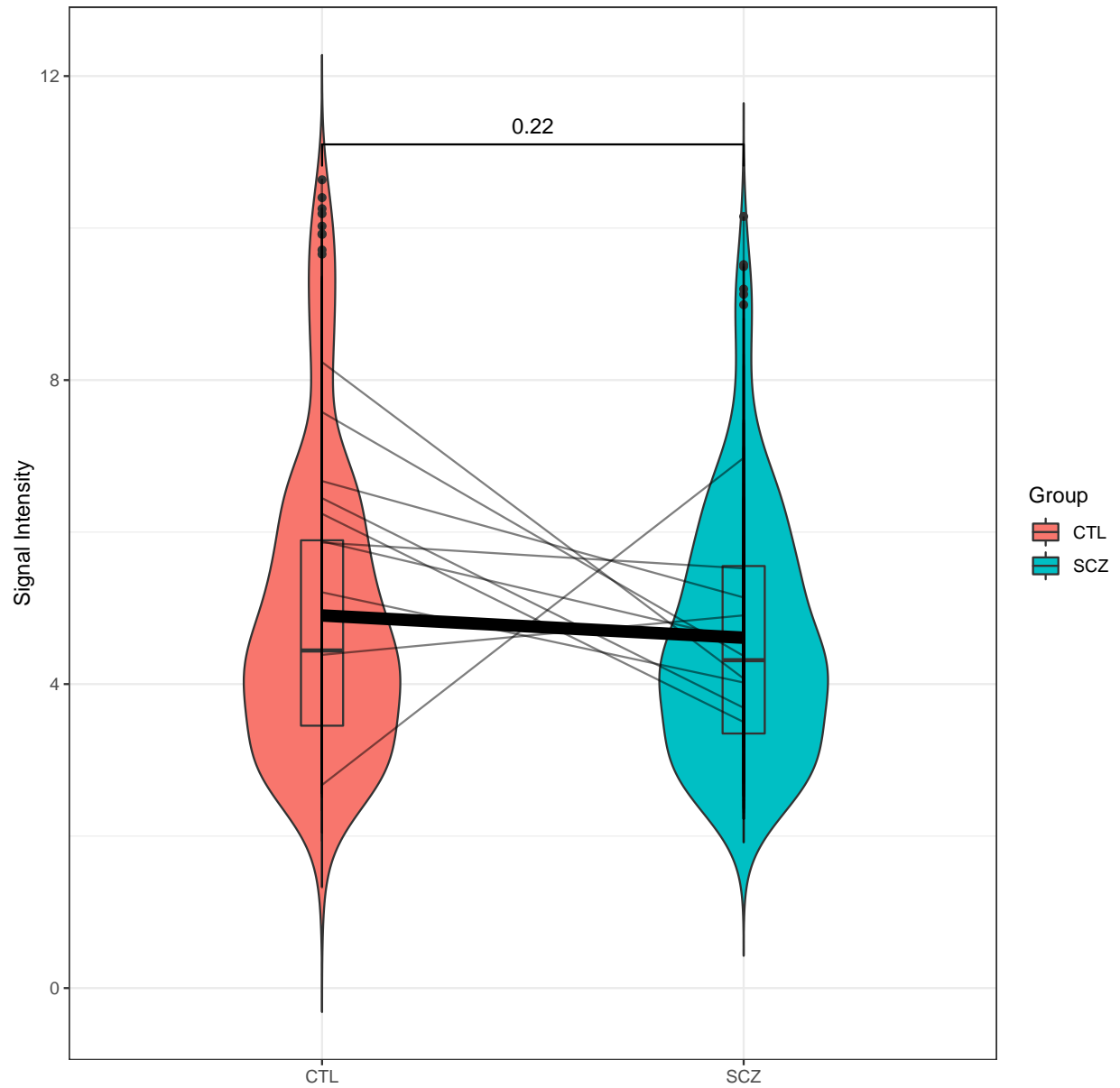
```
# generates a violin plot using the selected groups and peptides
```

```
krsa_violin_plot(data_modeled$scaled, sigPeps$meanLFC[,0.2], "Barcode", groups = comparisons$Comp1)
```

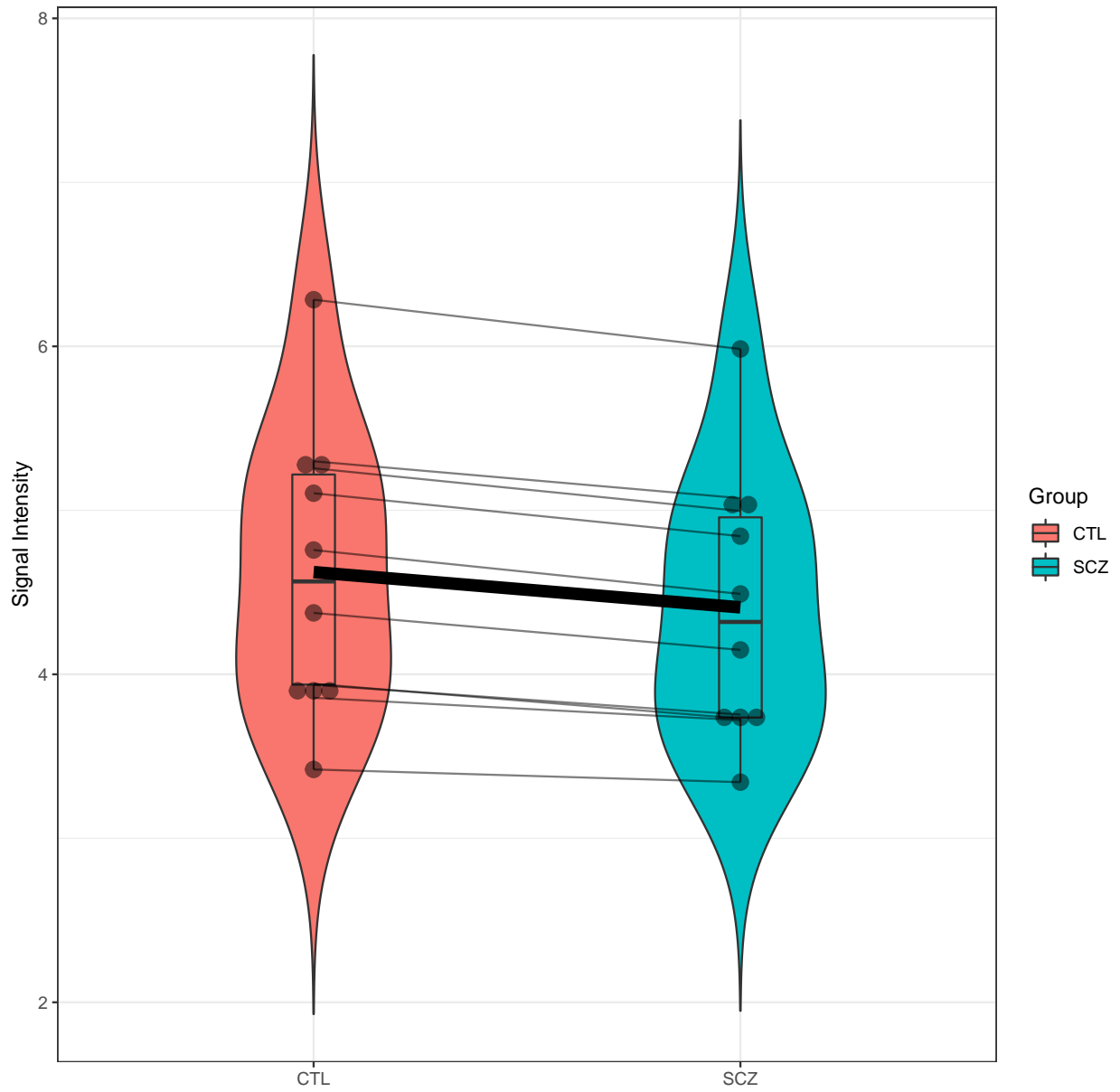


```
# generate a grouped violin/boxplot plot using the selected groups and peptides with more options, like
```

```
krsa_violin_plot_grouped(data_modeled$scaled, sigPeps$meanLFC`0.2`, comparisons,
  dots = F,
  groups = comparisons$Comp1, avg_line = T)
```



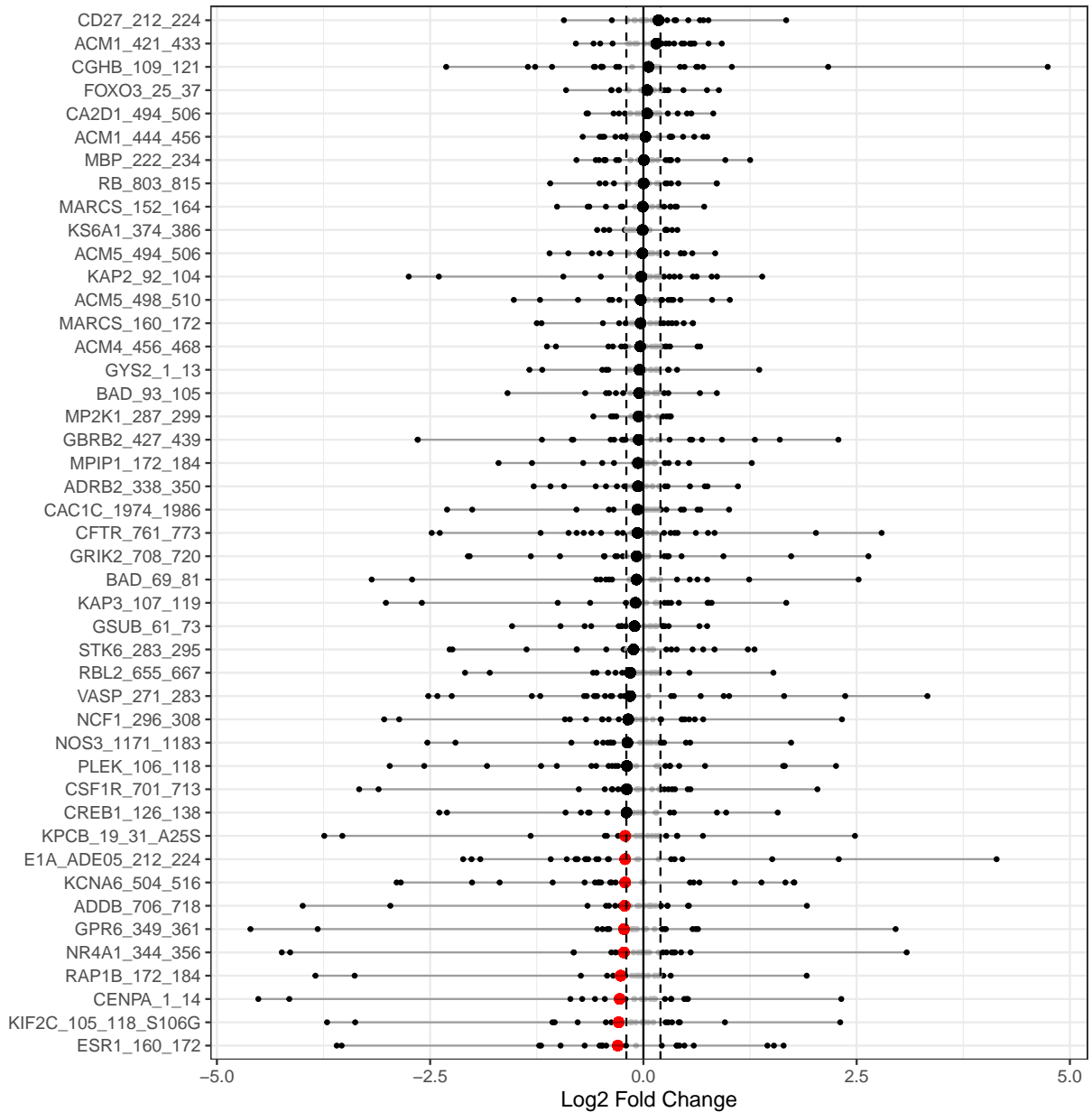
```
krsa_violin_plot_grouped(data_modeled$grouped, sigPeps$meanLFC$`0.2`, comparisons,
  test = F,
  groups = comparisons$Comp1, avg_line = T)
```



B vs A (Waterfall Plot)

This waterfall represents the log2 fold changes between the two groups at each peptide.

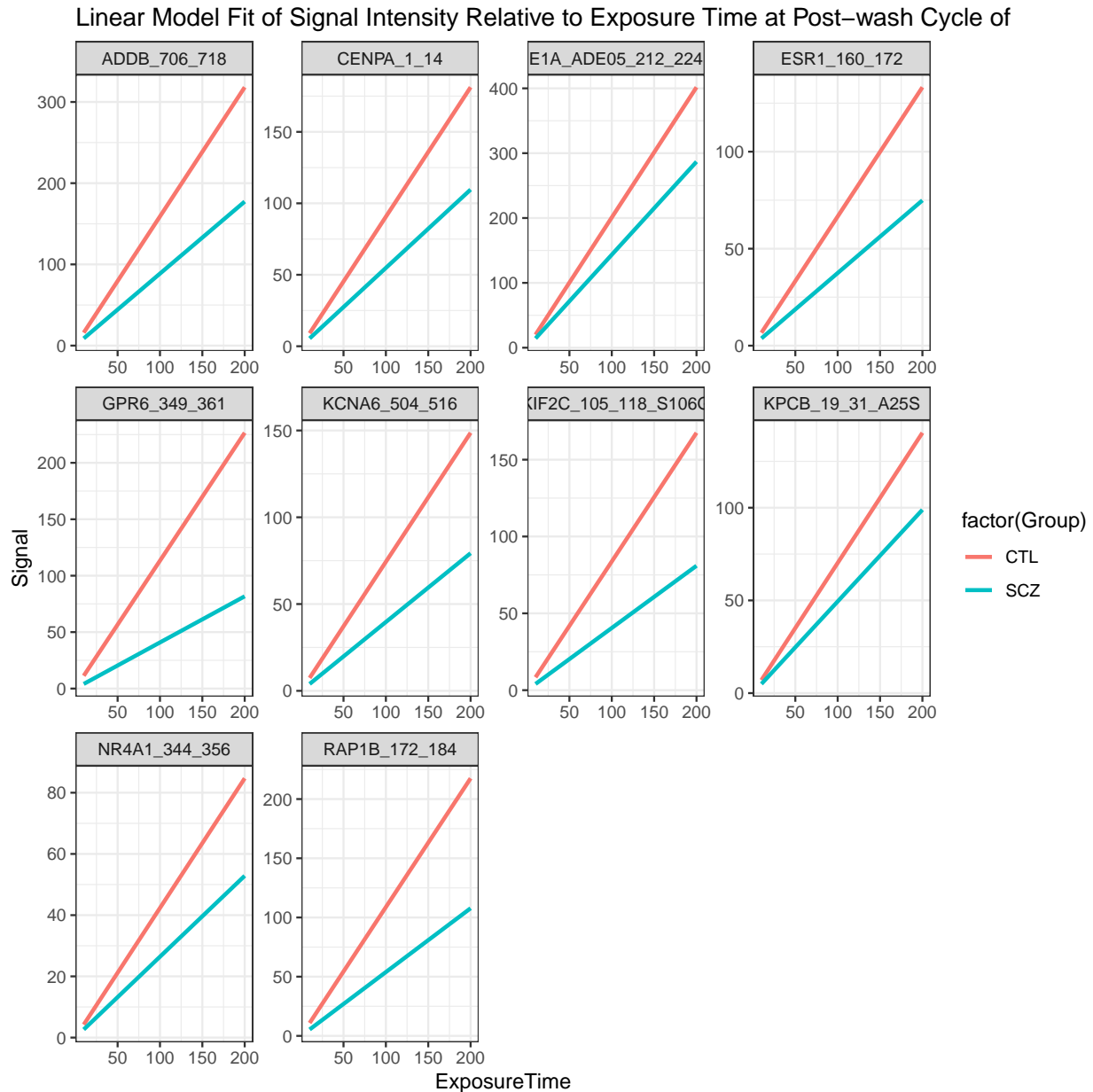
```
# generates a waterfall of the log2 fold change values for the selected peptide (top peptides)
krsa_waterfall(diff_df, 0.2, byChip = T)
```



B vs A (Curve Plot)

We generate curves plots per peptide using the last cycle data to visualize the linear fit model

```
# generates a curve plot representing the linear model fit for the selected peptide (top peptides, only)
krasa_curve_plot(data_pw, sigPeps$meanLFC$`0.2`[1:10], groups = comparisons$Comp1)
```



B vs A (Upstream Kinase Analysis)

The lab carefully curated and mapped the kinases that can act and phosphorylate each peptide present on the chip. This was achieved by using multiple sources including GPS 3.0, Kinexus Phosphonet, PhosphoELM and PhosphoSite Plus. Based on that association between peptides and kinases, a random sampling analysis is performed for these hits. The basic idea of *KRSA* is: For each iteration (2000 iterations performed in

this analysis), the same number of hits are randomly selected from the total 141/or 193 peptides present on the chip. Predicted kinases are then mapped to this sample list of peptides and number of kinases are determined. The kinase count from the actual hits and random sampling is then compared to determine the significance.

```
# load in chip coverage and kinase-substrate files OR upload your own files
# if PTK chip, use:
# chipCov <- KRSA_coverage_PTK_PamChip_86402_v1
# KRSA_file <- KRSA_Mapping_PTK_PamChip_86402_v1

# STK chip
chipCov <- KRSA_coverage_STK_PamChip_87102_v1
KRSA_file <- KRSA_Mapping_STK_PamChip_87102_v1

# run the KRSA function to do the random sampling analysis, set seed that can be used later to reproduc
krsa(sigPeps$meanLFC$`0.2`, return_count = T, seed = 123, itr = 2000,
     map_file = KRSA_file, cov_file = chipCov) -> fin

# View the Z score table
kable(head(fin$KRSA_Table,25), digits = 3)
```

Kinase	Observed	SamplingAvg	SD	Z
DMPK	10	7.289	1.398	1.940
MTOR	0	1.776	1.142	-1.555
AUR	2	0.779	0.801	1.523
PKA	2	4.244	1.506	-1.489
CAMK1	0	1.503	1.088	-1.382
PKCA	10	8.584	1.037	1.366
CK1	1	2.890	1.424	-1.327
MLK	0	1.222	1.009	-1.211
RIPK	1	2.667	1.383	-1.205
PKCH	0	1.120	0.950	-1.179
RAF	3	1.704	1.154	1.123
STE7	2	3.651	1.477	-1.118
IRAK	0	1.028	0.952	-1.080
ATR	0	0.941	0.885	-1.063
PKCD	0	0.886	0.859	-1.031
MAPKAPK	5	3.482	1.482	1.024
GRK	0	0.887	0.877	-1.012
PRKDC	0	0.837	0.843	-0.993
PKCI	0	0.791	0.804	-0.984
LKB	0	0.753	0.804	-0.937
PLK	2	3.257	1.454	-0.865
AKT	4	2.844	1.360	0.850
PHK	8	6.894	1.364	0.811
STKR	0	0.499	0.652	-0.765
ERK	3	4.114	1.507	-0.739

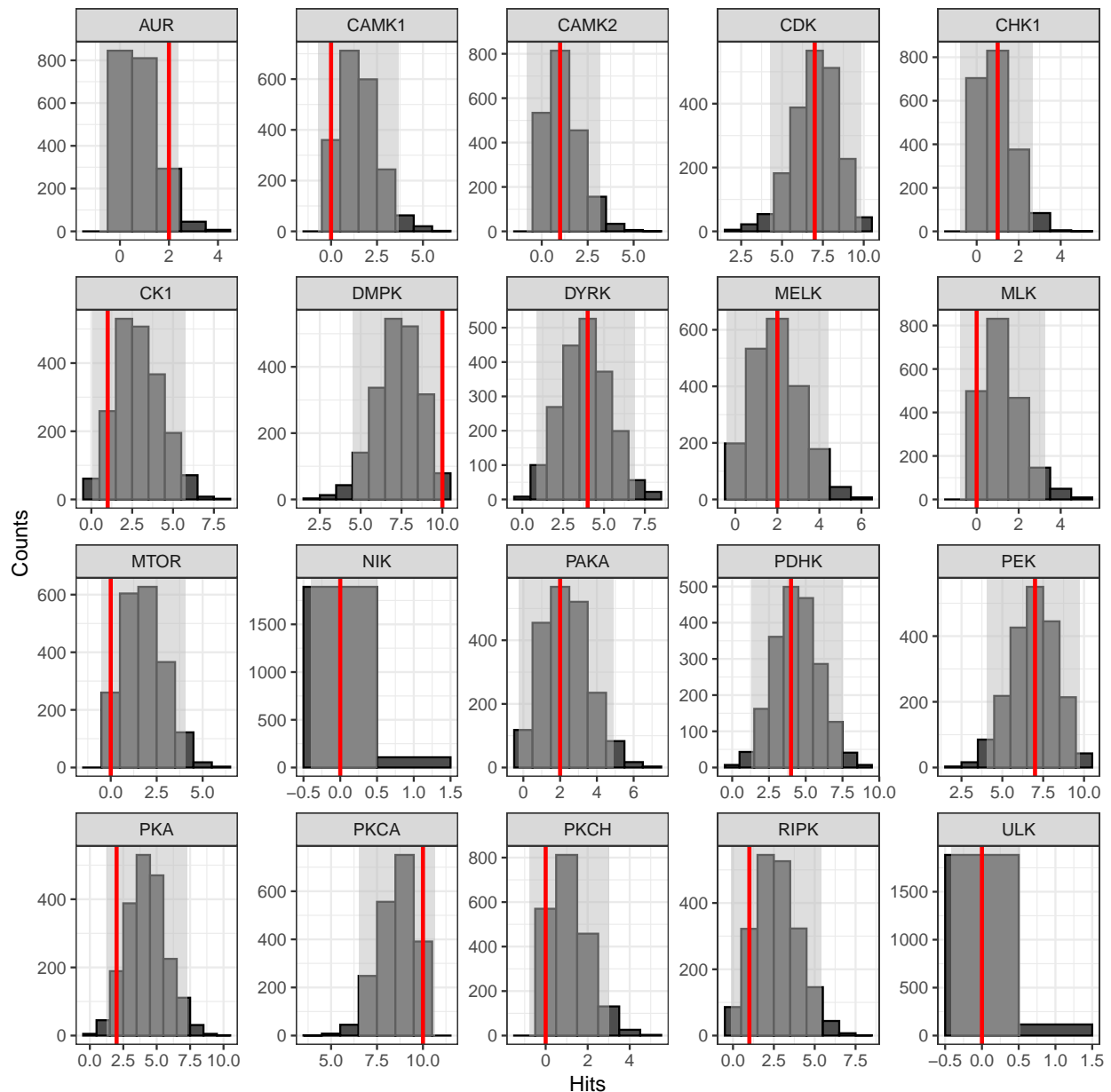

```

# to save file
#fin$KRSA_Table %>% write_delim("acrossChip_KRSA_FullTable_comp1.txt", delim = "\t")

# find top and bottom kinases
bothways <- c(pull(head(fin$KRSA_Table, 10), Kinase), pull(tail(fin$KRSA_Table, 10), Kinase))

# Use these kinase to generate histogram plots for each selected kinase
krsa_histogram_plot(fin$KRSA_Table, fin$count_mtx, bothways)

```



```

# For parallel computing, load the furrr package:
# opens multiple R sessions to run faster

```

```

plan(multisession)

# Run the KRSA function across the different sets of peptides using the furrr package for parallel computing
future_map(sigPeps_total, krsa) -> mutiple_krsa_outputs

# For none parallel computing:
# Run KRSA function across the different sets of peptides
#map(sigPeps_total, krsa) -> mutiple_krsa_outputs

# Tidy output
df <- data.frame(matrix(unlist(mutiple_krsa_outputs), ncol = max(lengths(mutiple_krsa_outputs)), byrow = TRUE))
df <- setNames(do.call(rbind.data.frame, mutiple_krsa_outputs), names(mutiple_krsa_outputs$meanLFC.0.2))

df %>% rownames_to_column("method") %>% select(Kinase, Z, method) %>%
  mutate(method = str_extract(method, "\\w+\\.\\.\\w+\\.\\.\\w+")) %>%
  mutate(method = gsub("(^\\.\\.\\w+)[\\.\\.]", "\\1>", method)) %>%
  mutate_if(is.numeric, round, 2) -> df

df %>%
  pivot_wider(names_from = method, values_from = Z) -> df2

# Generates a table of the Z scores across the different sets of peptides
# df2 %>%
#   gt() %>% tab_spanner_delim(delim = ">")

# Creates an average Z score table using the across chip analysis
df %>%
  filter(grepl("mean", method)) %>%
  select(Kinase, Z, method) %>% group_by(Kinase) %>% mutate(AvgZ = mean(Z)) -> AvgZTable

# save file
#AvgZTable %>% write_delim("acrossChip_KRSA_Table_comp1.txt", delim = "\t")

# Creates an average Z score table using the within chip analysis
df %>%
  filter(!grepl("mean", method)) %>%
  select(Kinase, Z, method) %>% group_by(Kinase) %>% mutate(AvgZ = mean(Z)) -> AvgZTable2

# save file
#AvgZTable %>% write_delim("withinChip_KRSA_Table_comp1.txt", delim = "\t")

# Extract top kinases based on abs(Z) score
krsa_top_hits(AvgZTable2, 2) -> kinases_hits

#krsa_top_hits(AvgZTable2, 1.75)
#krsa_top_hits(AvgZTable2, 1.5)

# Show the number of peptides per each set in a table
krsa_show_peptides(sigPeps_total)
#> # A tibble: 26 x 2
#>   Method      NumberOfPeptides
#>   <chr>          <int>

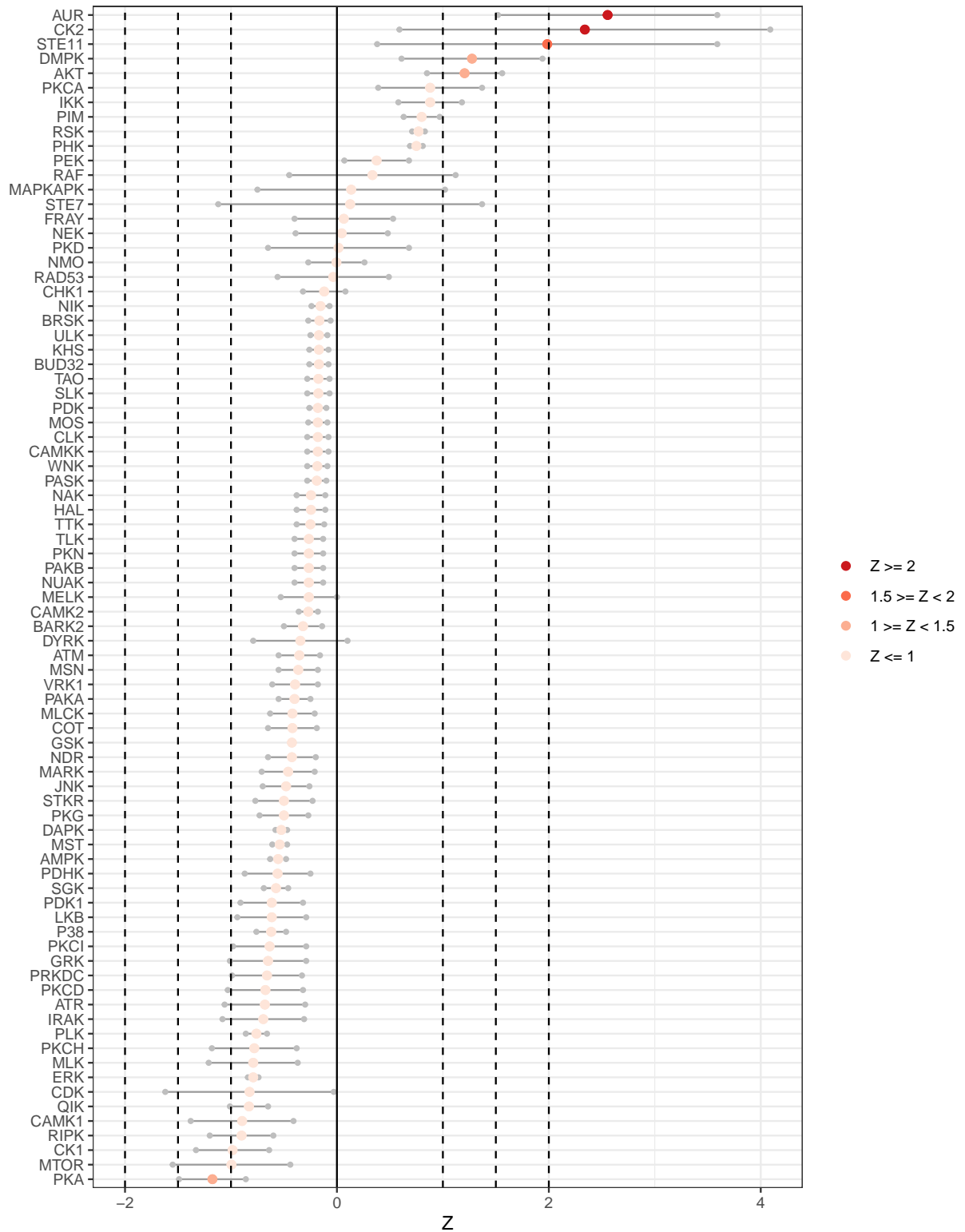
```

```
#> 1 meanLFC.0.2 10
#> 2 meanLFC.0.3 1
#> 3 710230321.0.2 27
#> 4 710230321.0.3 18
#> 5 710230322.0.2 44
#> 6 710230322.0.3 38
#> 7 710230323.0.2 31
#> 8 710230323.0.3 24
#> 9 710317302.0.2 26
#> 10 710317302.0.3 18
#> # ... with 16 more rows
```

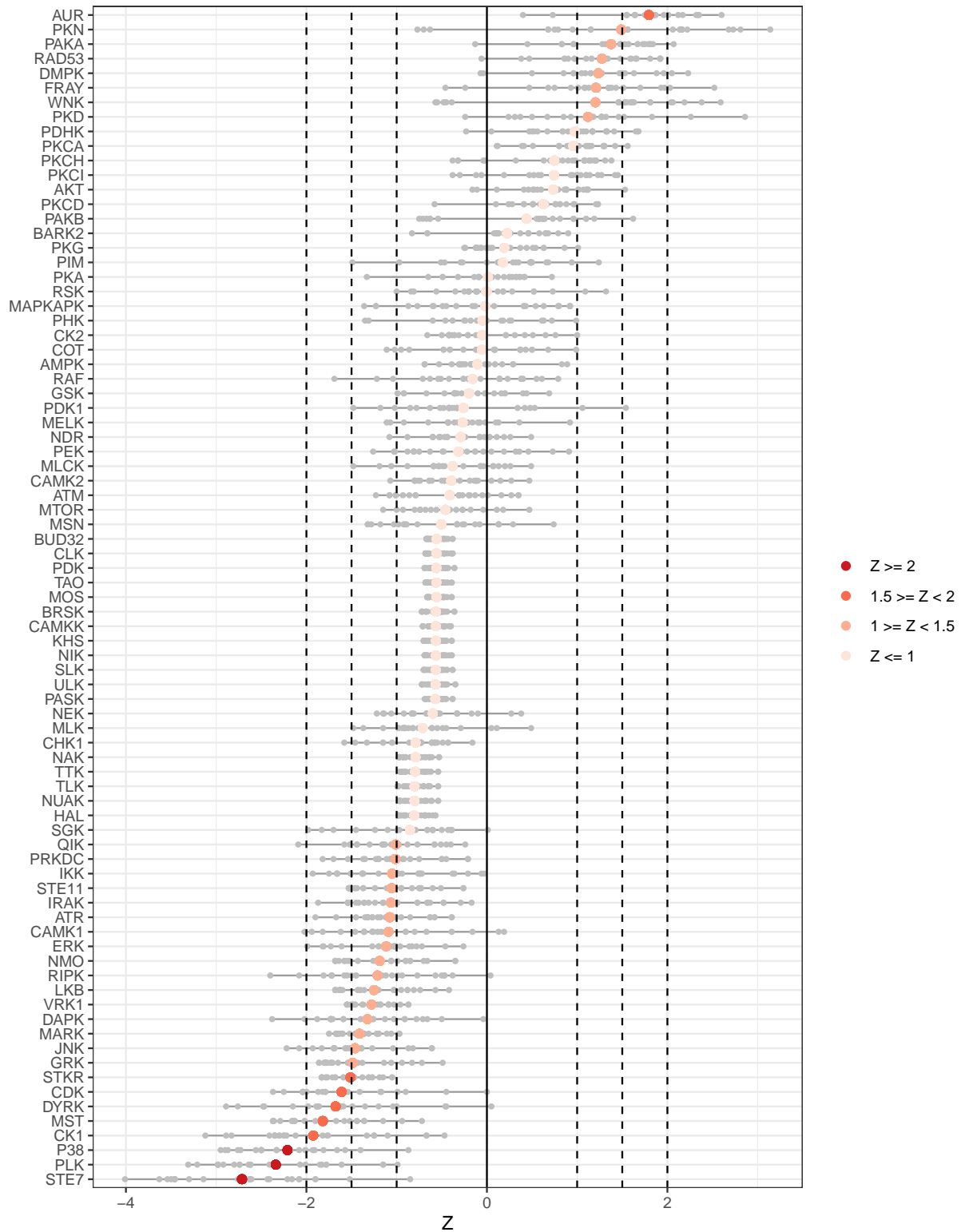
B vs A (Z Scores Plot)

We will plot the individual and averaged Z scores using both the across and within chip analyses.

```
# Generates Z scores waterfall plots  
krsa_zscores_plot(AvgZTable)
```



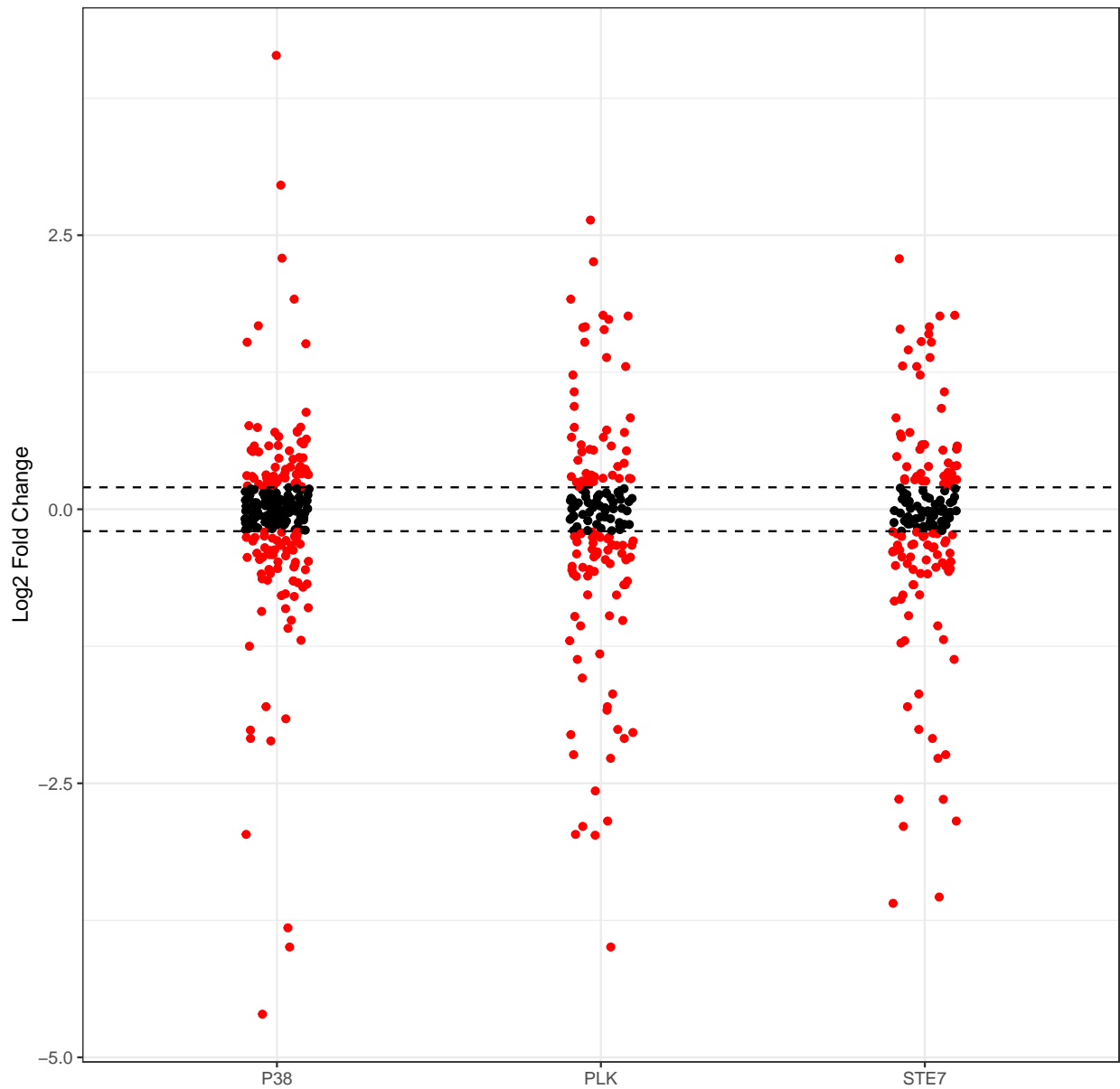
```
krsa_zscores_plot(AvgZTable2)
```



B vs A (Reverse KRSA Plot)

We will use the reverse KRSA plot function, to plot the log2 fold change values for all peptides mapped to kinase hits. This will help us examine the activity of the kinase

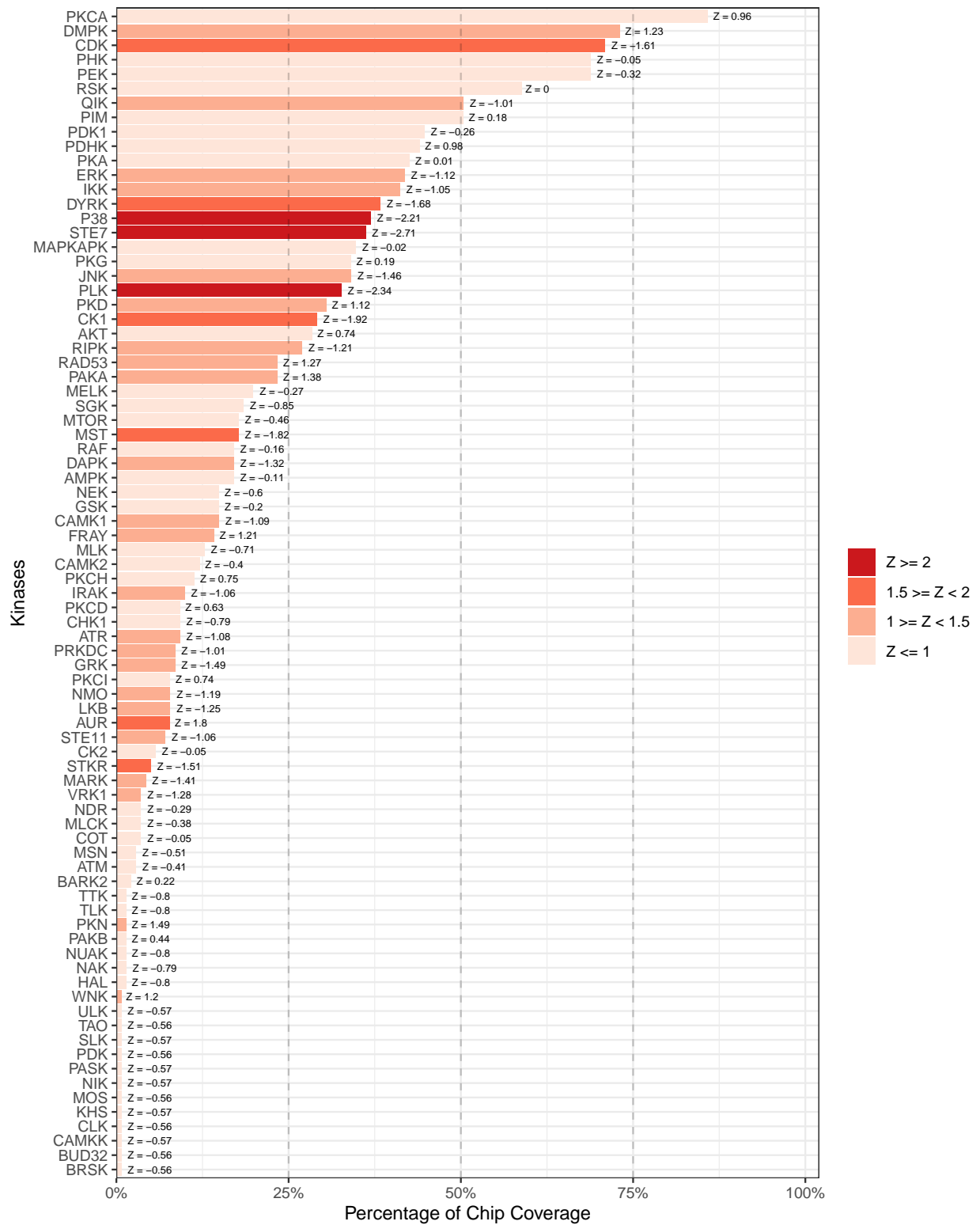
```
# plot the reverse KRSA figure for top kinases to determine their activity levels  
krsa_reverse_krsa_plot(chipCov, diff_df, kinases_hits, 0.2, byChip = F)
```



B vs A (Coverage Plot)

To view the coverage of kinases across the full list of peptides on the chip, we will use the coverage plot function

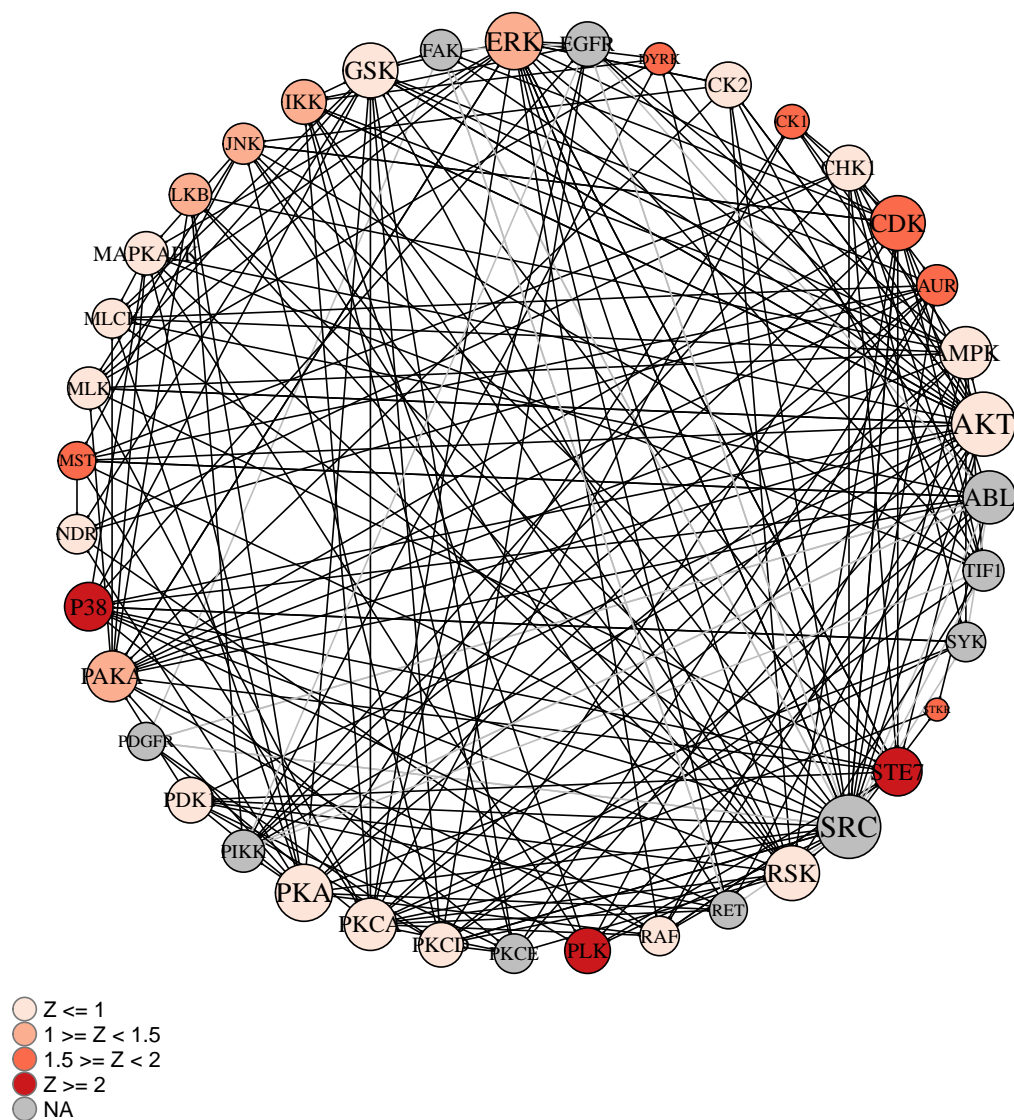
```
# generates a kinase coverage plot  
krsa_coverage_plot(chipCov, AvgZTable2, chipType)
```

B vs A (Ball Model Network)

We will view the ball model network function, to generate a model representing the protein-protein interactions between kinases

```
# Plot the network ball model  
krsa_ball_model(kinases_hits, AvgZTable2, 10, 2.5, 4.8)
```



```
installed.packages()[names(sessionInfo())$otherPkgs, "Version"]
#>    frrrr    future      gt  forcats  stringr  dplyr    purrr    readr
```

```
#>  "0.2.2"  "1.21.0"  "0.3.0"   "0.5.1"   "1.4.0"   "1.0.6"   "0.3.4"   "1.4.0"
#>    tidyr    tibble  ggplot2 tidyverse    knitr      KRSA
#>  "1.1.3"  "3.1.2"  "3.3.3"   "1.3.1"   "1.33"    "0.9.4"
```

Session Info