

R plotting

Zhuyu Qiu

11/12/2019

Question 2

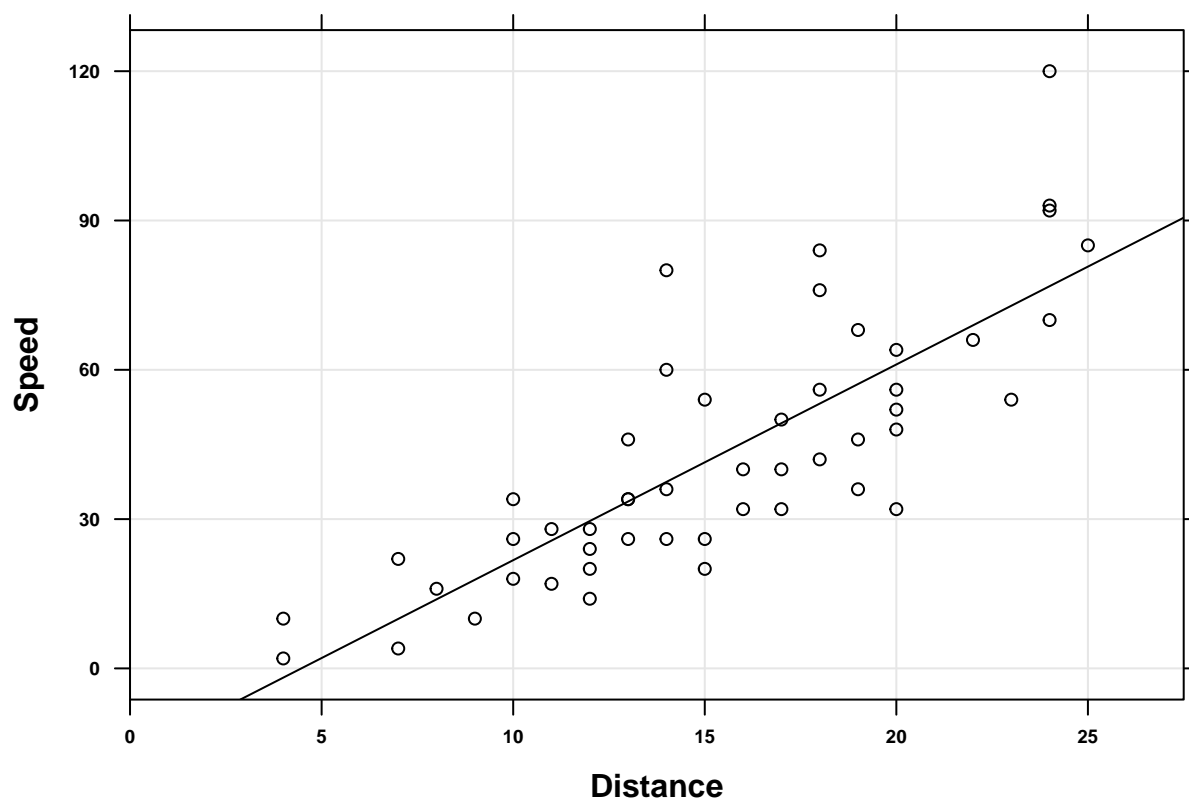
2a. Create a simple scatterplot using `BoutrosLab.plotting.general`

```
cars = datasets::cars;
summary(cars);
```

```
##      speed      dist
##  Min.   : 4.0    Min.   :  2.00
## 1st Qu.:12.0    1st Qu.: 26.00
##  Median:15.0    Median : 36.00
##   Mean  :15.4    Mean   : 42.98
## 3rd Qu.:19.0    3rd Qu.: 56.00
##   Max.  :25.0    Max.    :120.00
```

```
create.scatterplot(
  formula = dist ~ speed,
  data = cars,
  main = "Scatter plot of distance and speed",
  xlab.label = "Distance",
  ylab.label = "Speed",
  yat = seq(0,150,30),
  xaxis.cex = 0.6,
  yaxis.cex = 0.6,
  xlab.cex = 1,
  ylab.cex = 1,
  main.cex = 1,
  pch = 1,
  col = "black",
  type = c ( "p" , "g" , "r" )
)
```

Scatter plot of distance and speed



2b. Create a heatmap displaying data found in the 'Loblolly' dataset

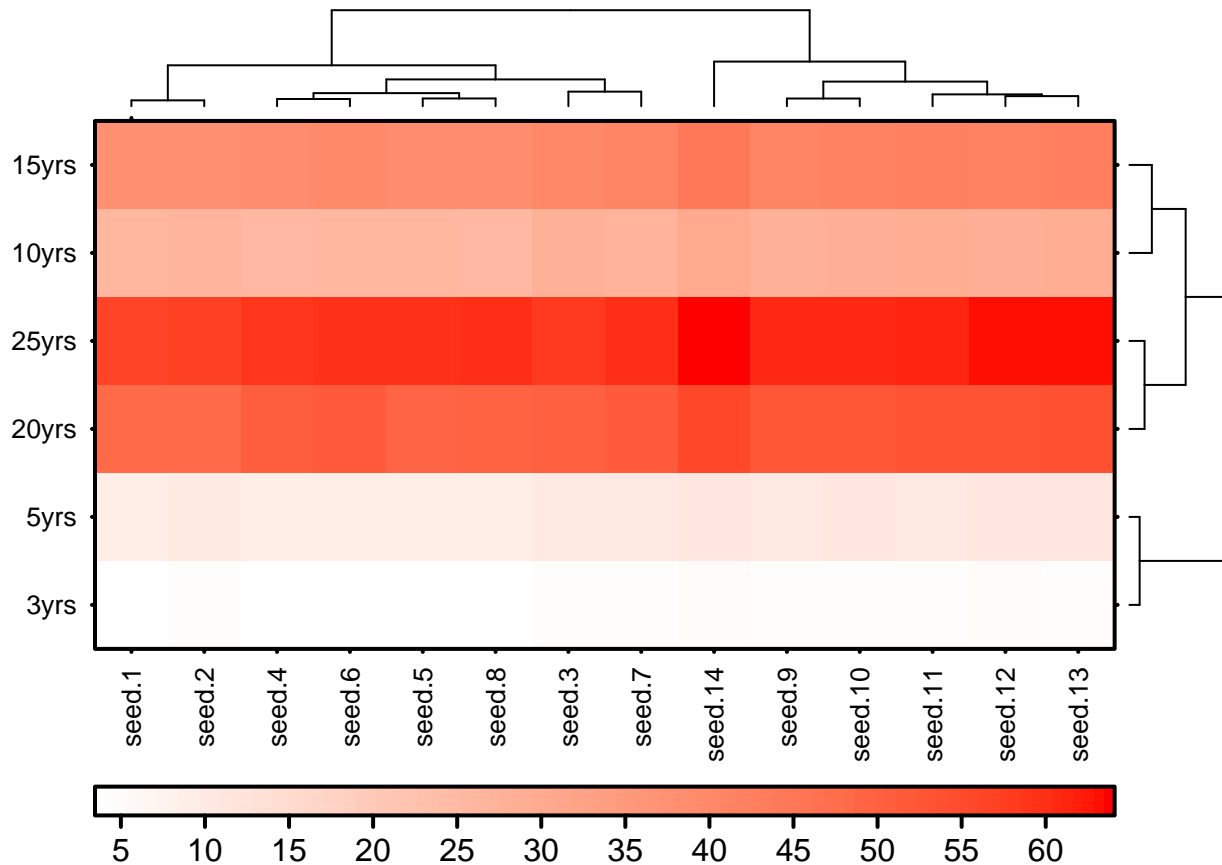
```
loblolly = datasets::Loblolly;
loblolly = reshape(data = loblolly, idvar = "Seed",
                   v.names = "height",
                   timevar = "age",
                   direction = "wide");
loblolly$Seed = as.numeric(loblolly$Seed);
loblolly.sort = loblolly[order(loblolly$Seed),];
loblolly.matrix = as.matrix(loblolly.sort[,-1]);
row.names(loblolly.matrix) = paste("seed", 1:14, sep = ".");
colnames(loblolly.matrix) = c("3yrs", "5yrs", "10yrs", "15yrs", "20yrs", "25yrs");
create.heatmap(
  x = loblolly.matrix,
  # format the colour key
  colourkey.cex = 1,
  colourkey.labels.at = seq(0, 70, 5),
  # set labels to NA -- results in default labels
  xaxis.lab = NA,
  yaxis.lab = NA,
  xaxis.cex = 0.8,
  yaxis.cex = 0.8,

  # set font style (default is bold, 1 is roman)
  xaxis.fontface = 1,
  yaxis.fontface = 1,
```

```

# specify clustering method
# if no clustering is desired, set this to "none"
clustering.method = "complete",
# select distance measure
rows.distance.method = "euclidean",
cols.distance.method = "manhattan"
);

```



2c. Take a look at the ‘ChickWeight’ dataset

```

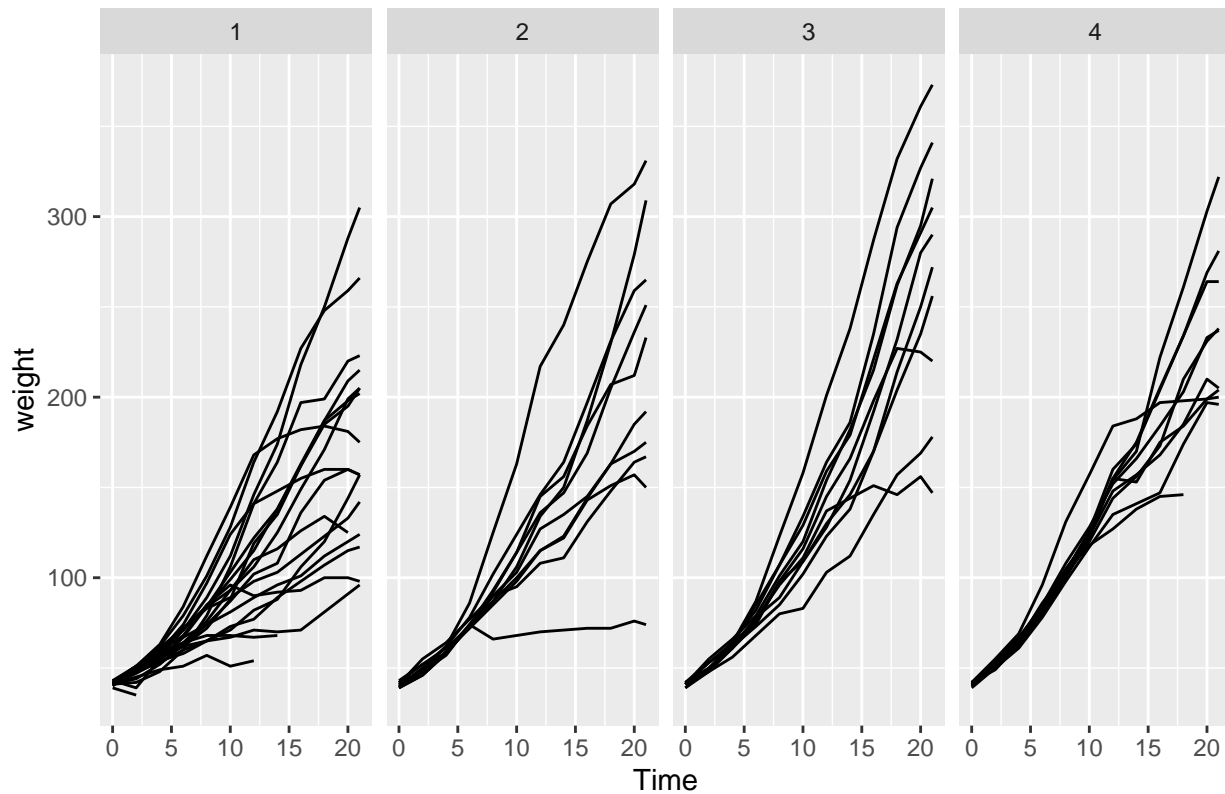
chickweight = datasets::ChickWeight;

# chicken weight by time, spaghetti plot

ggplot(data = chickweight,
  aes(x = Time, y = weight, group = Chick)) +
  geom_line() + facet_grid(~ Diet, scales='free') +
  ggtitle("Chicken Weight By Time Under Different Diet, Spaghetti Plot");

```

Chicken Weight By Time Under Different Diet, Spaghetti Plot



From the spaghetti plot, chicken tend to have a bigger increasement in weight under Diet 3.

```
chick.residual = chickweight %>%
  group_by(Time,Diet) %>%
  mutate(meanweight = mean(weight)) %>%
  ungroup() %>%
  mutate(residual = weight - meanweight) %>%
  group_by(Chick,Diet) %>%
  mutate(median.residual = median(residual)) %>%
  ungroup();

chick.stat = chick.residual %>%
  group_by(Diet) %>%
  mutate(min = min(median.residual),
         q1 = quantile(median.residual,c(.25)),
         q2 = quantile(median.residual,c(.5)),
         q3 = quantile(median.residual,c(.75)),
         max = max(median.residual)) %>%
  ungroup()%>%
  select(Diet,min, q1, q2, q3, max) %>%
  unique();

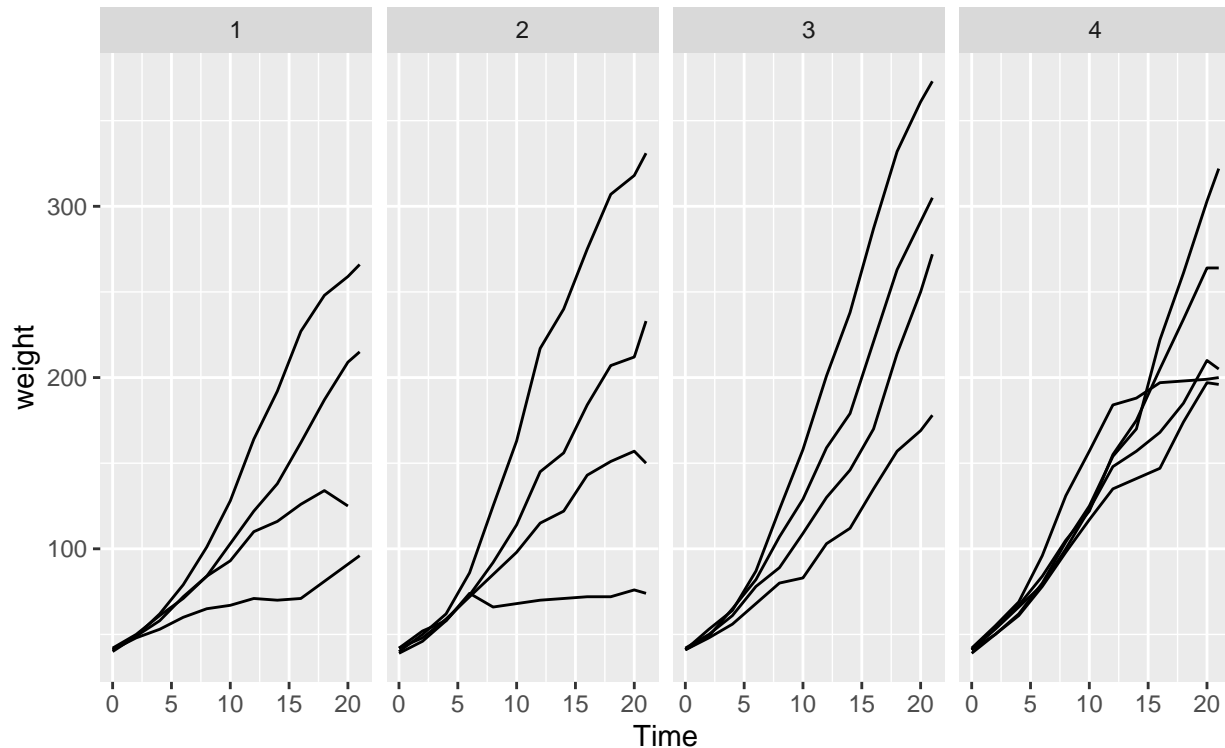
chick.id = chick.residual %>%
  filter(median.residual %in% as.matrix(chick.stat[,-1]));

ggplot(chick.id, aes(x = Time, y = weight, group = Chick)) +
```

```
geom_line()+facet_grid(~ Diet, scales='free') +
ggtitle("Weight By Time Under Different Diet, Spaghetti Plot",
        subtitle = "min, 25%, 50%, 75%, max of the weight residuals");
```

Weight By Time Under Different Diet, Spaghetti Plot

min, 25%, 50%, 75%, max of the weight residuals



- Chicken tend to have higher variation in weight as time goes by under Diet2. Least variation in weight was shown in chicken under diet4.

```
### Function 1 #####
##Input variables:
#number of row, number of columns, column data type, and column names
#Output variables:
#empty dataset
#Description:
#Function that create empty dataframe
```

```
emptydf = function(numrow, numcol, type, name){
  df = data.frame(matrix(NA, nrow = numrow, ncol = numcol));
  for (i in 1:numcol){
    print(type[i])
    if('numeric' == type[i]) {df[,i] = as.numeric(df[,i])
    colnames(df)[i] = name[i]};
    if('character' == type[i]) {df[,i] = as.character(df[,i])
    colnames(df)[i] = name[i]};
    if('logical' == type[i]) {df[,i] = as.logical(df[,i])
    colnames(df)[i] = name[i]};
    if('factor' == type[i]) {df[,i] = as.factor(df[,i])
    colnames(df)[i] = name[i]};
  }
```

```

}
return(df);
}

# chicken weight difference by time, box plot
chickweight.dif = emptydf(nlevels(as.factor(chickweight$Chick)),
                          nlevels(as.factor(chickweight$Time))+2 ,
                          c('character', 'character', rep('numeric',nlevels(as.factor(chickweight$Time))) ))
                          c('chick', 'diet', paste("dif", 0:nlevels(as.factor(chickweight$Time)) ,sep = "

## [1] "character"
## [1] "character"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"
## [1] "numeric"

chickweight$Time = as.character(chickweight$Time);
chickweight.wide = chickweight %>%
  group_by(Chick, Diet) %>%
  spread(Time,weight, fill=NA, sep = ".");

chickweight.wide = chickweight.wide[, c(1,2,3,9,12,13,14,4,5,6,7,8,10,11)];

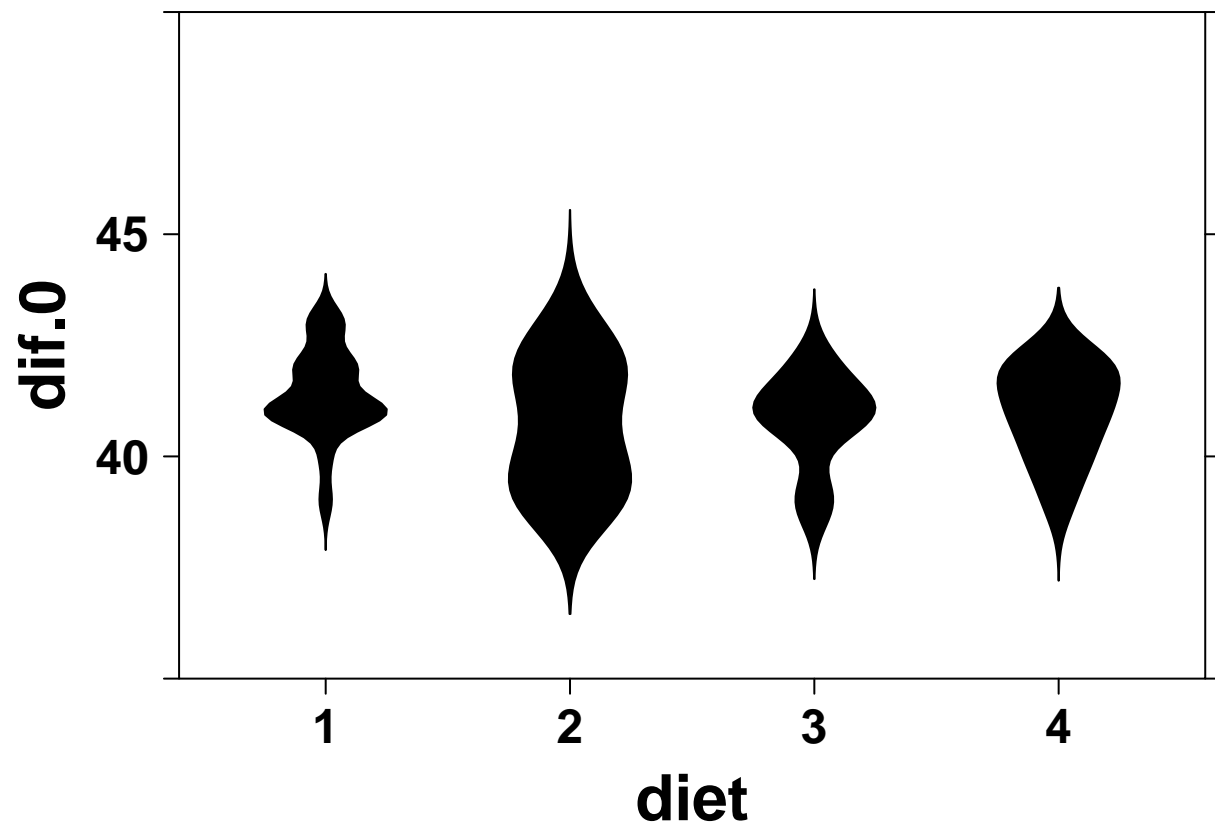
# Calculate weight difference between each measurement
for(i in 1: (nlevels(as.factor(chickweight$Time))-1) ){
  chickweight.dif [,i+3]= chickweight.wide[,i+3]-chickweight.wide[,i+2];
}

chickweight.dif[,1:3] = chickweight.wide[,1:3];

chickweight.dif.long = chickweight.dif %>%
  gather(interval,dif,dif.0:dif.11 );

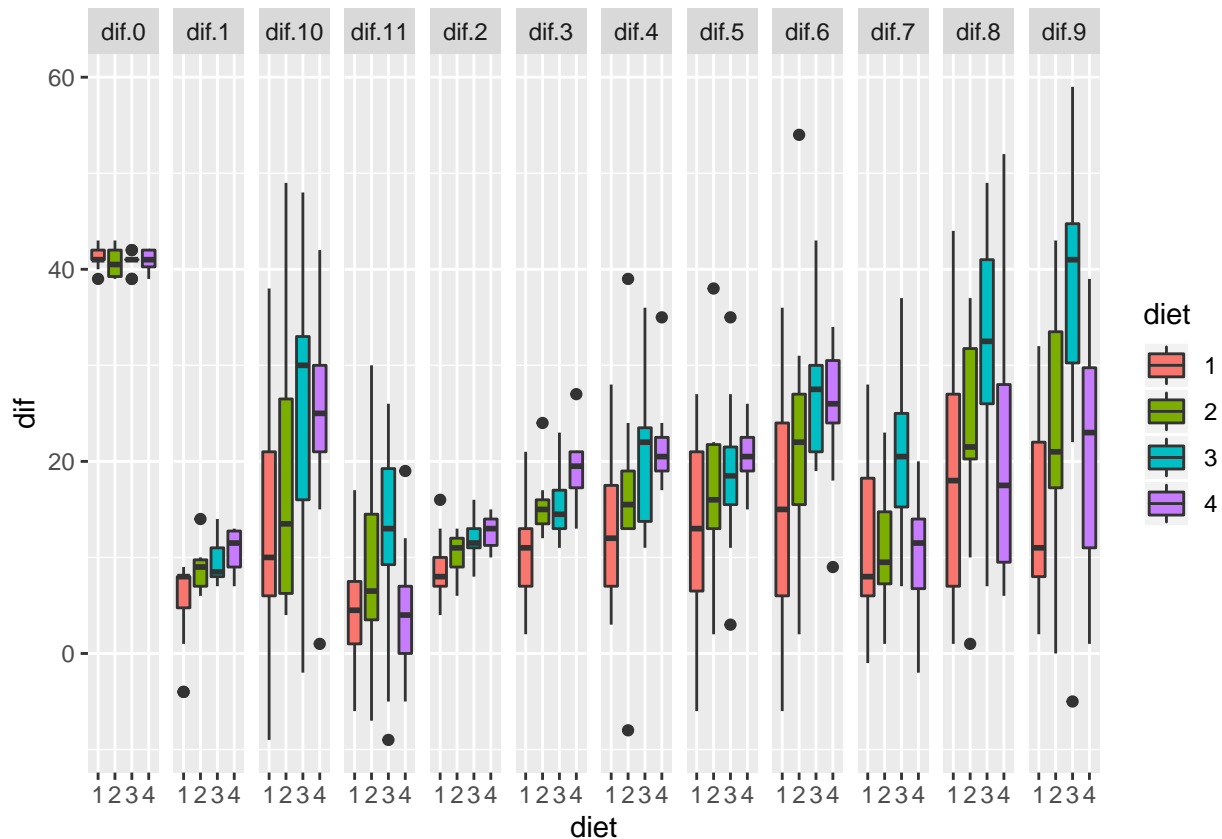
create.violinplot(formula = dif.0 ~ diet,
                  data = chickweight.dif,
                  ylimits = c(35,50)
);

```



```
ggplot(data = chickweight.dif.long,  
  aes(x = diet, y = dif, fill = diet)) +  
  geom_boxplot() +  
  facet_grid(. ~ interval);
```

Warning: Removed 22 rows containing non-finite values (stat_boxplot).



- From the boxplot, we can find out that there is not a big difference in weight between 4 groups at time0.
- Chicken on Diet 1 tend to have the least weight changes, while chicken on Diet 3 have the most weight changes during the time.
- Chicken on Diet 3 tend to have the most weight changes in the last few weeks.
- In the last few weeks the weight changes tend to have higher variance among diet groups.

Question 3

```
seq.control = read.table("/cloud/project/Q3_SeqControl_data", header = T);

# Ste 1 Reorder data
seq.control1 = seq.control[order(seq.control$yes.votes, decreasing = T),];

# Step 2 Create the CPCG bars
colour.scheme.large = c(
  'rosybrown1',
  'rosybrown4',
  'red',
  'darkred',
  'darkorange',
  'gold',
  'darkolivegreen3',
  'darkgreen',
  'aquamarine',
  'cyan4',
  'dodgerblue',
```



```

'darkblue',
'plum',
'magenta',
'darkorchid',
'purple4',
'gray70',
'gray30'
);

### Function 2 #####
##Input variables:
# data that need to create a bar plot, colour scheme
##Output variables:
# covariate bars
##Description:
##Function that create covariate bars

heatmap = function(data,colour){
  plot = create.heatmap(
    x = t(as.matrix(as.numeric(data))),
    clustering.method = "none",
    scale.data = FALSE,
    colour.scheme = colour,
    total.col = 12,
    force.grid.col = TRUE,
    grid.col = TRUE,
    print.colour.key = FALSE,
    # remove y-axis ticks
    yaxis.tck = 0,
    height = 1,
    xaxis.lab = NULL,
    yaxis.lab = NULL,
    yat = 1,
    yaxis.cex = 1);
  return(plot);
};

gene.heatmap = heatmap(data = seq.control1$CPCG,colour = colour.scheme.large[1:12]);
average.reads.start.heatmap = heatmap(data = seq.control1$Average.reads.start,
                                         colour = c("white", "deeppink"));
unique.start.points.heatmap = heatmap(data = seq.control1$Unique.start.points,
                                         colour = c("white", "darkblue"));
x.base.0.quality.heatmap = heatmap(data = seq.control1$X..Bases...0.quality,
                                     colour = c("white", "darkorange"));

# Create FFPE bar
seq.control1 = seq.control1 %>%
  dplyr::mutate(FFPE = ifelse("CPCG0102P" == CPCG | "CPCG0103P" == CPCG,1,0) );

FFPE.heatmap = heatmap(data = seq.control1$FFPE,
                        colour = c("white", "darkslategrey"));

```

```

# Step 5 Create the barplot
barplot.colour.choice = c("grey", "black");
barplot.colour = barplot.colour.choice[factor(seq.control1$outcome, levels = c(0,1))];

yes.votes.barplot = create.barplot(
  formula = yes.votes ~ c(1:72),
  border.col = 'transparent',
  data = seq.control1,
  col = barplot.colour,
  right.padding = 2,
  xaxis.cex = 0.5,
  abline.h = 0.5,
  abline.lty = 2,
  abline.col = 'darkgrey'
);

# Step 6 Create a legend for each of the covariates

# create legend for cpcgene sample heatmap
sample.legend = list(
  legend = list(
    colours = colour.scheme.large[1:12],
    title = expression(underline("Sample")),
    labels = levels(seq.control1$CPCG),
    continuous = FALSE
  )
);

# create legend for outcome heatmap
prep.legend = list(
  legend = list(
    colours = c('white', 'black'),
    labels = c('Frozen', 'FFPE'),
    title = expression(bold(underline('Sample preparation'))),
    continuous = FALSE
  )
);

# create legend for x..base....0.quality heatmap
qual.legend = list(
  legend = list(
    colours = c("white", "darkorange"),
    labels = c("97.0", "83.0"),
    title = expression(bold(underline('%Base > 0 quality'))),
    continuous = TRUE
  )
);

# create legend for unique.start.point.heatmap
l.s = scientific.notation(x=min(seq.control1$Unique.start.points),digits=2);

```

```

h.s = scientific.notation(x=max(seq.control1$Unique.start.points),digits=2);

uni.start.legend = list(
  legend = list(
    colours = c("white", "darkblue"),
    labels = c(h.s, l.s),
    title = expression(bold(underline('Unique start point'))),
    continuous = TRUE
  )
);

# create legend for average.reads.start.heatmap
l.r = round(min(seq.control1$Average.reads.start),2);
h.r = round(max(seq.control1$Average.reads.start),2);

ave.start.legend = list(
  legend = list(
    colours = c("white", "deeppink"),
    labels = c(h.r, l.r),
    title = expression(bold(underline('Unique start point'))),
    continuous = TRUE
  )
);

#combine all covariates legends
covariate.legend = c(
  sample.legend,
  prep.legend,
  qual.legend,
  uni.start.legend,
  ave.start.legend
);

legends1 = BoutrosLab.plotting.general::legend.grob(
  legends = covariate.legend,
  title.cex = 0.75,
  title.just = 'left',
  label.cex = 0.65,
  size = 1.5,
  between.row = 1.0,
  between.col = 0.5,
  layout = c(1,6)
);

```

```
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
```

```
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
```

```

# create legend for barplot
barplot.legend = list(
  legend = list(
    colours = c('grey', 'black'),

```

```

    labels = c(
      as.expression(substitute(x < '50x',list(x = ' '))),
      as.expression(substitute(x >= '50x',list(x = ' ')))
    ),
    title = expression(underline('Observed'))
  )
);

legends2 = BoutrosLab.plotting.general::legend.grob(barplot.legend);

## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL

# Multiplot

plot.objects = list(
  average.reads.start.heatmap,
  unique.start.points.heatmap,
  x.base.0.quality.heatmap,
  FFPE.heatmap,
  gene.heatmap,
  yes.votes.barplot
);

# identify where plotting objects should be placed in the multiplot
yat.vals = list();
for (n in 1:(length(plot.objects)-1)) {
  yat.vals = c(yat.vals, list(NULL));
}

# combine plots
create.multiplot(
  plot.objects = plot.objects,
  #filename = 'testing_votes.tiff',
  panel.heights = c(1, rep(0.05, length(plot.objects)-1)),
  yat = c(yat.vals, list(seq(0,1,0.25))),
  yaxis.cex = 1.15,
  ylab.label = c(' ', 'Fraction Yes-Votes', ' ', ' ', ' ', ' '),
  ylab.padding = 6.5,
  right.padding = 35,
  bottom.padding = -5,
  xat = NULL,
  y.spacing = 0.5,
  ylimits = list(c(0.9,1), c(0.9,1), c(0.9,1), c(0.9,1), c(0.9,1), c(0.00,1.05)),
  legend = list(
    inside = list(
      fun = legends1,
      x = 1.02,
      y = 1
    ),
    inside = list(
      x = 0.85,
      y = 0.98,
      fun = legends2
    )
  )
)

```

```

),

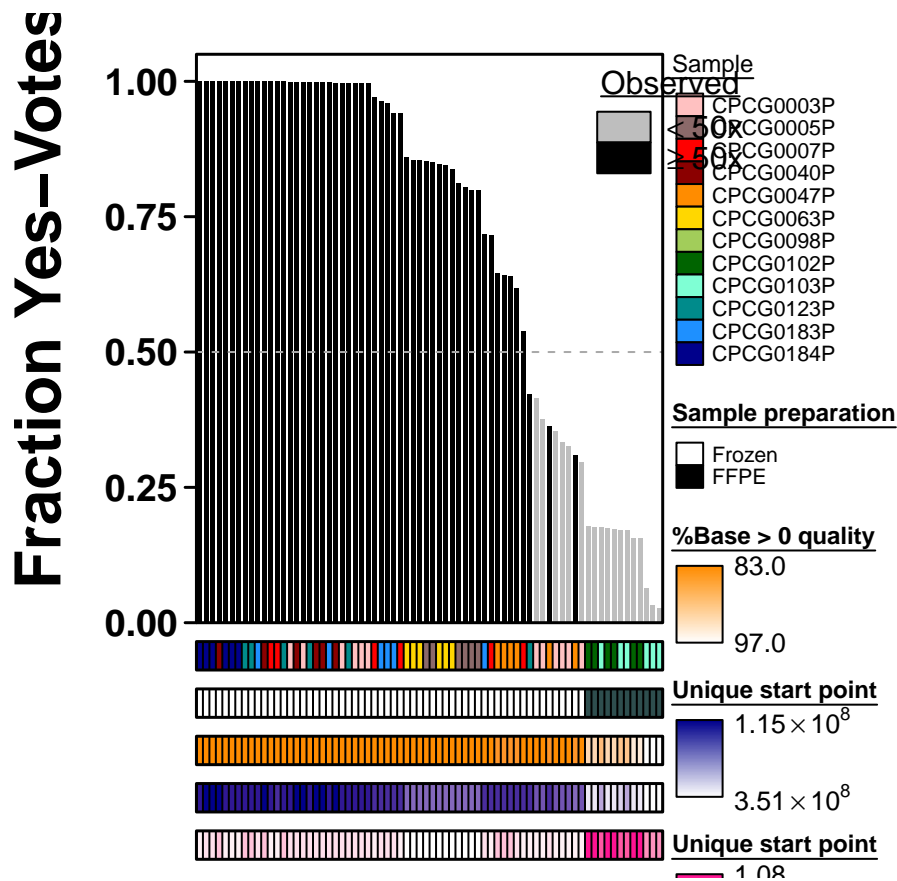
print.new.legend = TRUE,
width = 12,
height = 6,
resolution = 1200,
);

```

```

## Warning in formals(fun): argument is not a function
## Warning in formals(fun): argument is not a function
## Warning in formals(fun): argument is not a function
## Warning in formals(fun): argument is not a function
## Warning in formals(fun): argument is not a function

```



Question 4

```

het = read.table("/cloud/project/Q4_HetStudy_data.txt", header=T);

# create covariate table
covariate = as.data.frame(colnames(het[,1:28]));
names(covariate) = c("id");

```

```

covariate$sample = substr(covariate$id, 1,8);
covariate$cohort = c(rep("Bx",5),rep("Sx",23));
# convert gleason score 3+4 as 1, 4+3 as 2, 4+4 as 3
covariate$gleason.score = c(1,1,2,1,2,1,2,2,3,3,1,1,1,1,1,2,1,2,2,2,1,1,1,3,2,2,2,1);
covariate$gleason.score.plus = c(1,rep(NA,14),rep(1,4),rep(NA,9));

covariate$tissue.type = substr(covariate$id, 9,10);

# create matrix for gleason score positive
gleason.score.plus.matrix = data.frame(matrix(NA, nrow = 28, ncol = 2));
gleason.score.plus.matrix = cbind(gleason.score.plus.matrix,covariate$gleason.score.plus);
gleason.score.plus.matrix = cbind(gleason.score.plus.matrix,data.frame(matrix(NA, nrow = 28, ncol = 1)));

covariate.numeric = data.frame(lapply(covariate, as.character),stringsAsFactors = FALSE);
covariate.numeric = covariate.numeric[,2:6];
covariate.numeric = covariate.numeric %>%
  dplyr::select(-c("gleason.score.plus"));

# convert covariate data to numeric
# create sample id
covariate.numeric = covariate.numeric %>%
  dplyr::mutate(sample.id = as.integer(as.factor(sample)));
covariate.numeric$sample = as.character(covariate.numeric$sample.id);
covariate.numeric = covariate.numeric[,-5]

#convert Bx (biopsy) as 11, and Sx (surgery) as 12
covariate.numeric$cohort["Bx" == covariate.numeric$cohort] = 11;
covariate.numeric$cohort["Sx" == covariate.numeric$cohort] = 12;

#convert gleason score
covariate.numeric$gleason.score[1 == covariate.numeric$gleason.score] = 13;
covariate.numeric$gleason.score[2 == covariate.numeric$gleason.score] = 14;
covariate.numeric$gleason.score[3 == covariate.numeric$gleason.score] = 15;

# concert "F0" samples are Frozen(2), while all other samples are FPPE(1)
covariate.numeric$tissue.type["F0" != covariate.numeric$tissue.type] = 17;
covariate.numeric$tissue.type["F0" == covariate.numeric$tissue.type] = 16;

# set colour scheme
sample.colour = c('blue','purple','green','orange','yellow','black','wheat4','green4','grey','red4');
cohort.colour = c('royalblue', 'pink');
gleason.score.colour = c("yellow1", "orange","red");
tissue.type.colour = c('Frozen' = colours()[532], 'FPPE' = colours()[557]);

# create covariate bar on the right
covariate.bar = create.heatmap(x = t(data.matrix(covariate.numeric)),
                              clustering.method = "none",

```

```

        print.colour.key = FALSE,
        total.colours = 17,
        colour.scheme = c(sample.colour,cohort.colour,gleason.score.colour,tiss
        at=seq(0.5,17.5,1),
        # add row lines
        force.grid.col = TRUE,
        grid.col = TRUE,
        grid.row = TRUE,
        row.colour = "black",
        col.colour = "black",
        row.pos = which(1 == t(gleason.score.plus.matrix[1:28,1:4]), arr.ind = '
        col.pos = which(1 == t(gleason.score.plus.matrix[1:28,1:4]), arr.ind = '
        cell.text =rep("+", 5),
        text.cex = 1,
        xaxis.tck = 0,
        yaxis.tck = 0
    );

    # create fraction plot
    het.frac = het %>%
      dplyr::select(-Baca, -Berger, -Weischenfeldt);
    het.frac$frac = rowSums(het.frac != 0)/28;

    het.frac = het.frac%>%
      dplyr::mutate(
        ends = sapply( strsplit(rownames(het),"-", fixed = TRUE),tail, 1),
        chr = sapply( strsplit(rownames(het),":", fixed = TRUE),head, 1),
        number = c(1:3113));

    het.frac= het.frac%>%
      dplyr::mutate(chr.num = gsub("chr", "", chr));

    frac.plot = create.barplot(formula = frac ~ number,
                              data = het.frac,
                              main = NULL,
                              stack = FALSE,
                              xlab.top.label = NULL,
                              xlab.label = NULL,
                              ylab.label = "Fraction",
                              ylab.cex = 0.8,
                              xaxis.tck = 0,
                              xaxis.lab = rep('',3113),
                              ylimits = c(0, 0.5),
                              yat = seq(0,0.6,0.25),
                              yaxis.cex = 0.6,
                              yaxis.tck = c(1,0)
    );

    # create literature covariate bars
    het.literature = cbind(het.frac[,30:32],het[,29:31]);

```

```

het.literature = data.frame(lapply(het.literature, as.character), stringsAsFactors = FALSE);

# set colour scheme
literature.colour = c('white', 'darkred', 'cornflowerblue', 'darkolivegreen4');

# create
publication.bar = create.heatmap(x = data.matrix(het.literature[,4:6]),
                                clustering.method = "none",
                                print.colour.key = FALSE,
                                #total.colours = 4,
                                colour.scheme = c( literature.colour ),
                                # add row lines
                                at=seq(-0.5,3.5,1),

                                col.lines = which("1000000" == het.literature$ends),
                                force.grid.col = TRUE,
                                grid.col = TRUE,
                                force.grid.row = TRUE,
                                grid.row = TRUE,
                                row.colour = "black",
                                yaxis.tck = 0
                                );

# create main heatmap
literature.colour = c('white', 'cornflowerblue', 'darkolivegreen4', 'darkred');

# create x axis location
het.frac.plot = het.frac%>%
  dplyr::group_by(chr.num) %>%
  dplyr::mutate(x.location = ifelse (number == round(mean(number), digits = 0), 1, 0));

het.frac.plot = data.frame(lapply(het.frac.plot, as.character), stringsAsFactors = FALSE);

# create y axis location
covariate = covariate %>%
  dplyr::group_by(sample) %>%
  dplyr::mutate(first = row_number() == min( row_number() ));

# create main heat map
main.heatmap = create.heatmap(x = data.matrix(het.frac.plot[,1:28]),
                              clustering.method = "none",
                              print.colour.key = FALSE,

                              total.colours = 4,
                              colour.scheme = c(literature.colour),
                              at = seq(-0.5, 3.5, 1),
                              # add row lines
                              row.lines = which(TRUE == covariate$first)+0.5,

```



```

        grid.row = TRUE,

        # add col lines
        col.lines = which("1000000" == het.literature$ends),
        force.grid.col = TRUE,
        grid.col = TRUE,

        #set labels for x and y axis
        xaxis.lab = c(1:22,"X", "Y"),
        xat = which(1 == het.frac.plot$x.location),
        xaxis.cex = 0.6,
        xaxis.rot = 0,
        yaxis.lab = covariate$tissue.type,
        yaxis.tck = c(1,0),
        yaxis.cex = 0.6

        #row.colour = "black",
        #col.colour = "black"
);

# create the bottom notation heatmap
notation = matrix(1:4, nrow = 1);
notation.heatmap = create.heatmap(x = notation,
    clustering.method = "none",
    print.colour.key = FALSE,
    total.colours = 4,
    colour.scheme = c('white', 'cornflowerblue','darkolivegreen4','darkred4'),
    at = seq(0.5,4.5,1),
    force.grid.col = TRUE,
    grid.col = TRUE,

    xaxis.lab = c("", "None", "CTX", "ITX","INV"),

    xaxis.cex = 0.6,
    xaxis.rot = 0,
    yaxis.lab = NULL,
    yaxis.cex = 0.6,
    yaxis.tck = 0,
    right.padding = 0
);

#create legend on the left

# create legend for samples
pid.legend = list(
    legend = list(
        colours = c( 'blue','purple','green','orange','yellow','black','wheat4','green4','grey','red4'),
        labels = unique(covariate$sample),
        title = expression(bold(underline('Patient ID'))),

```

```

    continuous = FALSE
  )
);

# create legend for cohort
cohort.legend = list(
  legend = list(
    colours = c('pink', 'royalblue' ),
    labels = c('Sx', 'Bx'),
    title = expression(bold(underline('Cohort'))),
    continuous = FALSE
  )
);

# create legend for Gleason score
gs.legend = list(
  legend = list(
    colours = c('yellow', 'orange', 'red' ),
    labels = c('3+4', '4+3', '4+4'),
    title = expression(bold(underline('Gleason score'))),
    continuous = FALSE
  )
);

# create legend for Tissue type
tissue.legend = list(
  legend = list(
    colours = c(colours()[532], colours()[557]),
    labels = c('FFPE', 'Frozen'),
    title = expression(bold(underline('Tissue type'))),
    continuous = FALSE
  )
);

# create legend for Publication
pub.legend = list(
  legend = list(
    colours = c('darkred', 'cornflowerblue', 'darkolivegreen4'),
    labels = c('Baca', 'Berger', "Weischenfeldt"),
    title = expression(bold(underline('Publication'))),
    continuous = FALSE
  )
);

left.legends = legend.grob(
  legends = c(pid.legend, cohort.legend, gs.legend, tissue.legend, pub.legend),
  title.cex = 0.75,
  title.just = 'left',
  label.cex = 0.65,
  size = 1,
  between.row = 1.0,

```

```

    between.col = 0.5
);

## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
## Warning in FUN(X[[i]], ...): 'x' is NULL so the result will be NULL
# create multiplot
plot.objects = list(frac.plot,publication.bar, main.heatmap,covariate.bar,notation.heatmap );

create.multipanelplot(
  plot.objects = plot.objects,
  #filename = 'HetStudy.tiff',

  plot.objects.heights = c(0.6,0.3,1.4,0.2),
  plot.objects.widths = c(1.1, 0.1),

  layout.skip = c(FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE),

  layout.height = 4,
  layout.width = 2,

  #yaxis.cex = 1.15,
  ylab.axis.padding = 3.5,
  #right.padding = 35,
  #bottom.padding = 1,
  y.spacing = c(-1,-1,-1),
  x.spacing = 0,

  legend = list(
    left = list(
      fun = left.legends,
      x = 1.02,
      y = 1
    )
  ),

  left.legend.padding = 2,

  width = 12,
  height = 6,
  resolution = 1200
);

## Warning in rbind(plot.objects.heights, y.spacing): number of columns of
## result is not a multiple of vector length (arg 2)

```

Patient ID

CPCG0001
CPCG0003
CPCG0006
CPCG0020
CPCG0042
CPCG0099
CPCG0102
CPCG0103
CPCG0183
CPCG0184

Cohort

Sx
Bx

Gleason score

3+4
4+3
4+4

Tissue type

FFPE
Frozen

Publication

Baca
Berger
Weischenfeldt

