Hallmarks of TCR:peptide:MHC interactions inferred from structural data mining.

Mikhail Shugay

August 15, 2016

```
# All imports

library(plyr)
library(ggplot2)
library(reshape2)
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
# Helper functions

corrt <- function(r,n) r*sqrt((n-2)/(1-r^2))
calcpvalr <- function(r, n=20) {
  pval <- pt(corrt(r, n), n)
  min(pval, 1-pval)
}

p.annot <- function(p.val) {
  ifelse(p.val<0.001, "***", ifelse(p.val<0.01, "**", ifelse(p.val<0.05, "*", ifelse(p.val<0.1, "+", "")
}
```

Structural data used in the study

TCR:peptide:MHC complexe entries were obtained from PDB by a batch query with corresponding keywords. Complex records were then automatically annotated using in-house scripts that performed:

- TCR, MHC and antigen flags were assigned to chain records
- Antigen and host species were inferred
- MHC alleles were assigned using blast search against a database of MHC protein sequences manually assembled from public databases
- TCR partitioning into CDR and Framework regions was performed using custom IgBlast wrapper

This dataset was then used to generate a flat table with annotated TCR:antigen amino acid pairs.

```
df <- read.table("../result/structure.txt", header=T, sep="\t")
df$energy[is.na(df$energy)] <- 0

df <- ddply(df, .(tcr_v_allele), transform,
            tcr_chain = factor(substr(as.character(tcr_v_allele[1]), 1, 3)))
```

The total number of complexes that were successfully annotated was

```
length(levels(df$pdb_id))
```

```
[1] 103
```

and the total number of amino acid pairs was

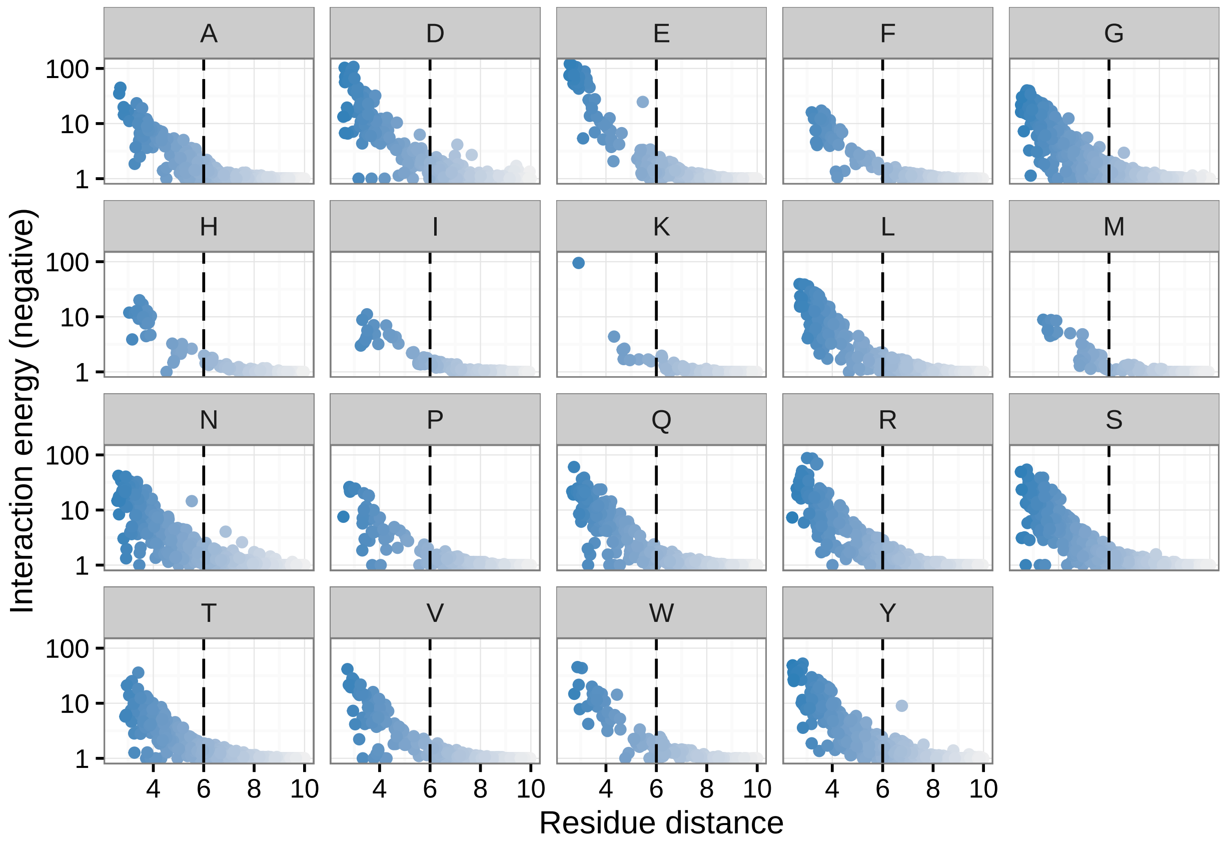
```
nrow(df)
```

```
[1] 51904
```

Selecting distance threshold for amino acid contacts

GROMACS software was used to calculate point energies for amino acid contacts using TCR:peptide:MHC structures. Each amino acid pair record in the database was then assigned with an interaction energy value using in-house scripts. Distances between residues were computed as the minimal distance between a pair of atoms using Bio.PDB python package. Interaction energies grouped by CDR3 amino acid are plotted against residue distances below. Selected distance threshold for contacting residues is shown as a vertical line, the same value is used further throughout the manuscript.

```
DIST_THRESHOLD = 6

ggplot(subset(df, distance <= 10 & energy <= 0),
       aes(x=distance, y=-energy+1, colour=distance)) +
  geom_point() + geom_vline(xintercept = DIST_THRESHOLD, linetype = "longdash") +
  scale_y_log10(name = "Interaction energy (negative)") + xlab("Residue distance") +
  scale_colour_gradient(guide = FALSE, low = "#2c7fb8", high = "#f0f0f0") +
  facet_wrap(~aa_tcr) + theme_bw()
```



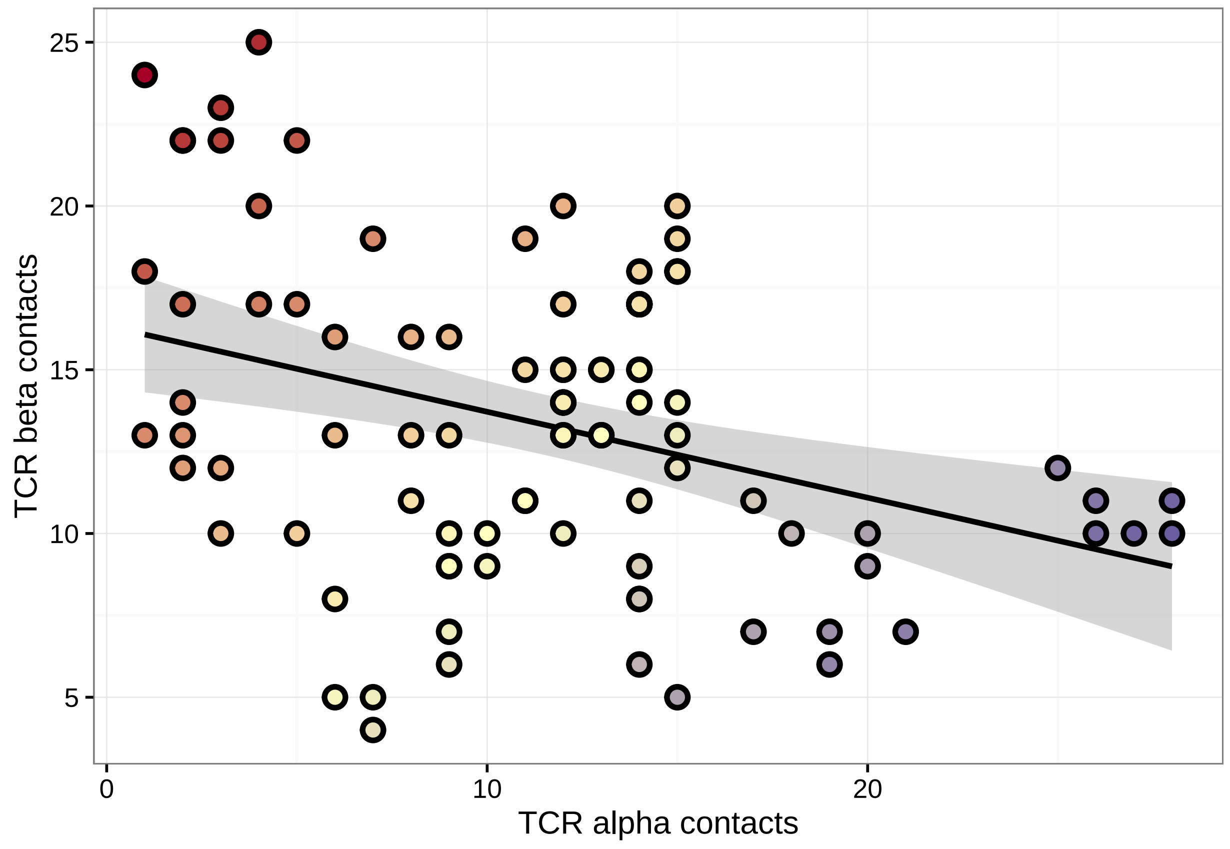
Studying distribution of TCR:antigen contact residues

Number of TCR alpha and beta chain contacts is inversely correlated

The first hypothesis we've tested was the correlation between number of antigen contacts with TCR alpha and beta chains within the same TCR:peptide:MHC complex. We've focused on CDR3 region as the one having a critical role in antigen specificity and MHCI molecules. Based on previous observations of Yokosuka et al. 2002 and Turner et al. 1997, we have hypothesized that while the number of CDR3-antigen contacts per complex is quite stable, CDR3 regions of alpha and beta chains "compete" for antigen binding, resulting in TCRs in which one of the chains has a dominant role in antigen binding. Indeed, an inverse correlation between number of TCR alpha and beta CDR3-antigen contacts was observed as shown in the plot below.

```
df.s <- ddply(subset(df, tcr_region == "CDR3"),
              .(pdb_id, tcr_chain),
              summarize,
              csum= sum(distance <= DIST_THRESHOLD),
              len = mean(len_tcr)
              )

no_contact_pdb <- unique(with(subset(df.s, csum == 0), paste(pdb_id, tcr_chain)))

df.s <- subset(df.s, !(paste(pdb_id, tcr_chain) %in% no_contact_pdb))
df.s <- merge(subset(df.s, tcr_chain=="TRA"), subset(df.s, tcr_chain=="TRB"),
              by="pdb_id", suffixes = c(".TRA",".TRB"))
```

```r
ggplot(df.s, aes(x=csum.TRA, y=csum.TRB)) +
  geom_smooth(method="lm", color="black") +
  geom_point(size=4) +
  geom_point(size=2, aes(color=csum.TRA-csum.TRB)) +
  scale_color_gradient2(guide = FALSE, low = "#a50026", mid = "#ffffbf", high = "#313695") +
  scale_x_continuous(name = "TCR alpha contacts") +
  scale_y_continuous(name = "TCR beta contacts") +
  theme_bw()
```



```r
# The statistical significance of the observed dependency is given below.

summary(lm(formula = csum.TRA ~ csum.TRB, data = df.s))
```

```
##
## Call:
## lm(formula = csum.TRA ~ csum.TRB, data = df.s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5826  -4.4805   0.4174   3.8919  15.4684
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.7507     1.8676   9.504 2.78e-15 ***
## csum.TRB     -0.4745     0.1319  -3.597 0.000524 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.015 on 91 degrees of freedom
## Multiple R-squared:  0.1245, Adjusted R-squared:  0.1148
## F-statistic: 12.94 on 1 and 91 DF,  p-value: 0.0005238
```

```
summary(lm(formula = csum.TRA / len.TRA ~ I(csum.TRB / len.TRB), data = df.s))
```

```
##
## Call:
## lm(formula = csum.TRA/len.TRA ~ I(csum.TRB/len.TRB), data = df.s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.77556 -0.28688  0.00986  0.22202  1.00204
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.2922     0.1383   9.343 6.04e-15 ***
## I(csum.TRB/len.TRB)  -0.4664     0.1371  -3.402 0.000996 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4063 on 91 degrees of freedom
## Multiple R-squared:  0.1128, Adjusted R-squared:  0.1031
## F-statistic: 11.57 on 1 and 91 DF,  p-value: 0.0009957
```

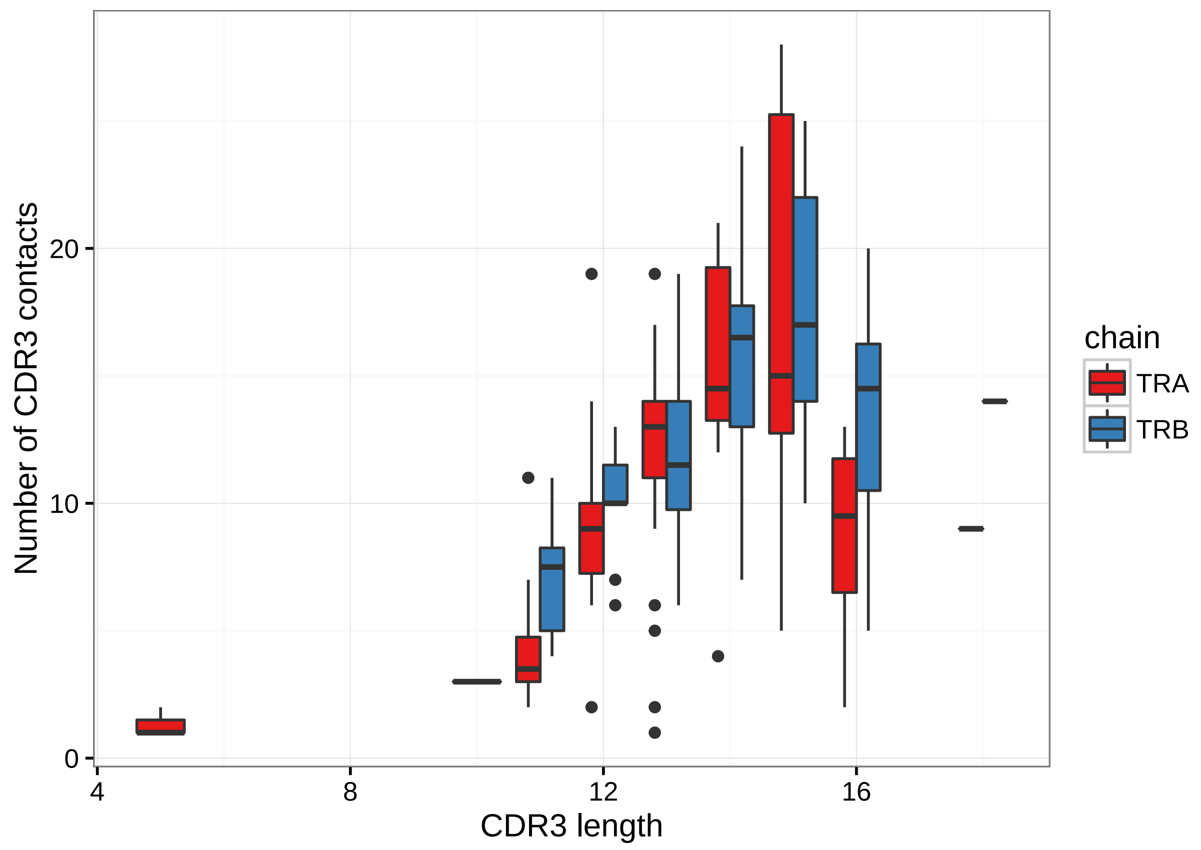
Distribution of total number of CDR3 contacts per complex is given below.

```
summary(ddply(df.s, .(pdb_id), summarize, csum=csum.TRA+csum.TRB))
```

```
##      pdb_id        csum
##  1ao7   : 1   Min.   :11.00
##  1bd2   : 1   1st Qu.:20.00
##  1d9k   : 1   Median :25.00
##  1fyt   : 1   Mean   :24.76
##  1g6r   : 1   3rd Qu.:29.00
##  1j8h   : 1   Max.   :39.00
##  (Other):87
```

Number of antigen contacts is correlated with CDR3 length:

```
df.s1 <- data.frame(csum = df.s$csum.TRA, len = df.s$len.TRA, chain="TRA")
df.s1 <- rbind(df.s1, data.frame(csum = df.s$csum.TRB, len = df.s$len.TRB, chain="TRB"))

ggplot(df.s1, aes(x=len, y=csum)) +
  geom_boxplot(aes(group=interaction(chain,len),fill=chain)) +
  #geom_smooth(method="lm", aes(color=chain)) +
  #geom_point(size=4, color="black") +
  #geom_point(aes(color=chain), size=2, alpha=0.5) +
  scale_x_continuous("CDR3 length") +
  scale_y_continuous("Number of CDR3 contacts") +
  scale_fill_brewer(palette = "Set1") + theme_bw()
```



```r
summary(lm(formula = csum.TRA ~ len.TRA, data = df.s))
```

```
##
## Call:
## lm(formula = csum.TRA ~ len.TRA, data = df.s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6734  -3.6734   0.2536   2.7718  13.8083
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -8.0346     3.6108  -2.225   0.0285 *
## len.TRA       1.4818     0.2715   5.458 4.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.58 on 91 degrees of freedom
## Multiple R-squared:  0.2466, Adjusted R-squared:  0.2384
## F-statistic: 29.79 on 1 and 91 DF,  p-value: 4.126e-07
```

```r
summary(lm(formula = csum.TRB ~ len.TRB, data = df.s))
```

```
##
## Call:
```

```
## lm(formula = csum.TRB ~ len.TRB, data = df.s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2639  -2.2901  -0.2901   2.7099  10.3853
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.9294     4.0022  -1.232    0.221
## len.TRB       1.3246     0.2883   4.595 1.39e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.306 on 91 degrees of freedom
## Multiple R-squared:  0.1883, Adjusted R-squared:  0.1794
## F-statistic: 21.11 on 1 and 91 DF,  p-value: 1.391e-05
```

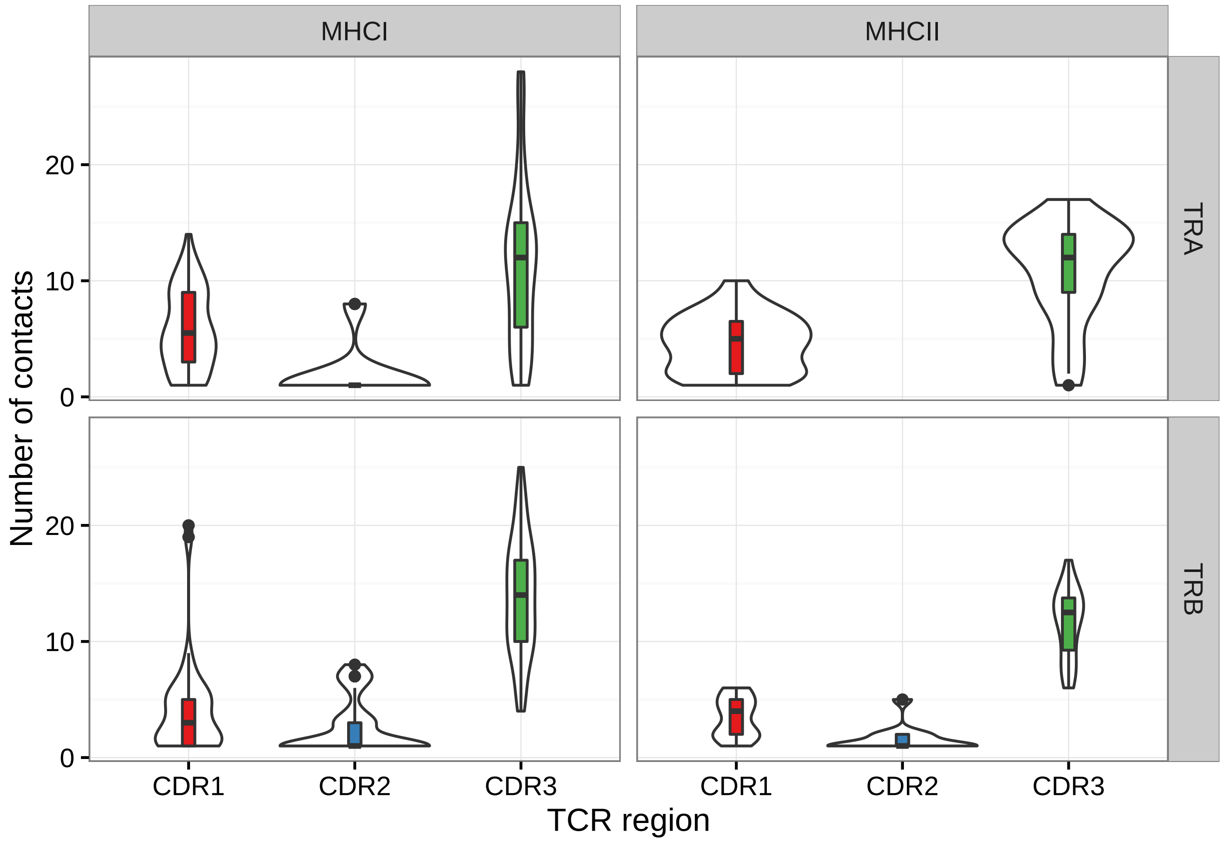
TCR region and MHC contact preferences

Next, we've extended our analysis on other TCR regions (CDR1 and 2 in germline) and MHCII. Surprisingly, CDR1, but not CDR2, appears to confer a substantial number of TCR-antigen contacts, comparable to that of CDR3.

> Given the apparent role of CDR1 in peptide recognition, we present a hypothesis (yet to be tested) stating that the observed germline Variable (V) segment restriction and cross-reactivity to MHC alleles Garcia 2012 can be at least partially explained by thymic selection due to the CDR1-encoded specificity to the self-peptide pool presented by a certain MHC. Note that this hypothesis is strengthened by the study of Cole et al 2014 demonstrating that TCR-peptide specificity overrides affinity-enhancing TCR-MHC interactions.

Additionally, it appears that MHCI complexes have a higer number of TCR-antigen contacts than MHCII complexes. Our observations are summarized in the figure below.

```r
df.r <- ddply(df, .(pdb_id, tcr_chain, tcr_region, mhc_type), summarize,
             csum=sum(distance <= DIST_THRESHOLD))

ggplot(subset(df.r, csum > 0), aes(x=tcr_region,group=tcr_region,y=csum)) +
  geom_violin() +
  geom_boxplot(aes(fill=tcr_region),width=0.1) +
  xlab("TCR region") + ylab("Number of contacts") +
  scale_fill_brewer(guide = F, palette = "Set1") +
  facet_grid(tcr_chain~mhc_type) + theme_bw()
```



Statistical significance of results described in this section is provided below.

```r
a <- aov(csum~tcr_region + tcr_chain + mhc_type, df.r)
summary(a)
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)    
## tcr_region     2  12607    6304 340.264 < 2e-16 ***
## tcr_chain      1     10      10   0.538 0.46345    
## mhc_type       1    202     202  10.914 0.00101 ** 
## Residuals    612  11338      19                    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(a, "mhc_type")
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = csum ~ tcr_region + tcr_chain + mhc_type, data = df.r)
## 
## $mhc_type
##                 diff       lwr        upr     p adj
## MHCII-MHCI -1.30093 -2.074274 -0.5275864 0.0010102
```

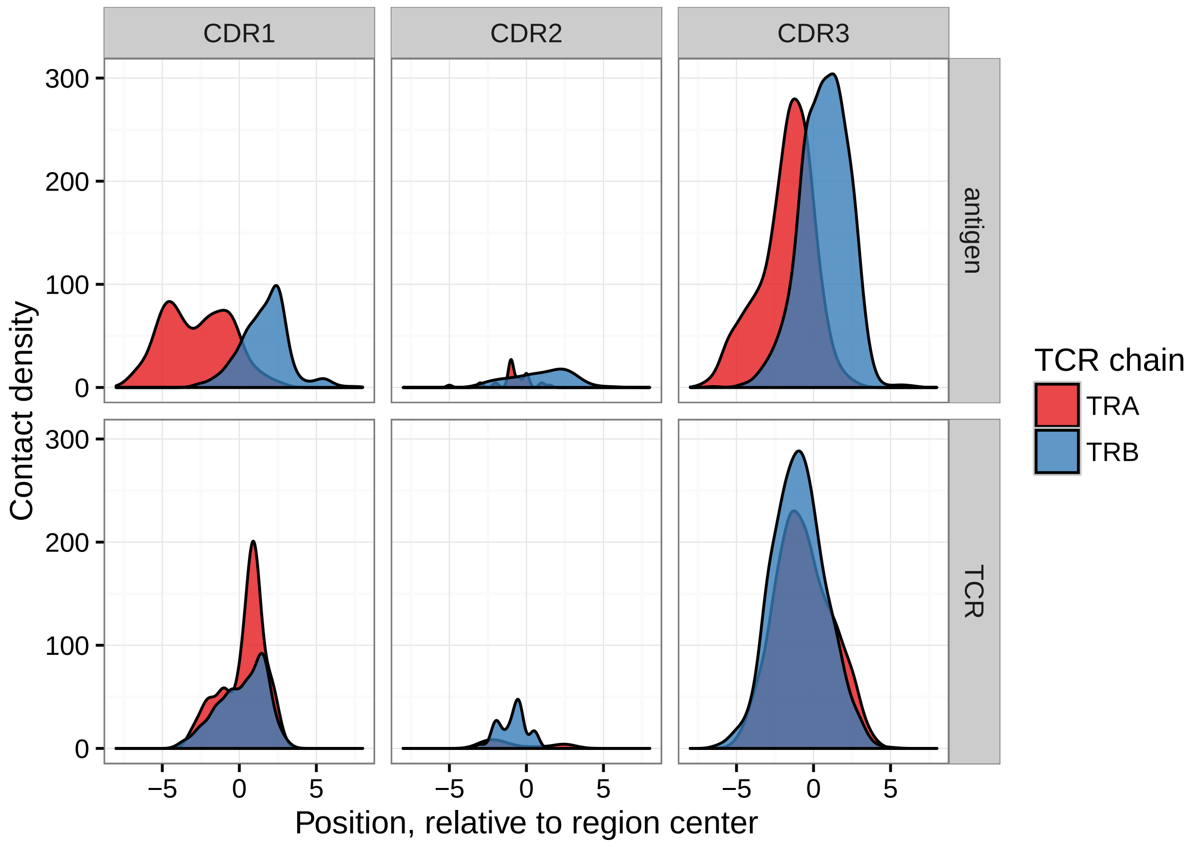
```
TukeyHSD(a, "tcr_region")
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = csum ~ tcr_region + tcr_chain + mhc_type, data = df.r)
##
## $tcr_region
##                diff       lwr       upr p adj
## CDR2-CDR1 -3.317641 -4.315244 -2.320039     0
## CDR3-CDR1  7.490291  6.493903  8.486680     0
## CDR3-CDR2 10.807933  9.810330 11.805535     0
```

Contact residues are tightly clustered on TCR and antigen length

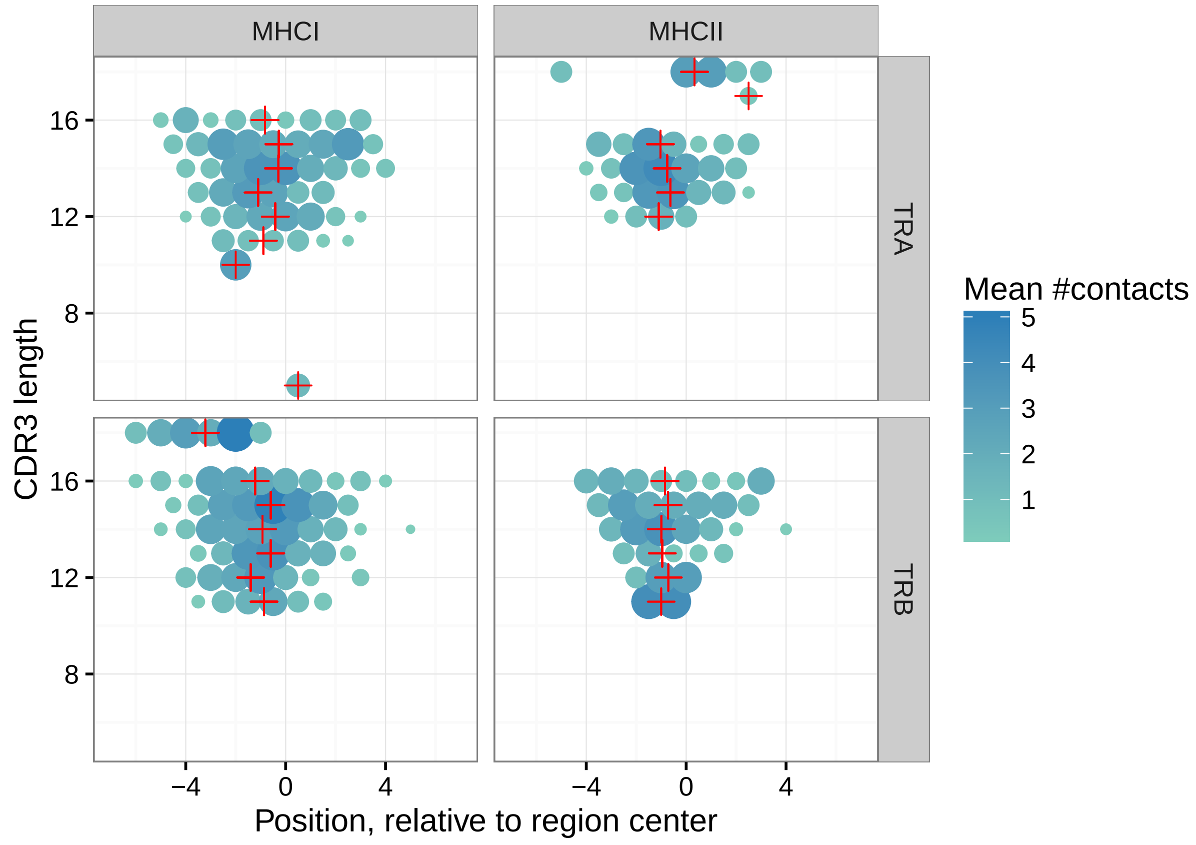
We have next studied the distribution of interacting residues by their position on antigen and CDR sequences. Below are the plots of contact distribution grouped by TCR region conferring the contact (columns) and contact residue parent sequence (rows). CDR contacts tend to cluster near the center of corresponding region. Note the clear difference between TCR alpha and beta contacts that are closer to **N** and **C** terminus of the antigen peptide respectively.

```
df.p1 <- ddply(df, .(pdb_id, tcr_chain, tcr_region, mhc_type, pos_tcr), summarize,
              pos_norm = mean(pos_tcr - len_tcr / 2),
              contacts = sum(distance <= DIST_THRESHOLD))
df.p1$pos_tcr <- NULL
df.p1$sequence <- "TCR"

df.p2 <- ddply(df, .(pdb_id, tcr_chain, tcr_region, mhc_type, pos_antigen), summarize,
              pos_norm = mean(pos_antigen - len_antigen / 2),
              contacts = sum(distance <= DIST_THRESHOLD))
df.p2$pos_antigen <- NULL
df.p2$sequence <- "antigen"

df.p <- rbind(df.p1, df.p2)

ggplot(subset(df.p, contacts > 0),
       aes(x=pos_norm, weight=contacts, fill=tcr_chain)) +
  geom_density(alpha=0.8) +
  scale_fill_brewer(name = "TCR chain", palette = "Set1") +
  ylab("Contact density") +
  scale_x_continuous(name = "Position, relative to region center", limits=c(-8, 8)) +
  facet_grid(sequence~tcr_region) +
  theme_bw()
```



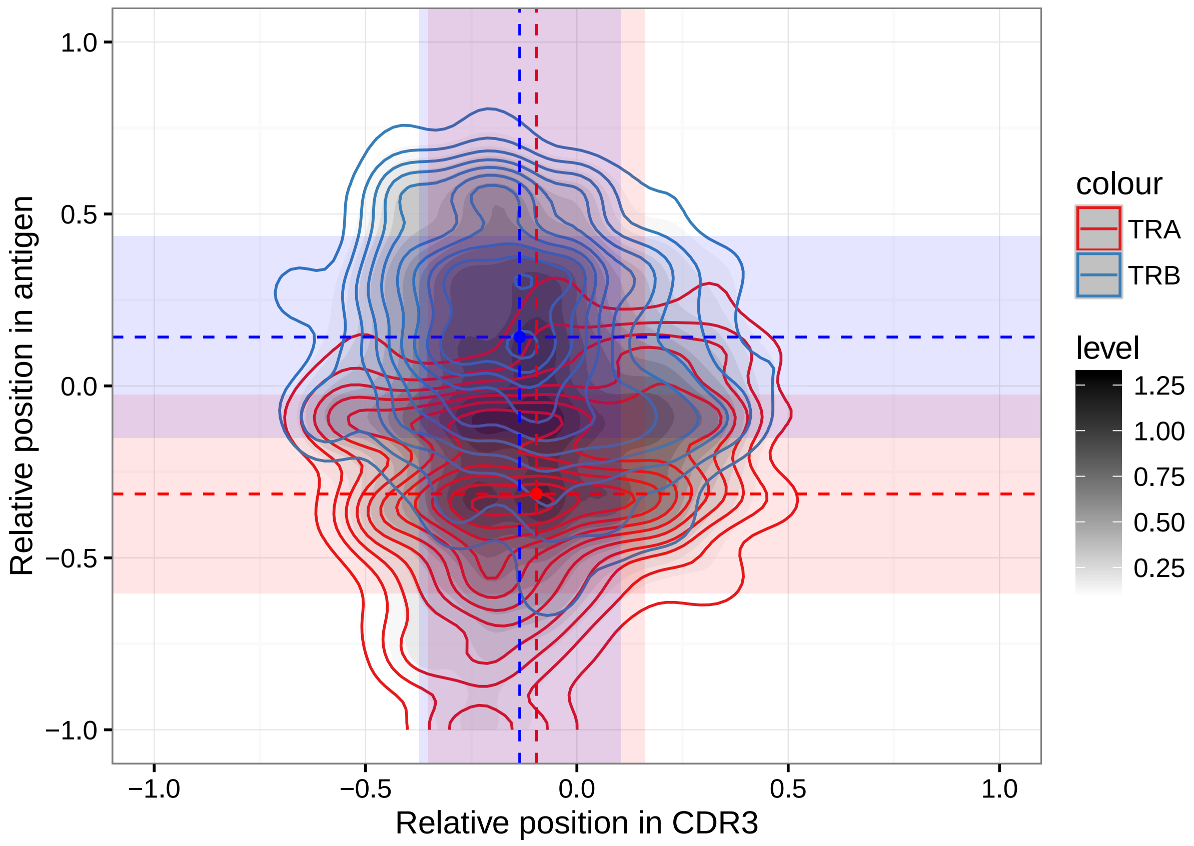
Relative contact position density and CDR3 length.

```
df.p1 <- ddply(subset(df, tcr_region == "CDR3"), .(pdb_id, len_tcr, tcr_chain, mhc_type, pos_tcr), summa
              contacts = sum(distance <= DIST_THRESHOLD))

df.p1 <- ddply(df.p1, .(len_tcr, tcr_chain, mhc_type, pos_tcr), summarize,
               contacts = mean(contacts))

df.p1 <- ddply(df.p1, .(len_tcr, tcr_chain, mhc_type), transform,
               mean_contacts_pos = sum(pos_tcr * contacts / sum(contacts)))

ggplot(subset(df.p1, contacts > 0),
       aes(y=len_tcr)) +
  geom_point(aes(x=pos_tcr - len_tcr / 2, size = contacts, color=contacts)) +
  geom_point(aes(x=mean_contacts_pos - len_tcr / 2), color="red", shape =3, size=3) +
  ylab("CDR3 length") +
  scale_x_continuous(name = "Position, relative to region center", limits=c(-7,7)) +
  scale_color_gradient(name = "Mean #contacts", low="#7fcdbb", high="#2c7fb8") +
  scale_size(guide=F)+
  facet_grid(tcr_chain ~ mhc_type) +
  theme_bw()
```



Contour map for contacts between alpha-beta CDR3 and antigen. Dashed lines show mean contact position in CDR3 (vertical) and antigen (horizontal), shaded areas show +/- SD.

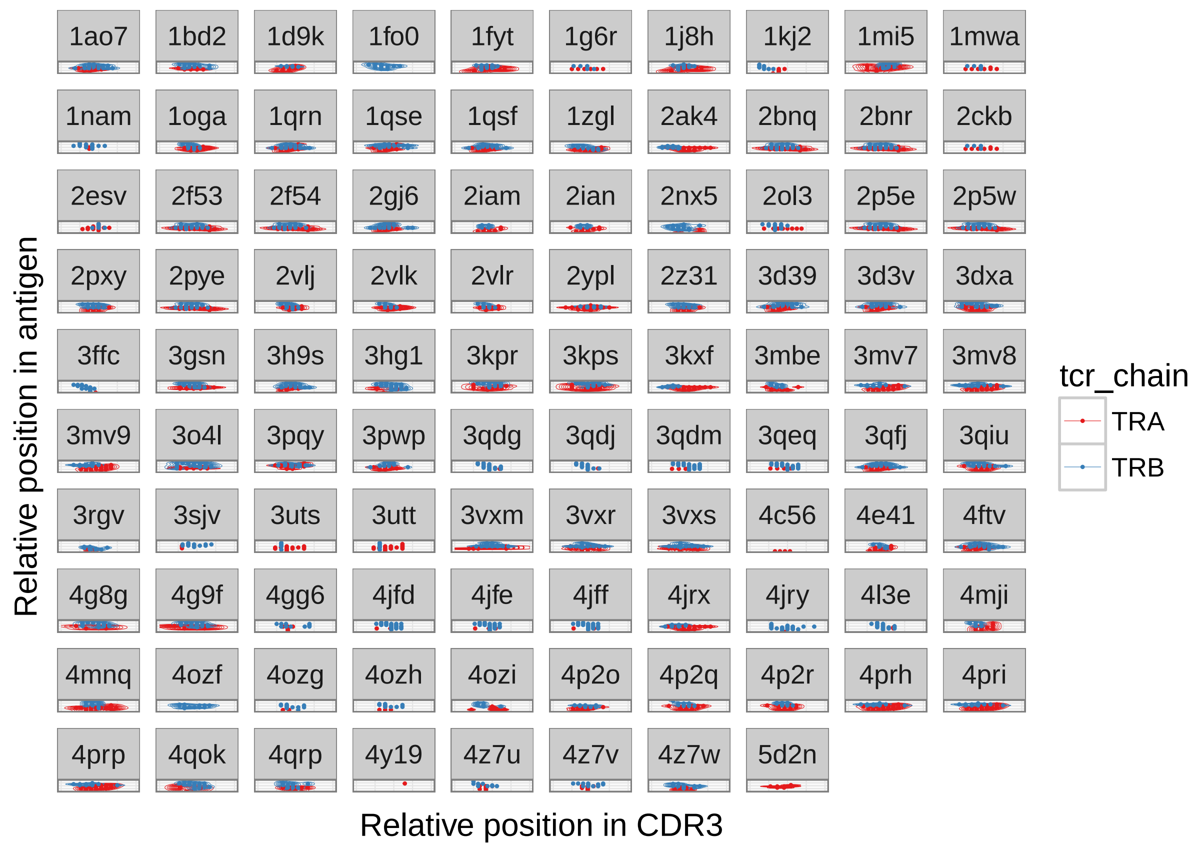
```
df.p2 <- ddply(subset(df, tcr_region == "CDR3" & distance <= DIST_THRESHOLD),
               .(pdb_id, tcr_chain, pos_tcr, pos_antigen), summarize,
               pos_norm_tcr = 2 * (pos_tcr - len_tcr / 2) / len_tcr,
               pos_norm_antigen = 2 * (pos_antigen - len_antigen / 2) / len_antigen,
               odd = len_tcr %% 2 == 0)

df.p3 <- ddply(df.p2, .(tcr_chain), summarize,
               pos_mean_tcr = mean(pos_norm_tcr),
               pos_sd_tcr = sd(pos_norm_tcr),
               pos_mean_antigen = mean(pos_norm_antigen),
               pos_sd_antigen = sd(pos_norm_antigen))

ggplot(df.p2, aes(x=pos_norm_tcr, y=pos_norm_antigen)) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", alpha=0.3) +
  geom_density_2d(aes(color=tcr_chain)) +
  scale_color_brewer(palette = "Set1") +
  annotate(geom="rect", ymin=df.p3[1,4]-df.p3[1,5], ymax=df.p3[1,4]+df.p3[1,5], xmin=-Inf, xmax=+Inf,
           fill="red", alpha=0.1) +
  geom_hline(yintercept=df.p3[1,4], color="red", linetype="dashed") +
  annotate(geom="rect", ymin=df.p3[2,4]-df.p3[2,5], ymax=df.p3[2,4]+df.p3[2,5], xmin=-Inf, xmax=+Inf,
           fill="blue", alpha=0.1) +
  geom_hline(yintercept=df.p3[2,4], color="blue", linetype="dashed") +
  annotate(geom="rect", xmin=df.p3[1,2]-df.p3[1,3], xmax=df.p3[1,2]+df.p3[1,3], ymin=-Inf, ymax=+Inf,
           fill="red", alpha=0.1) +
  geom_vline(xintercept=df.p3[1,2], color="red", linetype="dashed") +
  annotate(geom="rect", xmin=df.p3[2,2]-df.p3[2,3], xmax=df.p3[2,2]+df.p3[2,3], ymin=-Inf, ymax=+Inf,
           fill="blue", alpha=0.1) +
  geom_vline(xintercept=df.p3[2,2], color="blue", linetype="dashed") +
  annotate(geom="point", x=df.p3[1,2], y=df.p3[1,4],
           color="red") +
  annotate(geom="point", x=df.p3[2,2], y=df.p3[2,4],
           color="blue") +
  #annotate(geom="rect", ymin=df.p3[1,4]-df.p3[1,5], ymax=df.p3[1,4]+df.p3[1,5], xmin=-Inf, xmax=+Inf,
  #          fill="red", alpha=0.5) +
  scale_fill_gradient(low="white", high="black") +
  scale_x_continuous("Relative position in CDR3", limits=c(-1,1))+
  scale_y_continuous("Relative position in antigen", limits=c(-1,1))+
  theme_bw()
```



Contour plots of CDR3:antigen contact positions for each complex, colored by chain. Linear fitting is performed to bulk contact position plot showing some degree of dependency between CDR3 and antigen position.

```
library(scales)

ggplot(df.p2, aes(x=pos_norm_tcr, y=pos_norm_antigen, color=tcr_chain)) +
  geom_density_2d(size=0.1) +
  geom_point(size=0.1) +
  facet_wrap(~pdb_id) +
  scale_x_continuous("Relative position in CDR3", limits=c(-1,1), oob = rescale_none)+
  scale_y_continuous("Relative position in antigen", limits=c(-1,1), oob = rescale_none) +
  scale_color_brewer(palette = "Set1") +
  scale_fill_brewer(palette = "Set1") +
  theme_bw() +
  theme(axis.line=element_blank(),
        axis.text.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks=element_blank())
```

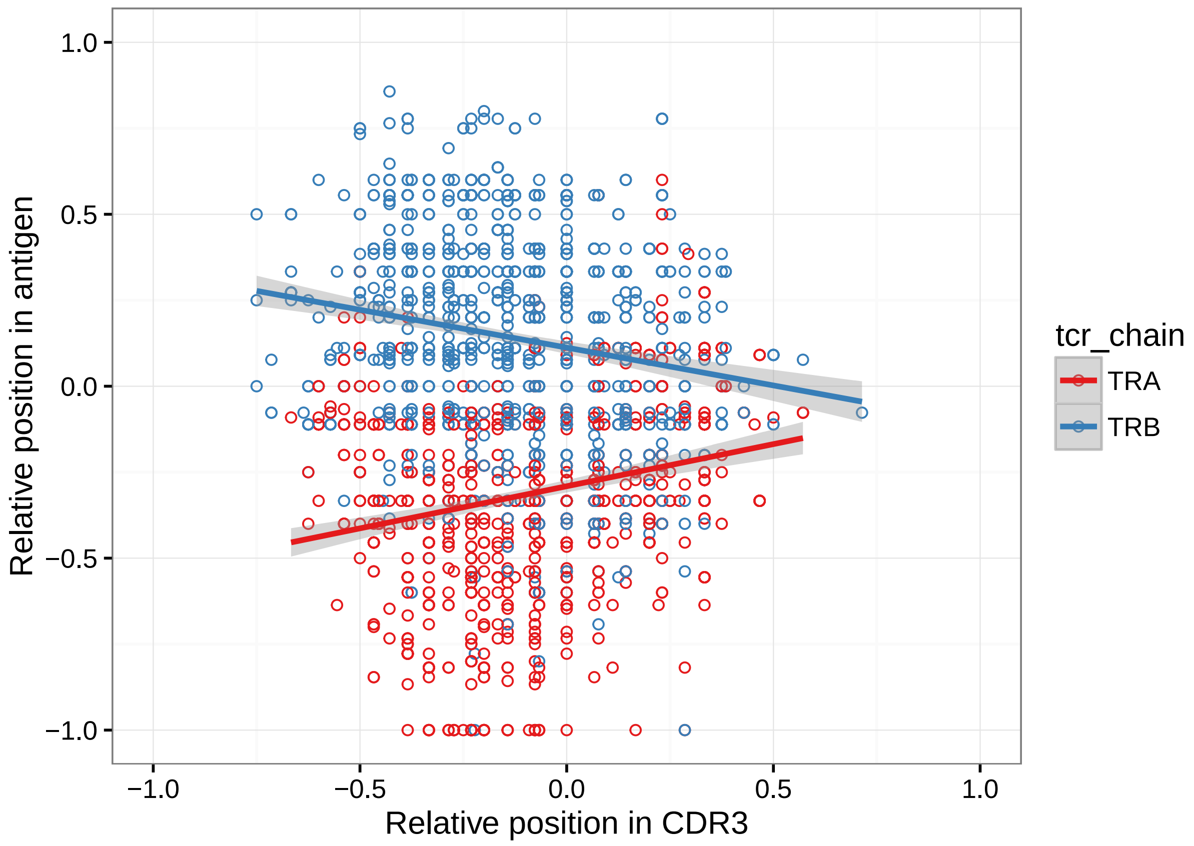
```
## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_density2d()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive
## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## missing value where TRUE/FALSE needed

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive

## Warning: Computation failed in `stat_density2d()`:
## bandwidths must be strictly positive
```



```
summary(lm(pos_norm_antigen ~ pos_norm_tcr * tcr_chain, df.p2))
```

```
##
## Call:
## lm(formula = pos_norm_antigen ~ pos_norm_tcr * tcr_chain, data = df.p2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1612 -0.1831  0.0022  0.2064  0.8464
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -0.290839   0.009304 -31.259  < 2e-16 ***
## pos_norm_tcr                 0.244986   0.034062   7.192 8.53e-13 ***
## tcr_chainTRB                 0.403109   0.013100  30.772  < 2e-16 ***
## pos_norm_tcr:tcr_chainTRB -0.464983   0.047887  -9.710  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2857 on 2339 degrees of freedom
## Multiple R-squared:  0.4027, Adjusted R-squared:  0.402
## F-statistic: 525.7 on 3 and 2339 DF,  p-value: < 2.2e-16
```

```
ggplot(df.p2, aes(x=pos_norm_tcr, y=pos_norm_antigen, color=tcr_chain)) +
  geom_point(shape = 21) + geom_smooth(method="lm") +
  scale_x_continuous("Relative position in CDR3", limits=c(-1,1), oob = rescale_none)+
  scale_y_continuous("Relative position in antigen", limits=c(-1,1), oob = rescale_none) +
  scale_color_brewer(palette = "Set1") +
  theme_bw()
```



Features of CDR3 residues that explain number of antigen contacts they form

This section of the manuscript deals with amino acid features of TCR and antigen contact residues, such as physical properties and pairing preference. Positional information is not used, however, we restrict our analysis to CDR3 region.

```
df.0 <- subset(df, tcr_region == "CDR3" & !(paste(pdb_id, tcr_chain) %in% no_contact_pdb)) # also remov
```

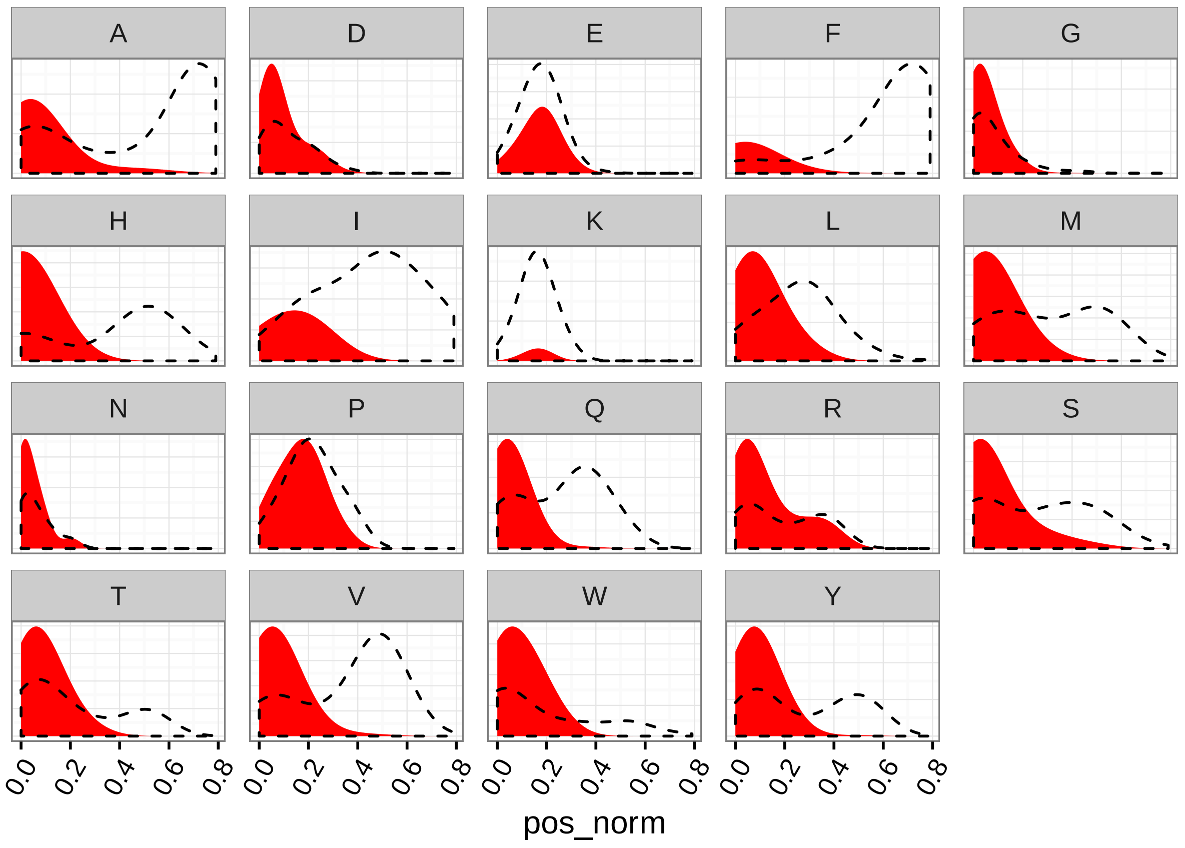
Most variance in the number of contacts of a given amino acid is explained by its positioning:

```
pos_mean_tcr <- df.p3$pos_mean_tcr
names(pos_mean_tcr) <- df.p3$tcr_chain

calc_pos_norm <- function(pos_tcr, len_tcr, tcr_chain) {
  half_len <- len_tcr[1] / 2
  ((pos_tcr - half_len) / half_len) ^ 2
}

df.c <- ddply(df.0, .(pdb_id, tcr_chain, aa_tcr, pos_tcr), summarize,
              pos_norm = calc_pos_norm(pos_tcr, len_tcr)[1],
              contacts = sum(distance <= DIST_THRESHOLD),
              count = 1)

aovsum <- summary(aov(contacts ~ aa_tcr * pos_norm, subset(df.c, aa_tcr != "C")))

var <- aovsum[[1]]["Sum Sq"]

print("Percent of variance explained")

print(var / sum(var) * 100)

aovres <- aov(pos_norm ~ aa_tcr, subset(df.c, aa_tcr != "C"))
summary(aovres)
```

Illustrate the above by plotting mean number of contacts vs CDR3 AA frequency at each relative CDR3 position.

```
df.c1 <- ddply(subset(df.c, aa_tcr != "C"), .(aa_tcr, pos_norm), summarize,
              contacts_sum = sum(contacts),
              count = sum(count))

ggplot(df.c1, aes(x=pos_norm)) +
  geom_density(aes(weight = contacts_sum), fill="red", color=NA) +
  geom_density(aes(weight = count), fill=NA, linetype="dashed") +
  #scale_x_continuous("Distance from CDR3 center", limits=c(0,1)) +
  ylab("") +
  facet_wrap(~aa_tcr, scales="free_y")+
  theme_bw() +
  theme(axis.line=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.text.x = element_text(angle = 60, hjust = 1))
```

```
## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density
## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density
## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density

## Warning in density.default(x, weights = w, bw = bw, adjust = adjust, kernel
## = kernel, : sum(weights) != 1 -- will not get true density
```



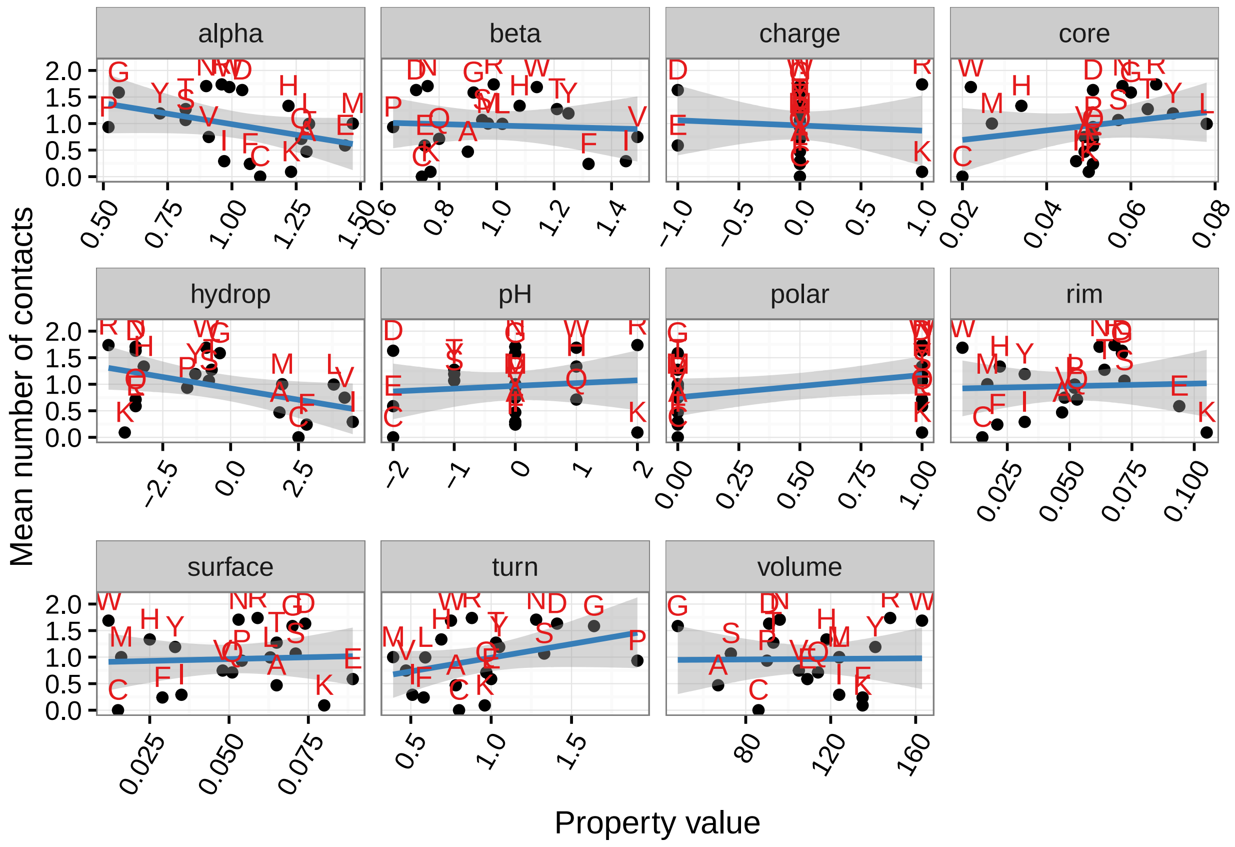
CDR3 AA features and number of contacts

The list of structural and basic physical properties of amino acids was taken from Elhanati et al 2015 (FIG. 13).

```
aa_prop <- read.table("aa_properties.txt", header=T)
```

None of aforementioned AA properties is correlated with mean number of contacts.

```
df.cc <- ddply(df.c, .(aa_tcr), summarize,
               contacts = sum(contacts),
               frequency = sum(count))
df.cc <- merge(df.cc, aa_prop, by.x = "aa_tcr", by.y="aa")

# Mean number of contacts vs property value

ggplot(df.cc, aes(x=value, y=contacts / frequency)) +
  geom_point() + geom_smooth(color="#377eb8", method="lm") +
  geom_text(aes(label=aa_tcr), color="#e41a1c", vjust=-0.5) +
  ylab("Mean number of contacts") + xlab("Property value") +
  facet_wrap(~property, scales="free_x") + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



```
# correlation with mean number of contacts
df.cor <- ddply(subset(df.cc, !(property %in% c("polar"))),
                        .(property), summarize, r = cor(contacts / frequency, value, method="pearson"))
df.cor$pvalue <- sapply(df.cor$r, function(r) calcpvalr(r))
df.cor$pvalue <- p.adjust(df.cor$pvalue)
df.cor$p.annot <- p.annot(df.cor$pvalue)

print(df.cor)
```

```
##    property           r    pvalue p.annot
## 1     alpha -0.37684463 0.4488850
## 2      beta -0.05998516 1.0000000
## 3    charge -0.07976082 1.0000000
## 4      core  0.24191454 1.0000000
## 5    hydrop -0.45642389 0.2084251
## 6        pH  0.10882205 1.0000000
## 7       rim  0.04631950 1.0000000
## 8   surface  0.05644740 1.0000000
## 9      turn  0.36931099 0.4488850
## 10   volume  0.01345887 1.0000000
```

```
# check polar separately
t.test(contacts / frequency ~ value, data = subset(df.cc, property == "polar"))
```

```
##
```

```
##  Welch Two Sample t-test
##
## data:  contacts/frequency by value
## t = -1.7718, df = 17.75, p-value = 0.09359
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.92056819  0.07868703
## sample estimates:
## mean in group 0 mean in group 1
##       0.7532883       1.1742289
```

Overall, these results suggest that AA placement in CDR3 is definitive for the number of contacts and this effect should be corrected for to account for individual AA features.

CDR3 AA contact number factors inferred using positional model

Fit total number of contacts of a given CDR3 AA using GLM. The formula is

`contacts ~ a(aa_tcr) * exp(-b(aa_tcr) * (pos_tcr - len_tcr/2)^2 / (len_tcr/2)^2)`

```
# Compute the table of number of CDR3 contacts
df.contpos1 <- ddply(df.0, .(pdb_id, tcr_chain, pos_tcr, aa_tcr, len_tcr), summarize,
              pos_norm = calc_pos_norm(pos_tcr, len_tcr, tcr_chain)[1],
              contacts = sum(distance <= DIST_THRESHOLD))

# Strip Cys at all
df.contpos1 <- subset(df.contpos1, aa_tcr != "C")
```

Run the fitting with GLM and Poisson distribution family (somewhat time consuming):

```
fit.c1 <- glm(contacts ~ aa_tcr + aa_tcr : pos_norm - 1, df.contpos1,
          family=poisson())

summary(fit.c1)
```

```
##
## Call:
## glm(formula = contacts ~ aa_tcr + aa_tcr:pos_norm - 1, family = poisson(),
##     data = df.contpos1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1507  -0.7612  -0.2358   0.2121   4.1612
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## aa_tcrA              0.71229    0.09804   7.265 3.73e-13 ***
## aa_tcrD              0.93381    0.12957   7.207 5.72e-13 ***
## aa_tcrE             -0.48196    0.35247  -1.367 0.171507
## aa_tcrF              1.26046    0.14411   8.746  < 2e-16 ***
## aa_tcrG              0.72981    0.05574  13.094  < 2e-16 ***
## aa_tcrH              1.64182    6.28816   0.261 0.794018
## aa_tcrI              1.22303    0.36325   3.367 0.000760 ***
## aa_tcrK           -2.47996    1.27230  -1.949 0.051273 .
## aa_tcrL            1.60210    0.10207  15.696  < 2e-16 ***
## aa_tcrM            1.63657    0.32267   5.072 3.94e-07 ***
## aa_tcrN            0.86293    0.10924   7.899 2.81e-15 ***
## aa_tcrP            0.86949    0.26192   3.320 0.000901 ***
## aa_tcrQ            1.31407    0.14386   9.134  < 2e-16 ***
## aa_tcrR            1.10456    0.10552  10.468  < 2e-16 ***
## aa_tcrS            0.96904    0.06641  14.591  < 2e-16 ***
## aa_tcrT            1.01318    0.08447  11.994  < 2e-16 ***
## aa_tcrV            1.52222    0.15118  10.069  < 2e-16 ***
## aa_tcrW            0.96080    0.15833   6.068 1.29e-09 ***
## aa_tcrY            1.48454    0.10981  13.519  < 2e-16 ***
## aa_tcrA:pos_norm  -6.18428    0.61383 -10.075  < 2e-16 ***
## aa_tcrD:pos_norm  -4.45935    1.14126  -3.907 9.33e-05 ***
## aa_tcrE:pos_norm  -0.29856    1.91759  -0.156 0.876275
## aa_tcrF:pos_norm  -9.53488    1.17824  -8.092 5.85e-16 ***
## aa_tcrG:pos_norm  -4.80813    0.73755  -6.519 7.07e-11 ***
## aa_tcrH:pos_norm -30.91806  759.62397  -0.041 0.967534
## aa_tcrI:pos_norm  -9.34198    1.97741  -4.724 2.31e-06 ***
## aa_tcrK:pos_norm   0.50949    7.35202   0.069 0.944752
## aa_tcrL:pos_norm -10.05591    0.76616 -13.125  < 2e-16 ***
## aa_tcrM:pos_norm -11.59845    3.50339  -3.311 0.000931 ***
## aa_tcrN:pos_norm  -6.45499    1.69932  -3.799 0.000146 ***
## aa_tcrP:pos_norm  -4.99247    1.37651  -3.627 0.000287 ***
## aa_tcrQ:pos_norm -13.44137    1.81316  -7.413 1.23e-13 ***
## aa_tcrR:pos_norm  -3.61823    0.60906  -5.941 2.84e-09 ***
## aa_tcrS:pos_norm  -6.12551    0.47222 -12.972  < 2e-16 ***
## aa_tcrT:pos_norm  -5.87292    0.65662  -8.944  < 2e-16 ***
## aa_tcrV:pos_norm  -9.91029    1.30806  -7.576 3.56e-14 ***
## aa_tcrW:pos_norm  -3.49034    1.01486  -3.439 0.000583 ***
## aa_tcrY:pos_norm  -8.26509    0.89453  -9.240  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4799.3  on 2383  degrees of freedom
## Residual deviance: 2168.9  on 2345  degrees of freedom
## AIC: 4874.3
##
## Number of Fisher Scoring iterations: 12
```

Check the fitting for the number of contacts for each of individual cases:

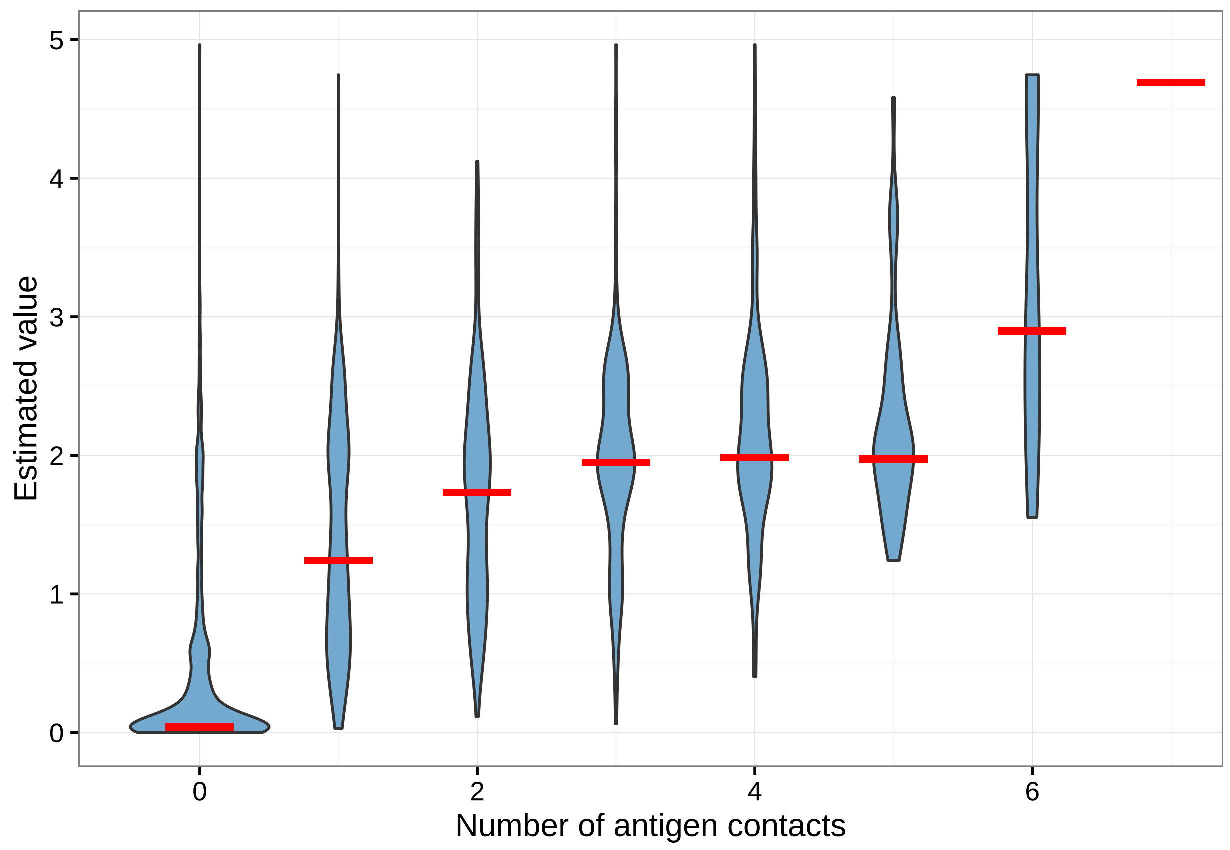
```
df.contpos1$contacts.fit <- fitted.values(fit.c1)

summary(aov(contacts ~ contacts.fit, df.contpos1))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## contacts.fit    1   2181  2181.2    1844 <2e-16 ***
## Residuals    2381   2816     1.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(lm(contacts ~ contacts.fit, df.contpos1))
```

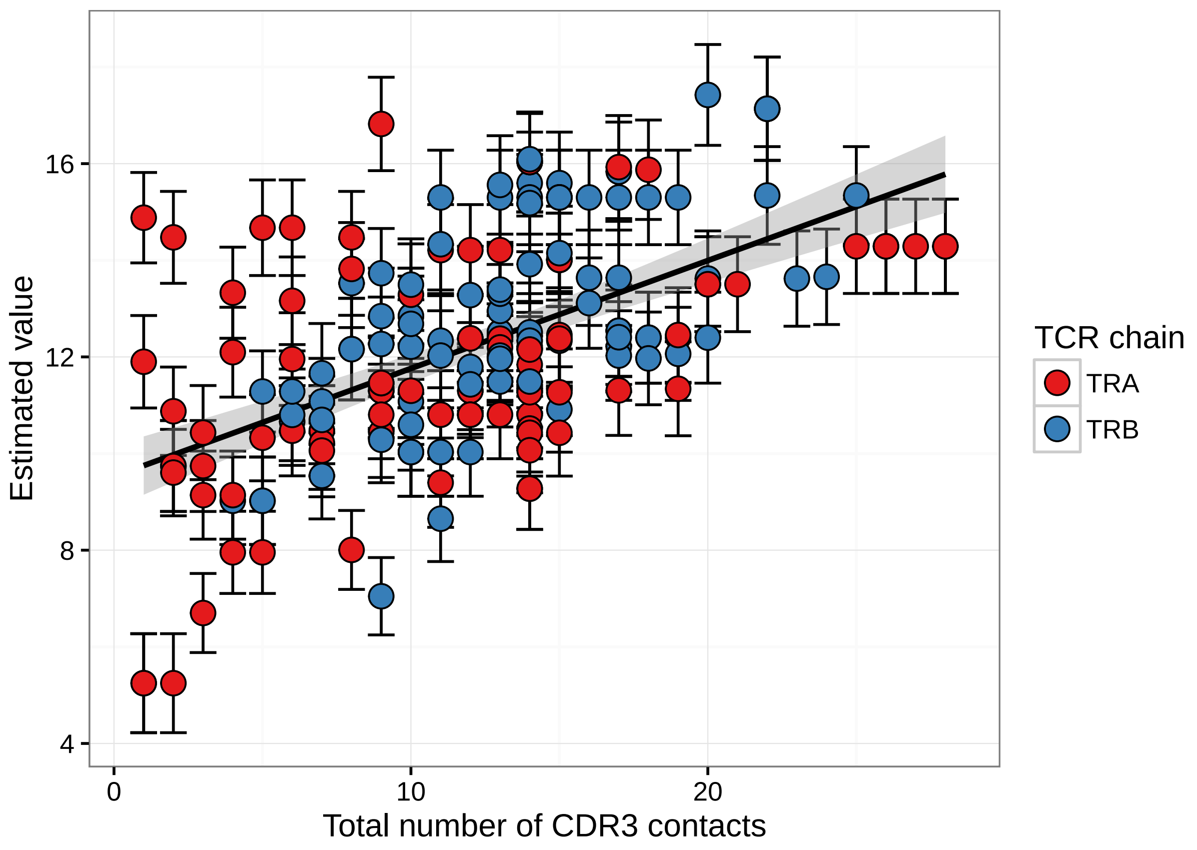
```
##
## Call:
## lm(formula = contacts ~ contacts.fit, data = df.contpos1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7439 -0.4338 -0.0849  0.2593  4.4787
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.05422    0.03105   1.746   0.0809 .
## contacts.fit  0.94485    0.02200  42.943   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.088 on 2381 degrees of freedom
## Multiple R-squared:  0.4365, Adjusted R-squared:  0.4362
## F-statistic:  1844 on 1 and 2381 DF,  p-value: < 2.2e-16
```

```r
ggplot() +
  geom_violin(data=df.contpos1,
              aes(x=contacts, y=contacts.fit, group = contacts), fill = "#74a9cf", width=1) +
  geom_point(data=ddply(df.contpos1, .(contacts), summarize, med=median(contacts.fit)),
             aes(x=contacts, y=med), color="red", size=15, shape=95) +
  xlab("Number of antigen contacts") + ylab("Estimated value") +
  theme_bw()
```



Check the fitting for total sum of contacts in each CDR3:

```
df.contpos1.sum <- ddply(df.contpos1, .(pdb_id, tcr_chain, len_tcr),
                         summarize,
                         contacts.sum = sum(contacts),
                         contacts.fit.sum = sum(contacts.fit))

ggplot(df.contpos1.sum, aes(contacts.sum, contacts.fit.sum)) +
  geom_errorbar(aes(ymin = contacts.fit.sum - sqrt(contacts.fit.sum / len_tcr),
                    ymax = contacts.fit.sum + sqrt(contacts.fit.sum / len_tcr))) +
  geom_smooth(method="lm", color="black") +
  geom_point(shape=21, size = 4, aes(fill = tcr_chain)) +
  scale_y_continuous("Estimated value") +
  scale_x_continuous("Total number of CDR3 contacts") +
  scale_fill_brewer("TCR chain", palette = "Set1") +
  theme_bw()
```



```
summary(lm(contacts.fit.sum ~ contacts.sum, df.contpos1.sum))
```

```
##
## Call:
## lm(formula = contacts.fit.sum ~ contacts.sum, data = df.contpos1.sum)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7265 -1.2747 -0.3728  1.1313  5.2837
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.52718    0.32772  29.071   <2e-16 ***
## contacts.sum  0.22335    0.02421   9.224   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.913 on 189 degrees of freedom
## Multiple R-squared:  0.3104, Adjusted R-squared:  0.3068
## F-statistic: 85.08 on 1 and 189 DF,  p-value: < 2.2e-16
```

```
summary(aov(contacts.fit.sum ~ contacts.sum, df.contpos1.sum))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## contacts.sum   1  311.5  311.45   85.08 <2e-16 ***
## Residuals    189  691.9    3.66
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now lets analyze the resulting coefficients for each amino acids. First, prepare coefficient table.

```
df.coef1 <- as.data.frame(summary(fit.c1)$coefficients)

df.coef1$type <- ifelse(grepl("pos_norm", rownames(df.coef1)), "b",
                        ifelse(grepl("aa_tcr", rownames(df.coef1)), "a", "c"))
df.coef1$aa <- sapply(rownames(df.coef1), function(x) strsplit(x,"")[[1]][7])

rownames(df.coef1) <- NULL
colnames(df.coef1) <- c("value", "std", "z", "P", "type", "aa")
df.coef1$z <- NULL
df.coef1$P <- NULL
```

Compute Z scores and P-values with respect to baseline (average coefficient value).

```
df.coef1.plt <- subset(df.coef1, type == "a")
df.coef1.plt$value <- df.coef1.plt$value - mean(df.coef1.plt$value)

# transform mean and std back
df.coef1.plt$value <- exp(df.coef1.plt$value)
df.coef1.plt$std <- df.coef1.plt$value * df.coef1.plt$std

# compute P-values
df.coef1.plt$z <- with(df.coef1.plt, (value - 1) / std)
df.coef1.plt$p <- 2*pnorm(-abs(df.coef1.plt$z))
df.coef1.plt$p.adj <- p.adjust(df.coef1.plt$p)
df.coef1.plt$p.txt <- p.annot(df.coef1.plt$p.adj)
df.coef1.plt
```

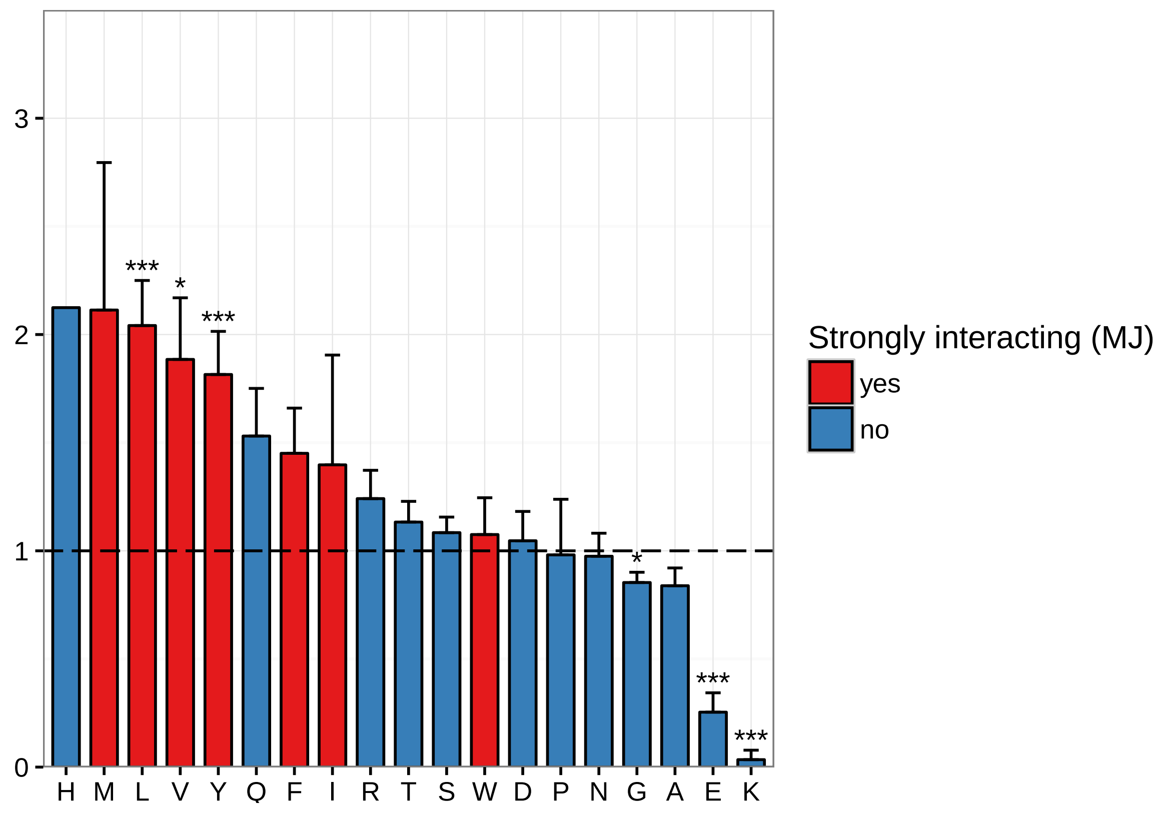
```
##         value         std type aa            z             p         p.adj
## 1  0.83855855  0.08221565    a  A  -1.96363412  4.957254e-02  5.452979e-01
## 2  1.04649876  0.13559618    a  D   0.34292091  7.316580e-01  1.000000e+00
## 3  0.25402680  0.08953658    a  E  -8.33149060  7.983078e-17  1.436954e-15
## 4  1.45077691  0.20907435    a  F   2.15606029  3.107896e-02  3.729475e-01
## 5  0.85338247  0.04756479    a  G  -3.08248015  2.052834e-03  2.873968e-02
## 6  2.12433333 13.35814079    a  H   0.08416840  9.329225e-01  1.000000e+00
## 7  1.39748426  0.50763874    a  I   0.78300616  4.336235e-01  1.000000e+00
## 8  0.03444762  0.04382787    a  K -22.03055487 1.467646e-107 2.788527e-106
## 9  2.04160986  0.20838655    a  L   4.99845050  5.779284e-07  9.824782e-06
## 10 2.11322836  0.68188586    a  M   1.63257289  1.025589e-01  9.230297e-01
## 11 0.97488853  0.10650100    a  N  -0.23578625  8.135985e-01  1.000000e+00
## 12 0.98130878  0.25702145    a  P  -0.07272241  9.420270e-01  1.000000e+00
## 13 1.53068515  0.22020416    a  Q   2.40996876  1.595389e-02  2.074005e-01
## 14 1.24135399  0.13098920    a  R   1.84254884  6.539491e-02  6.539491e-01
## 15 1.08402892  0.07199386    a  S   1.16716798  2.431425e-01  1.000000e+00
## 16 1.13294257  0.09570151    a  T   1.38913762  1.647909e-01  1.000000e+00
## 17 1.88487061  0.28495110    a  V   3.10534197  1.900592e-03  2.850887e-02
## 18 1.07513493  0.17022385    a  W   0.44138897  6.589314e-01  1.000000e+00
## 19 1.81518366  0.19933330    a  Y   4.08955075  4.322095e-05  6.915352e-04
##    p.txt
## 1
## 2
## 3    ***
## 4
## 5      *
## 6
## 7
## 8    ***
## 9    ***
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17     *
## 18
## 19   ***
```

Plot base coefficients (`a(tcr_aa)`):

```
df.coef1.plt$strong <- as.factor(ifelse(df.coef1.plt$aa %in% c("C","F","I","L","M","V","W","Y"), "yes",
df.coef1.plt$strong <- relevel(df.coef1.plt$strong, "yes")

# order AAs based on value
df.coef1.plt$aa = with(df.coef1.plt, factor(aa, levels=aa[order(-value)], ordered=TRUE))

ggplot(df.coef1.plt) +
  geom_bar(aes(x=aa, y=value, fill=strong), stat="identity", color="black", width=0.7) +
  geom_hline(yintercept = 1, color="black", linetype="longdash") +
  #geom_point(aes(x=aa, y=-0.1, color=strong), size=8, shape=15) +
  geom_errorbar(aes(x=aa, ymin=value, ymax=value+std), width=0.4) +
  geom_text(aes(x=aa, y = value+std + 0.05, label=p.txt)) +
  xlab("") + scale_y_continuous("",limits=c(-0, 3.5), expand = c(0,0)) +
  scale_fill_brewer("Strongly interacting (MJ)", palette = "Set1") +
  theme_bw()
```

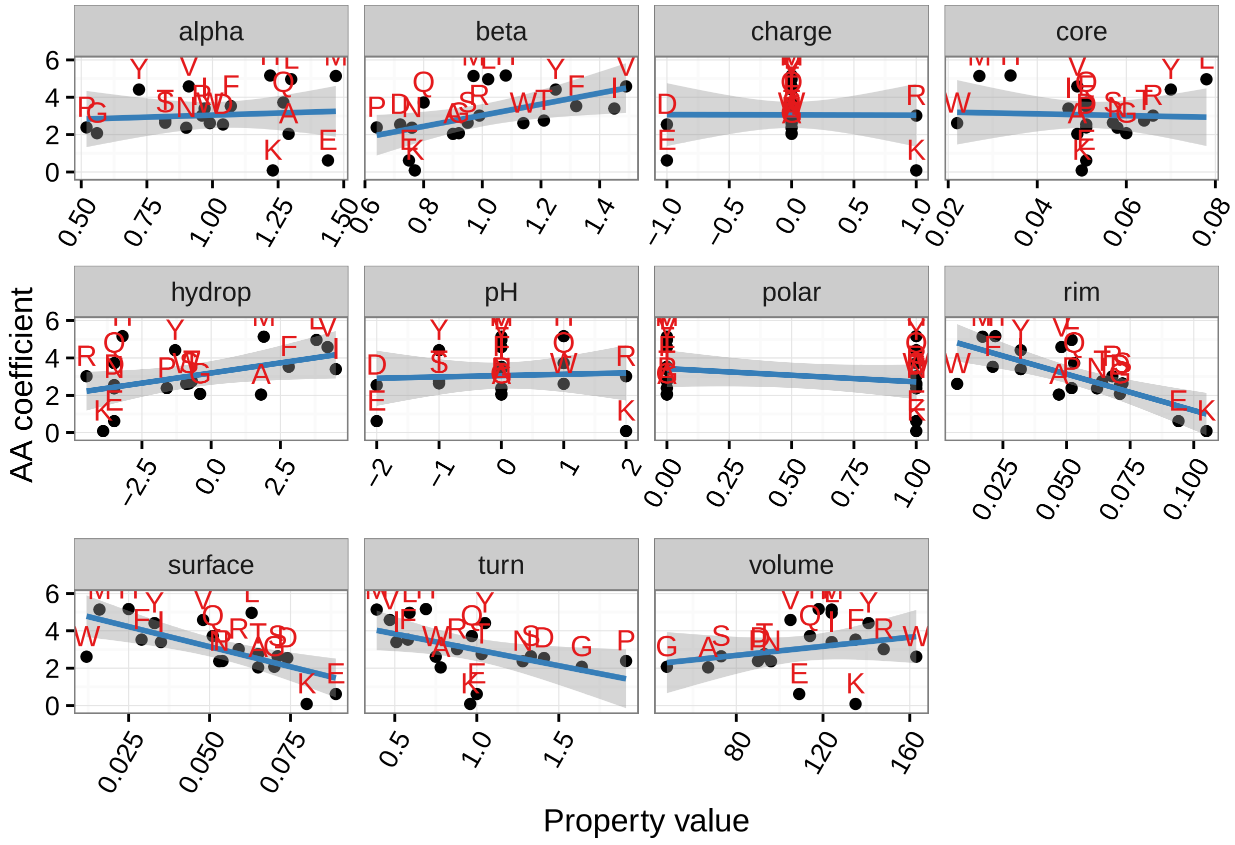
```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



Correlation of coefficients with AA features:

```
coef.aa_prop <- data.frame(aa=unique(aa_prop$aa))
coef.aa_prop$coeff <- sapply(coef.aa_prop$aa, function(x) coefficients(fit.c1)[paste("aa_tcr",x,sep = "
coef.aa_prop <- merge(coef.aa_prop, aa_prop)
coef.aa_prop <- subset(coef.aa_prop, !(aa %in% c("C")))

ggplot(coef.aa_prop, aes(x=value, y=exp(coeff))) +
  geom_point() + geom_smooth(color="#377eb8", method="lm") +
  geom_text(aes(label=aa), color="#e41a1c", vjust=-0.5) +
  ylab("AA coefficient") + xlab("Property value") +
  facet_wrap(~property, scales="free_x") + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



```r
df.cor <- ddply(subset(coef.aa_prop, !(property %in% c("charge", "polar"))),
                         .(property), summarize, r = cor(exp(coeff), value, method="pearson"))

df.cor$pvalue <- sapply(df.cor$r, function(r) calcpvalr(r, 19))
df.cor$pvalue <- p.adjust(df.cor$pvalue)

print(df.cor)
```

```
##   property           r      pvalue
## 1    alpha  0.08449567 1.000000000
## 2     beta  0.52511659 0.069317633
## 3     core -0.04534226 1.000000000
## 4   hydrop  0.45817323 0.117247942
## 5       pH  0.05618698 1.000000000
## 6      rim -0.71722817 0.001976531
## 7  surface -0.66867341 0.005970862
## 8     turn -0.49604399 0.088194231
## 9   volume  0.25220914 0.592002140
```

Analyze TCRs with known antigen specificity using the model

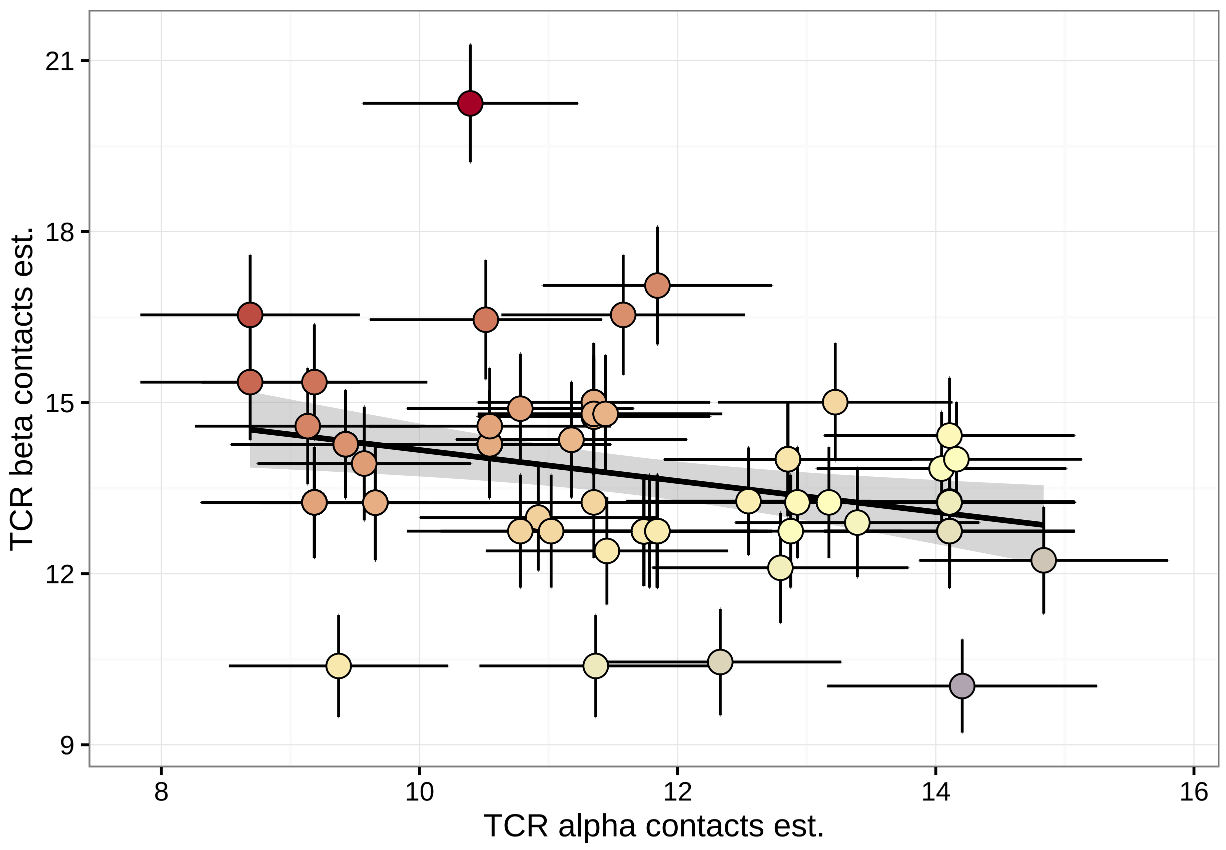
The following function is used to compute the number of contacts using the trained model:

```r
mdl.coef.df <- as.data.frame(summary(fit.c1)$coefficients)
mdl.coef <- mdl.coef.df$Estimate
```

```r
names(mdl.coef) <- rownames(mdl.coef.df)

calc_contact_num <- function(aa_tcr, pos_tcr, len_tcr, tcr_chain) {
  pos_norm <- calc_pos_norm(pos_tcr, len_tcr)
  exp(mapply(function(x, y, z) ifelse(x == "C", -Inf,
                                      mdl.coef[paste("aa_tcr", x, sep = "")] +
                                      mdl.coef[paste("aa_tcr", x, ":pos_norm", sep = "")] * y
           ), aa_tcr, pos_norm, len_tcr))
}

calc_contact_num_sum <- function(seq) {
  aa_tcr <- strsplit(seq, "")[[1]]
  len_tcr <- length(aa_tcr)
  ifelse(len_tcr == 0, 0,
         sum(calc_contact_num(aa_tcr, 1:len_tcr, len_tcr)))
}

#calc_contact_num_sum("CASSLAPGATNEKLFF")
```

Load paired TCR alpha beta data for *RPRGEVRFL* epitope from **PMID:20139278**:

```r
df.hsv2abpairs <- read.table("HSV-2_tcr_ab_pairs.txt", header = T, sep="\t", stringsAsFactors = F)

df.hsv2abpairs <- ddply(df.hsv2abpairs, .(cdr3.alpha, cdr3.beta), transform,
                  len.alpha = nchar(cdr3.alpha),
                  len.beta = nchar(cdr3.beta),
                  contacts.alpha = calc_contact_num_sum(cdr3.alpha),
                  contacts.beta = calc_contact_num_sum(cdr3.beta))

ggplot(df.hsv2abpairs, aes(x=contacts.alpha, y=contacts.beta)) +
  geom_smooth(method="lm", color="black") +
  geom_errorbar(aes(ymin = contacts.beta - sqrt(contacts.beta / len.beta),
                    ymax = contacts.beta + sqrt(contacts.beta / len.beta))) +
  geom_errorbarh(aes(xmin = contacts.alpha - sqrt(contacts.alpha / len.alpha),
                    xmax = contacts.alpha + sqrt(contacts.alpha / len.alpha))) +
  geom_point(size=4, aes(fill=contacts.alpha-contacts.beta), shape=21) +
  scale_fill_gradient2(guide = FALSE, low = "#a50026", mid = "#ffffbf", high = "#313695") +
  scale_x_continuous(name = "TCR alpha contacts est.") +
  scale_y_continuous(name = "TCR beta contacts est.") +
  theme_bw()
```



```
summary(lm(contacts.beta ~ contacts.alpha, df.hsv2abpairs))
```

```
##
## Call:
## lm(formula = contacts.beta ~ contacts.alpha, data = df.hsv2abpairs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9585 -0.9288 -0.0556  0.6137  6.1859
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    16.89005    1.14381  14.766  < 2e-16 ***
## contacts.alpha -0.27204    0.09697  -2.805  0.00644 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.453 on 73 degrees of freedom
## Multiple R-squared:  0.09731,    Adjusted R-squared:  0.08495
## F-statistic:  7.87 on 1 and 73 DF,  p-value: 0.006439
```

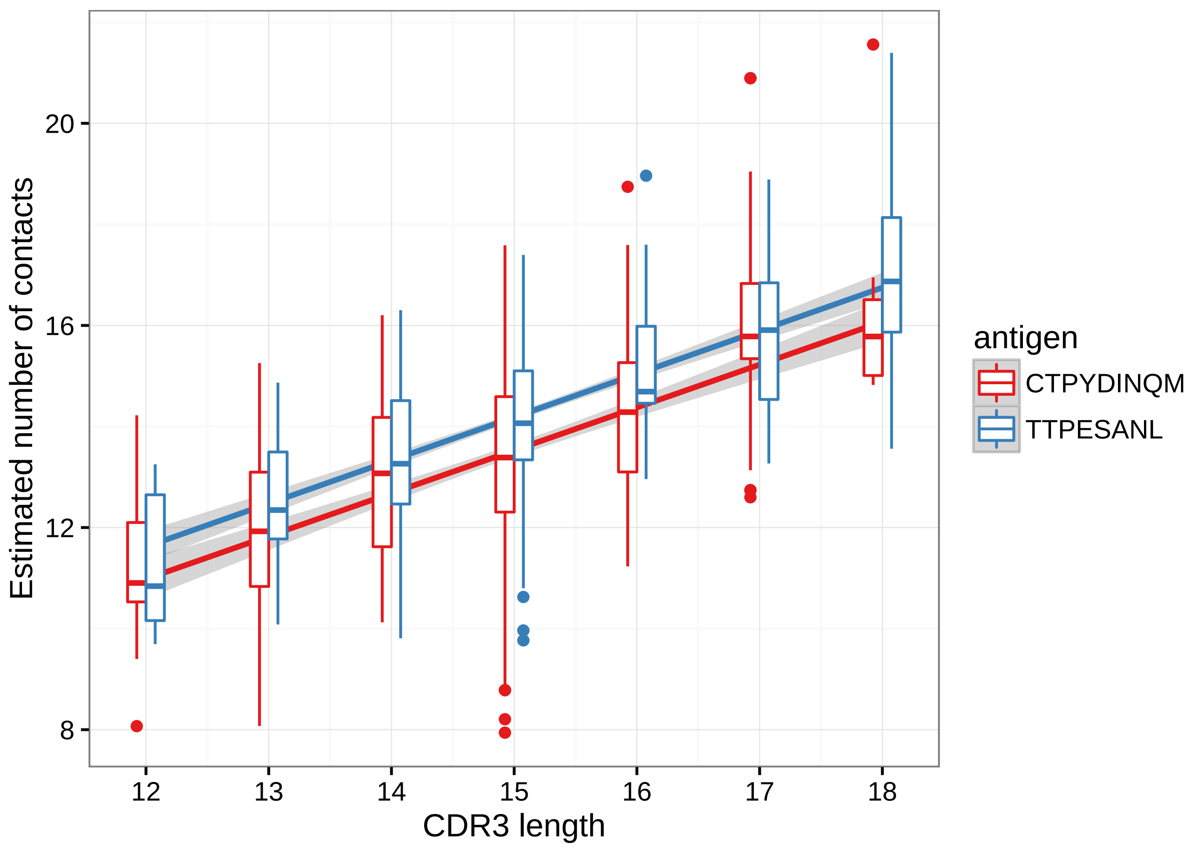
```
# normalized by length
summary(lm(contacts.beta / len.beta ~ I(contacts.alpha / len.alpha), df.hsv2abpairs))
```

```
##
```

```
## Call:
## lm(formula = contacts.beta/len.beta ~ I(contacts.alpha/len.alpha),
##     data = df.hsv2abpairs)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.220165 -0.042238  0.009163  0.052161  0.152135
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.22239    0.08802  13.888  < 2e-16 ***
## I(contacts.alpha/len.alpha) -0.30531    0.10369  -2.944  0.00434 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07317 on 73 degrees of freedom
## Multiple R-squared:  0.1062, Adjusted R-squared:  0.09392
## F-statistic:  8.67 on 1 and 73 DF,  p-value: 0.004337
```

SIV-specific TCRs from David Price et al. monkey study. Differences between CM9- and TL8-specific TCRs:

```r
df.tcr_siv <- read.table("siv_tcrs.txt", header=T, stringsAsFactors = F)

df.tcr_siv <- ddply(df.tcr_siv, .(cdr3), transform,
                  len = nchar(cdr3),
                  contacts = calc_contact_num_sum(cdr3))

df.tcr_siv <- subset(df.tcr_siv, len >= 12 & len <= 18)

ggplot(df.tcr_siv,
       aes(x=len, y = contacts, color=antigen)) +
  geom_smooth(method="lm") +
  geom_boxplot(aes(group=interaction(len, antigen)), width=0.4) +
  scale_color_brewer(palette = "Set1") +
  scale_x_continuous("CDR3 length", breaks=12:18) +
  ylab("Estimated number of contacts") +
  theme_bw()
```



```r
summary(aov(contacts ~ len + antigen, df.tcr_siv))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)    
## len             1   1057  1057.4  488.98 <2e-16 ***
## antigen         1    162   161.6   74.74 <2e-16 ***
## Residuals    1529   3306     2.2                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
df.pvals <- data.frame(len=12:18)
df.pvals$pvals <- sapply(df.pvals$len, function(x) wilcox.test(contacts ~ antigen,
                                                               subset(df.tcr_siv, len == x), exact=F)[[
print(df.pvals)
```

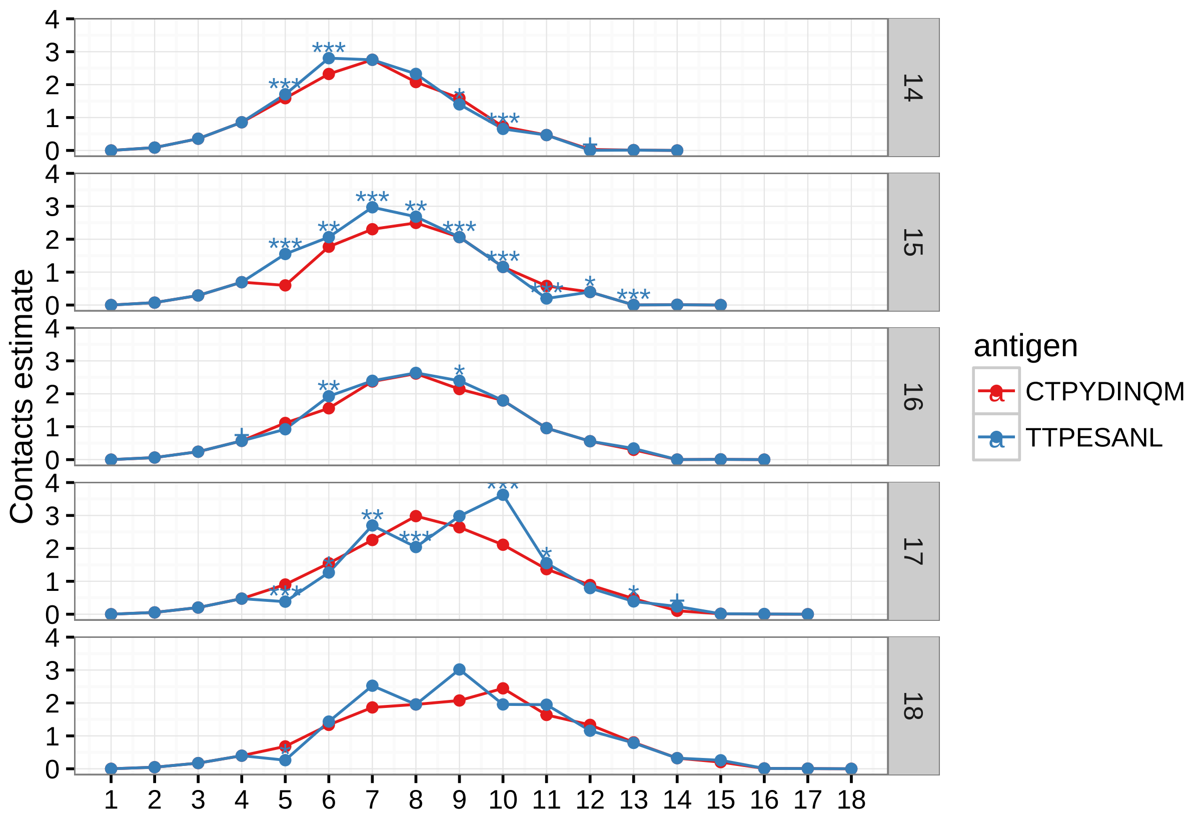
```
##   len        pvals
## 1  12 8.006507e-01
## 2  13 3.622960e-01
## 3  14 1.010461e-01
## 4  15 3.379839e-16
## 5  16 1.445509e-02
## 6  17 7.062190e-01
## 7  18 1.893744e-01
```

Comparing contact profiles for each CDR3 length between two epitopes:

```
df.tcr_siv.flat <- data.frame(cdr3.beta = character(), aa_tcr = character(),
                              pos_tcr = integer(),
                              antigen.species = character(), antigen.epitope = character())

for (i in 1:nrow(df.tcr_siv)) {
  .row <- df.tcr_siv[i, ]
  .aa_tcr <- strsplit(.row$cdr3, split="")[[1]]
  .df.ext <- data.frame(cdr3 = .row$cdr3,
                        aa_tcr = .aa_tcr,
                        pos_tcr = 1:length(.aa_tcr),
                        len_tcr = length(.aa_tcr),
                        antigen = .row$antigen)
  df.tcr_siv.flat <- rbind(df.tcr_siv.flat, .df.ext)
}

df.tcr_siv.flat$contacts <- with(df.tcr_siv.flat,
                                 mapply(function(x,y,z) calc_contact_num(x,y,z), aa_tcr, pos_tcr, len_t

get_contacts <- function(pos, len) {
  subset(df.tcr_siv.flat, len_tcr == len[1] & pos_tcr == pos[1] & antigen == "CTPYDINQM")$contacts
}

df.tcr_siv.plt <- ddply(subset(df.tcr_siv.flat, len_tcr %in% c(14:18)),
                        .(pos_tcr, len_tcr, antigen), summarize,
                        contacts.mean=median(contacts),
                        pval=wilcox.test(contacts, get_contacts(pos_tcr, len_tcr))[[3]])
```

```
## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties
## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties
## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties
## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(contacts, get_contacts(pos_tcr, len_tcr)):
## cannot compute exact p-value with ties
```

```r
df.tcr_siv.plt$ptxt <- p.annot(ifelse(is.nan(df.tcr_siv.plt$pval), 1, df.tcr_siv.plt$pval))

ggplot(df.tcr_siv.plt, aes(x=pos_tcr, y=contacts.mean,
                          color=antigen)) +
  geom_line() +
  geom_point() +
  geom_text(aes(y = contacts.mean+0.2, label = ptxt)) +
  scale_color_brewer(palette = "Set1") +
  ylab("Contacts estimate") +
  scale_x_continuous("", breaks=1:18) +
  facet_grid(len_tcr~., scales="free_x") +
  theme_bw()
```

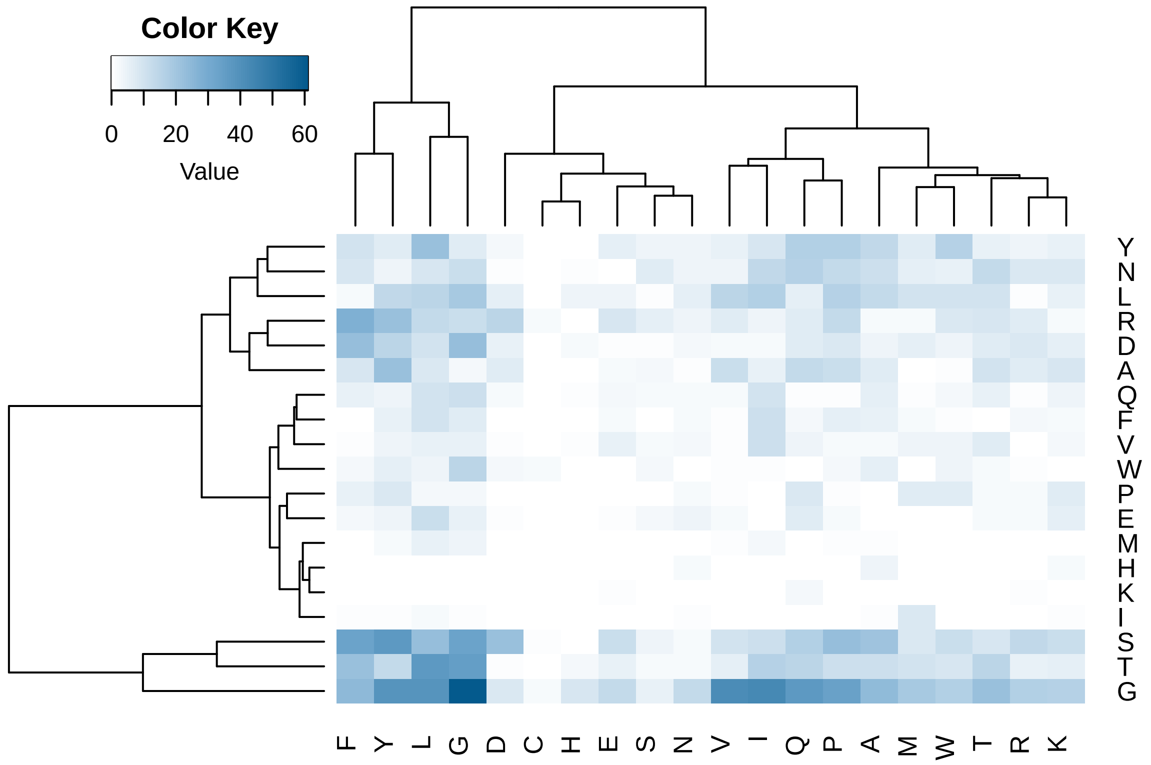


Application of findings to HIV escape epitope data

Pairwise contact preferences

```
df.cm <- ddply(df.0, .(aa_tcr, aa_antigen), summarize,
               contacts_sum = sum(distance <= DIST_THRESHOLD),
               total = length(distance))

contact_mat <- dcast(df.cm, aa_tcr ~ aa_antigen, value.var = "contacts_sum")
contact_mat[is.na(contact_mat)] <- 0
rownames(contact_mat) <- contact_mat[,1]
contact_mat[,1] <- NULL
contact_mat <- contact_mat[-2, ]
contact_mat <- as.matrix(contact_mat)

colgen <- function(aas, p, low, mid, hi) {
  panel <- colorpanel(100, low, mid, hi)
  df.1 <- merge(data.frame(aa=aas), subset(aa_prop, property == p))
  df.1$value <- as.integer((df.1$value - min(df.1$value)) / (max(df.1$value) - min(df.1$value)) * 99 + 
  return(panel[df.1$value])
}

cn <- colnames(contact_mat)
colgen_cols <- ifelse(cn %in% c("A", "C", "I", "L", "M", "F", "W", "V"), "blue",
                  ifelse(cn %in% c("G", "H", "P", "S", "T", "Y"), "red", "yellow"))
```

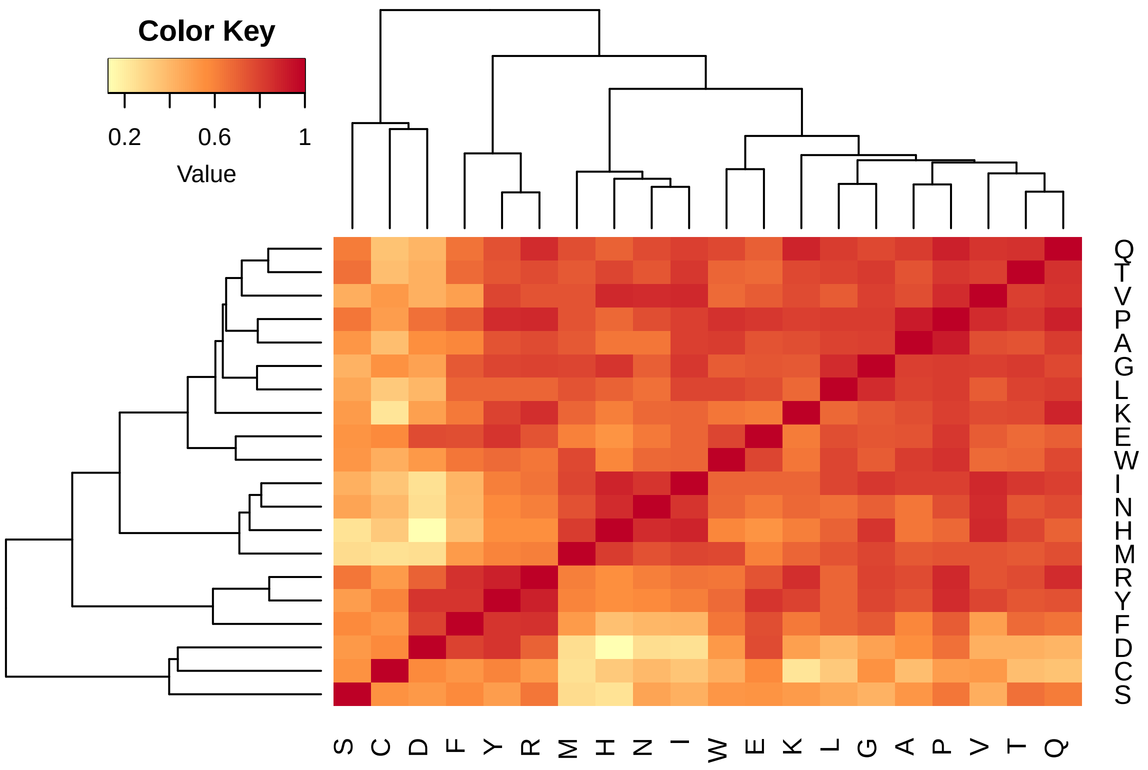
```r
heatmap.2(contact_mat,
          col=colorpanel(100, "white", "#74a9cf", "#045a8d"),
          #distfun = function(x) as.dist(1-cor(t(x))/2),
          #hclustfun = function(d) hclust(d, method="ward"),
          #ColSideColors = colgen_cols,
          #RowSideColors = colgen(rownames(mat.cm), "core", "white", "grey", "black"),
          density.info = "none", trace="none")
```



```r
calc_dist_1 <- function(x, y) cor(contact_mat[,x], contact_mat[,y])

df.ag.sim <- expand.grid(colnames(contact_mat), colnames(contact_mat))

df.ag.sim$r <- with(df.ag.sim, mapply(function(x,y) calc_dist_1(x, y), Var1, Var2))

mat.ag.sim <- dcast(df.ag.sim, Var1~Var2)
```

```
## Using r as value column: use value.var to override.
```

```r
rownames(mat.ag.sim) <- mat.ag.sim[,1]
mat.ag.sim[,1] <- NULL
mat.ag.sim <- as.matrix(mat.ag.sim)

heatmap.2(mat.ag.sim,
          col=colorpanel(100, "#ffffb2", "#fd8d3c", "#bd0026"),
         # distfun = function(x) as.dist(1-cor(t(x))/2),
         # hclustfun = function(d) hclust(d, method="ward"),
         # ColSideColors = colgen_cols,
          #RowSideColors = colgen(rownames(mat.cm), "hydrop", "white", "grey", "black"),
          density.info = "none", trace="none")
```



Analyzis of NIAID HIV db

Finally, we have applied our findings to study escape epitope variants from NIAID HIV databases database. The database was filtered to retain only variants with a single amino acid substitution. Escape variants were selected according to **E** (documented escape) and **NSF** (non-susceptible form) flags, and were further partitioned into three groups:

- **hla** variants that abrogate MHC binding (**DHB** flag, diminished HLA binding or increased off-rate)
- **tcr** variants that do not bind or show decreased binding to TCR (**TCR** flag).
- **other** unclassified escape variants

As only 8,9,10 and 11-mer epitopes had reported **tcr** escape variants, we've limited our analysis to this group.

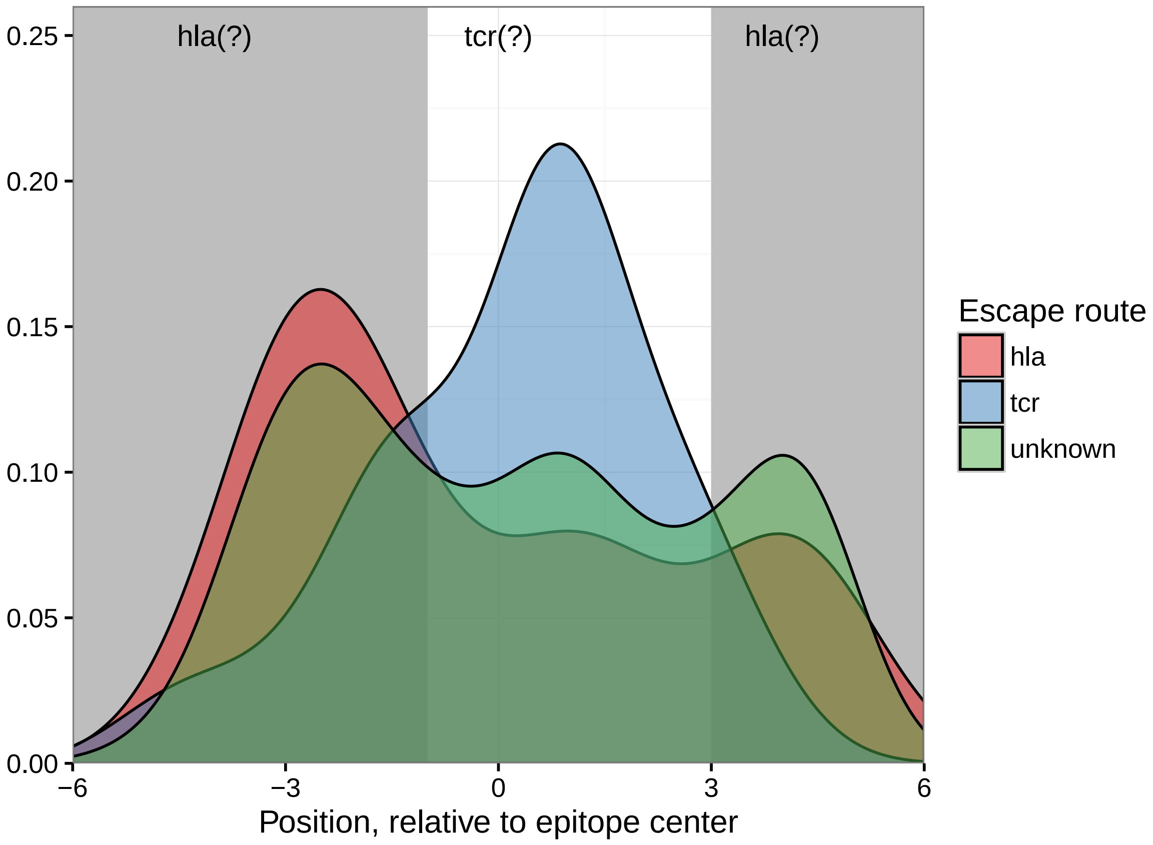
```
df.epi <- read.csv("ctl_variant.csv", header=T)

df.epi <- df.epi[grep("^[ACDEFGHIKLMNPQRSTVWY]\\d+[ACDEFGHIKLMNPQRSTVWY]$", df.epi$Mutation_epitope),]
df.epi <- df.epi[grep("^[ACDEFGHIKLMNPQRSTVWY]+$", df.epi$Epitope),]
len <- function(x) nchar(as.character(x))

df.epi$codes <- sapply(df.epi$Mutation_Type_Code, function(x) strsplit(as.character(x),", "))

isescape <- function(x) "E" %in% x || "LE" %in% x || "IE" %in% x || "NSF" %in% x

df.epi$type <- sapply(df.epi$codes,
                       function(x) ifelse("TCR" %in% x, "tcr",
                                   ifelse("DHB" %in% x, "hla",
                                   ifelse(isescape(x), "unknown",
                                          "non-escape"
                                          ))))

df.epi <- subset(df.epi, type != "non-escape")

df.epi <- data.frame(seq = df.epi$Epitope,
                     aa_from = as.character(sapply(df.epi$Mutation_epitope,
                                                   function(x) substr(x,1,1))),
                     aa_to = as.character(sapply(df.epi$Mutation_epitope,
                                                 function(x) substr(x,len(x),len(x)))),
                     pos = sapply(df.epi$Mutation_epitope,
                                  function(x) as.integer(substr(x,2,len(x)-1))),
                     type = as.factor(df.epi$type),
                     elen = sapply(df.epi$Epitope,
                                   function(x) len(x))
                     )

df.epi$pos_adj <- apply(df.epi, 1, function(x) as.numeric(x[4])-as.numeric(x[6]) / 2)

summary(df.epi)
```

```
##          seq          aa_from        aa_to           pos
##  SLYNTVATL : 35   K      : 71   R      : 53   Min.   : 1.000
##  FLKEKGGL  : 29   R      : 60   K      : 52   1st Qu.: 2.000
##  KRWIILGLNK: 23   T      : 52   T      : 45   Median : 5.000
##  TSTLQEQIGW: 19   V      : 49   A      : 44   Mean   : 4.979
##  KEKGGLEGL : 16   I      : 47   V      : 41   3rd Qu.: 7.000
##  TAFTIPSI  : 12   Y      : 46   L      : 39   Max.   :14.000
##  (Other)   :451   (Other):260   (Other):311
##       type           elen           pos_adj
##  hla    : 70   Min.   : 8.00   Min.   :-9.0000
##  tcr    : 20   1st Qu.: 9.00   1st Qu.:-2.5000
##  unknown:495   Median : 9.00   Median : 0.5000
##                Mean   : 9.59   Mean   : 0.1846
##                3rd Qu.:10.00   3rd Qu.: 2.5000
##                Max.   :20.00   Max.   : 6.5000
##
```

As expected, **tcr**-mediate escape mutations are clustered in the central part of antigen (showing a two-peak picture resembling TCR contact positioning plot above), while **hla**-mediate escape mutations are clustered on **N** and **C** termini.

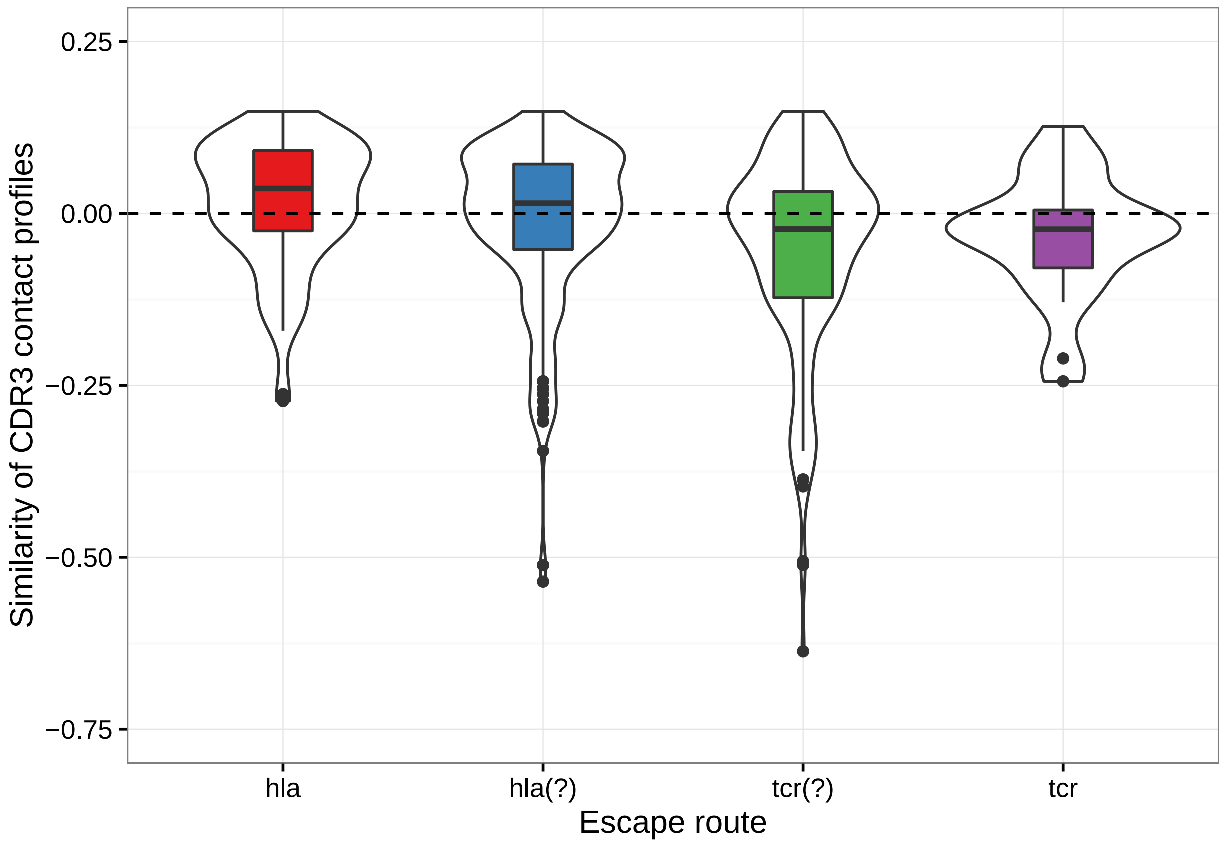
```
ggplot(df.epi, aes(x=pos_adj, fill=type)) +
  annotate("rect", xmin=-Inf, xmax=-1, ymin=0, ymax=Inf, fill="grey") +
  annotate("rect", xmin=3, xmax=Inf, ymin=0, ymax=Inf, fill="grey") +
  geom_density(alpha=0.5) +
  annotate("text", x=-4, y=0.25, label = "hla(?)") +
  annotate("text", x=-0, y=0.25, label = "tcr(?)") +
  annotate("text", x=4, y=0.25, label = "hla(?)") +
  xlab("Position, relative to epitope center") +
  ylab("") +
  scale_fill_brewer(name = "Escape route", palette = "Set1") +
  scale_x_continuous(limits=c(-6,6), expand=c(0,0)) +
  scale_y_continuous(limits=c(0,0.26), expand=c(0,0)) +
  theme_bw()
```

```
## Warning: Removed 3 rows containing non-finite values (stat_density).
```



Next, we have tested the hypothesis that **tcr**-mediated escape variants are biased in the direction of mutations that substantially change TCR recognition profile (*TRP*, i.e. a column from the contact amino acid matrix described above). We have therefore computed correlation coefficients between *TRPs* of original and substituted amino acids and compared them between **hla**, **tcr** and **other** mutation groups. We have also taken an advantage of positioning data and partitioned the **other** group into **other.tcr** and **other.hla** based on whether the variant was in *[-2, 2]* region in respect to antigen center or not.

```
calc_dist_1 <- function(x , y) cor(contact_mat[,x], contact_mat[,y])

mean_dist <- median(mapply(function(x,y) ifelse(x==y, NA, calc_dist_1(x,y)),
                     as.character(df.epi$aa_from),
                     as.character(df.epi$aa_to)), na.rm=T)

calc_dist <- function(row) calc_dist_1(row[2], row[3]) - mean_dist

df.epi$type2 <- factor(apply(df.epi, 1, function(x) ifelse(x[5] == "unknown",
                                                 ifelse(as.numeric(x[7]) >= -1 & as.numeric(x[7]) <= 3
                                                        "tcr(?)", "hla(?)"),
                                                      x[5])), levels=c("hla","hla(?)","tcr(?)","tcr"))

df.epi$dist <- apply(df.epi, 1, calc_dist)

ggplot(df.epi, aes(x = type2, group=type2, y=dist)) +
  geom_violin() +
  geom_boxplot(aes(fill=type2), width=0.3) +
  geom_hline(yintercept = 0, linetype="dashed") +
  xlab("Escape route") +
  ylab("Similarity of CDR3 contact profiles") +
  scale_fill_brewer(guide=F, palette = "Set1") +
  scale_y_continuous(limits=c(-0.75,0.25)) +
  theme_bw()
```



There was indeed a clear trend showing that **hla**-mediated escape mutations resulted in similar *TRPs*, while **tcr**-mediated mutations had least similar *TRPs*. Statistical details are given below.

```
kruskal.test(dist ~ type2, df.epi)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  dist by type2
## Kruskal-Wallis chi-squared = 18.788, df = 3, p-value = 0.0003025
```

```r
wilcox.test(subset(df.epi, type2 == "hla(?)")$dist, subset(df.epi, type2 == "tcr(?)")$dist)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  subset(df.epi, type2 == "hla(?)")$dist and subset(df.epi, type2 == "tcr(?)")$dist
## W = 34432, p-value = 0.001106
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(subset(df.epi, type2 == "hla")$dist, subset(df.epi, type2 == "tcr")$dist)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  subset(df.epi, type2 == "hla")$dist and subset(df.epi, type2 == "tcr")$dist
## W = 933.5, p-value = 0.02353
## alternative hypothesis: true location shift is not equal to 0
```

The analysis described here can be further extended to immunogenic and non-immunogenic mutated self-peptides from cancer peptidomes once a database of cancer epitopes similar to NIAID HIV databases will become available.